Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Паралелизација на алгоритми за предобработка и класификација на податоци

Докторска дисертација

Ефтим Здравевски

Ментор:
Проф. д-р Андреа Кулаков

Скопје, Август 2017

Комисија

Акад. проф. д-р Љупчо Коцарев, претседател
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Проф. д-р Андреа Кулаков, ментор
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Проф. д-р Владимир Трајковиќ, член
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Вонр. проф. д-р Боро Јакимовски, член
Факултет за информатички науки и компјутерско инженерство
Универзитет Св. "Кирил и Методиј", Скопје, Македонија

Проф. д-р Сашо Коцески, надворешен член
Факултет за информатика
Универзитет "Гоце Делчев", Штип, Македонија

Паралелизација на алгоритми за предобработка и класификација на податоци

Докторска дисертација

Ефтим Здравевски

Апстракт

Во текот на секојдневното работење компаниите, организациите и различните институции собираат големи количества на разнородни податоци. Кај многу од нив, успешноста на бизнисот директно зависи од нивната способност да ги анализираат собраните податоци. За оваа цел, може да се употреби машинско учење, кое овозможува автоматска анализа на огромни количини на податоци и изведување различни заклучоци. Пред да може да се употреби алгоритам за машинско учење, односно да се моделираат податоците, најчесто претходат фазите на разбирање на бизнисот и податоците, предобработка и одбирање на атрибути. Овие фази одземаат многу време, бидејќи во голема мера зависат од природата на податоците кои се обработуваат и процесот при кој биле собирани. При предобработката потребно е да се генерираат атрибути кои го опишуваат различните аспекти на бизнис проблемот и кои треба да им овозможат на алгоритмите за машинско учење градење на квалитетни модели. Притоа, генерирањето на атрибути треба да ги земе во предвид типовите на податоци (нумерички, номинални или текстуални) и начините на кои тие се генерирани (при случување на одредени настани или со константна рата). Со одбирањето на атрибути се идентификуваат најрелевантните атрибути и се овозможува алгоритмот за машинско учење да се посвети на тие аспекти на податоците кои се најкорисни за анализа и идно предвидување, со што се постигнува подобрување на предиктивните перформанси, забрзување на алгоритмите и намалување на нивната пресметковна и мемориска комплексност.

Во последните години, компаниите и организациите собираат податоци со забрзано темпо и значително зголемен волумен, па нивната обработка и анализа стануваат многу покомплексни. Причините за толкавиот раст на податоците лежат во: сеприсутните мобилни уреди и сензори; софтверски дневници (logs, анг.); интеракцијата на корисниците со бесконечниот број на веб страници, апликации и мултимедијални содржини; социјалните мрежи итн. Се проценува дека над 90% од податоците во светот се генерирани само во последните неколку години. Во зависност од конкретниот контекст, употребата на податоците се соочува со барем некои од следниве предизвици: собирање, анализа, обработка, пребарување, визуализација, пренос, безбедност, приватност, генерирање извештаи, превземање акции, давање препораки и совети за поддршка на разни одлуки итн. Кога податоците се толку големи што овие предизвици не можат или многу тешко може да се совладаат со традиционални алгоритми и методи, тие популарно се

нарекуваат Големи Податоци (Big Data, анг). Во контекст на Големи Податоци, имплементацијата и извршувањето на алгоритмите за предобработка на податоци, одбирање на атрибути и градење предиктивни модели стануваат многу посложени.

Докторската дисертација предлагајќи нова техника за трансформација на атрибути ги збогати достапните методи за предобработка на номинални податоци, како и податоци кои претставуваат временски серии. Предложените хибридни метод за одбирање на атрибути овозможуваат намалување на времето потребно за одбирање на атрибутите, но уште поважно намалување на времето за тренирање на алгоритмите за машинско учење. Исто така, со употреба на генерални архитектури за паралелна и дистрибуирана обработка успешно беа паралелизирани неколку алгоритми за одбирање на атрибути. Пристапот за паралелизација не користи специјализиран хардвер, не е наменет за конкретен домен, а овозможува скалирање кое може да го следи растот на податоците.

Заради генералноста на пристапите кои се опфатени во ова истражување, методите и резултатите се применливи во многу деловни процеси и научноистражувачки проекти. Имено, во многу бизнис процеси често се среќаваат податоци од разновидни типови, кои со предложените пристапи ќе може соодветно да бидат обработени, така што максимално ќе се искористи нивниот предиктивен потенцијал. Потоа, со примена на предложените алгоритми за одбирање на атрибутите е овозможено забрзување на градењето на предиктивните модели. Паралелната имплементација на алгоритмите помага во совладувањето на предизвикот на Големи Податоци и овозможува скалабилност. Со ова се постигнува следење на трендот на зголемување на податоците без да треба да се менува архитектурата на системите или повторно да се имплементираат нивните функционалности. Ова претставува придобивка за компаниите со што се овозможува анализа на големи податочни множества со помош на софистицирани алгоритми за машинско учење во реално време. Ваквата способност обезбедува конкурентска предност, подобрување на квалитетот на услугите кои ги нудат, зголемување на бројот на корисници и нивната лојалност итн. Од аспект на корисниците, ова води кон зголемување на квалитетот на услугите што ги користат и добивање на нови услуги кои можеби и не знаеле дека им се потребни или дека се возможни.

Клучни зборови: предобработка на податоци, генерирање на атрибути, филтрирање на атрибути, алгоритми за класификација, паралелизција, дистрибуирано пресметување, компјутерски кластери

Ментор: Проф. д-р Андреа Кулаков, редовен професор

Parallelization of algorithms for data processing and classification

PhD Thesis

Eftim Zdravevski

Abstract

Companies and organizations routinely collect data of various types. For many of them, the success directly depends on their ability to analyze the collected data. Machine learning offers powerful tools for automatic analysis of large amounts of data. In order to apply machine learning algorithms, business and data needs to be understood, data needs to be preprocessed, feature engineering needs to be performed and relevant features to be selected. These phases take up a lot of time because they hugely depend on the nature of the data and the data collection process. One of the main tasks in preprocessing is generation of features that describe various aspects of the business problem and they should facilitate building of high-quality machine learning models. Feature generation needs to take into account the data types (i.e. numeric, nominal or text) and the way it was generated (i.e. event-driven generation or constant rate collection). Feature selection identifies the most relevant features and enables machine learning algorithms to focus on the aspects of data that are most useful for analysis and prediction, which in turn improves the predictive performance and lowers the computational and memory complexity of algorithms.

In recent years, companies collect high-volume data with high velocity. IBM estimates that over 90% of the world data is generated in the last couple of years. The data volume growth is attributed to: ubiquitous mobile devices and sensors; software logs; user interaction with web sites, applications and multimedia; social networks, etc. Depending on the context, data utilization faces at least some of the following challenges: collection, analysis, processing, searching, visualisation, transfer, security, privacy, report generation, decision making, providing recommendations and decision support, etc. When the data is so big that these challenges cannot be easily overcame with the traditional algorithms and methods, it is popularly called Big Data. In this context, implementation and execution of algorithms for data preprocessing, feature engineering and building prediction models become even more complex.

This PhD thesis proposes novel techniques for feature transformation of nominal and time series data. It also proposes and analyzes hybrid methods for feature selection which significantly reduce the feature selection time, but more importantly lower the time and memory requirements needed for building prediction models. By utilizing generic architectures for parallel and distributed processing, it proposes implementation of several feature selection algorithms that offer high scalability. The parallelization methods do not require any specialized hardware, are suitable for

variety of domains, while providing scalability that can cope with the data growth.

Owing to the generic approaches that are proposed in this study, the methods and results can be utilized in many business applications and scientific projects. Namely, many businesses collect data in variety of data types, which can be appropriately processed with the proposed methods, so their predictive potential is used optimally. Then, by applying the proposed methods for feature engineering, which include feature extraction and selection, the predictive models could be created in shorter time, while being more robust. The parallel implementation of algorithms mitigates many of the Big Data challenges and provides scalability. As a result, the data growth trends could be easily followed, without the need of changes in the architecture of applications or reimplementation of their functionalities. This can be a huge benefit for companies because Big Data can be analyzed with sophisticated algorithms in real time. Consequently, it can bring companies competitive advantages by improving their services, while increasing the number of users and their loyalty. From user's perspective, the quality of services that they consume will be improved. Moreover, novel groundbreaking services that were not even possible earlier could be offered to users.

Keywords: data preprocessing, feature engineering, feature extraction, feature selection, classification aglrithms, parallelization, distributed computing, computer cluster

Supervisor: Prof. Andrea Kulakov, PhD

Благодарност

Најпрво, на менторот Андреа Кулаков му благодарам за довербата и креативната слобода во текот на мојата досегашна кариера и што ме охрабруваше да најдам проблеми што сакам да ги решам размислувајќи со своја глава.

Огромна благодарност му должам и на професорот Владимир Трајковиќ за неговите прагматични совети, континуирано водство и отварање на хоризонтите. Тој никогаш не ги штедеше своето време и енергија за да понуди искрена помош и да ми даде насока во моментите на лутање.

На Петре Ламески му се заблагодарувам за бескрајните дискусии на научни и ненаучни теми од петто одделение наваму и што континуирано ги предизвикува моите идеи, поради што мора да го давам најдоброто од себе.

За секојдневна поддршка уште од кога памтам за себе, сестра ми Александра заслужува посебна благодарност.

Мајка ми Елица и татко ми Славчо целиот живот несебично го посветија на градење трајни вредности и овозможување дом исполнет со хармонија, љубов и разбирање. Тие трпеливо ми стваараа работни навики, ме поттикнуваа на креативно размислување и се заслужни за тоа што сум, поради што вечно ќе им бидам благодарен.

На крај, бескрајно сум благодарен на сопругата Катерина за нејзината љубов, безрезервна поддршка, жртвување и несебична помош без кои немаше да биде можна изработката на оваа докторска дистертација и воопшто мојот напредок во кариерата. Се разбира, не можам а да не ги спомнам и синовите Славе и Диме, кои се мојот неисцрпен извор на мотивација и позитивна енергија.

Содржина

Листа на слики

Листа на табели

Глава 1

Вовед

Денес живееме во информатичко време кога генерирањето и собирањето на податоци е лесно, а нивното чување е ефтино. Континуираниот раст на количеството на податоци носи нови предизвици за индустријата. Како што волуменот на податоците се зголемува, нивната анализа и разбирање се отежнуваат. Од друга страна, податоците носат неверојатни можности за развој на науката и индустријата и тие се главен двигател на многу нови услуги и производи во денешно време. Со помош на машинско учење може автоматски да се анализираат огромни количини на податоци и потоа да се изведуваат различни заклучоци. За податоците да бидат употребливи за алгоритмите за машинско учење, тие треба да се предобработат и доведат во соодветен облик и оваа фаза одзема најголем дел од целокупната интеграција и обработка на податоците. Токму поради, во фокусот на научните и деловните кругови е развојот на нови и унапредување на постоечките техники за ефикасно собирање, предобработка, чување, заштита, откривање на шаблони и визуализација на податоците.

1.1 Мотивација

Распространетоста на паметните уреди, сензори, социјални мрежи и разни апликации резултира во огромни количества на податоци [33, 34, 35, 36, 37, 38]. Од друга страна, луѓето се привикнуваат на добивање персонализирани и корисни услуги што се достапни моментално. Главен фокус на многу компании и истражувачи е обезбедување на пресонализирани информации во медицински системи, амбиентално потпомогнато живеење, пребарување, разни системи за препораки, разни услуги и слично. Во минатото компаниите можеа да одбираат кои податоци да ги чуваат, но денес нивниот успех зависи во голема мера од способноста да ги анализираат достапните податоци и да извлечат корисно знаење за потребите и очекувањата на нивните клиенти. Поради волуменот, брзината

на генерирање и разновидноста на податоците кои се генерираат, континуирано се развиваат нови методологии и архитектури кои овозможуваат справување со новите предизвици.

Со помош на алгоритмите за машинско учење се овозможува автоматска анализа на големи количества на податоци и нивните перформанси генерално се зголемуваат со искуствтото. Во целиот процес клучно е искуството кое се претставува со помош на податочни множества. Ретко кога податочните множества се во формат и облик кој е соодветен за директна примена на алгоритми за машинско учење. Напротив, најчесто податоците потекнуваат од повеќе извори и со оглед на нивната разновидност, интеграцијата и предобработката на податоците е многу важна. Пред да може да се употреби алгоритам за машинско учење односно да се моделираат податоците, најчесто претходат фазите на разбирање на бизнисот и податоците, предобработка и одбирање на атрибути, како што е впрочем препорачано во [39]. Овие фази одземаат многу време, бидејќи во голема мера зависат од природата на податоците кои се обработуваат и процесот при кој биле собирани.

Од податочните множества е потребно да се генерираат атрибути со помош инженерство на атрибути (feature engineering, анг). Инженерството на атрибути е клучно за примената на машинско учење и тоа претставува тежок, долготраен и скап процес кој бара експертско знаење. Кога инженерството на атрибути се прави рачно, тогаш тоа се состои од следниве чекори: разгледување и тестирање на атрибути, одлучување кои атрибути да се генерираат, генерирање на атрибути, евалуација на перформансите на моделот изграден со генерираните атрибути, подобрување на атрибутите и повторување на целата постапка сè додека не се добијат задоволителни перформанси. За да се поедностави овој процес на рачно генерирање атрибути, често се применуваат методи за автоматско учење и генерирање на атрибути. Тие вообичаено зависат од доменот на проблемот и типовите на податоци. Пристапите опишани во литературата [40, 41, 42, 43, 44, 45, 46, 47, 48, 49] за инженерство на атрибути од временски серии се ограничени на доменот и целта на примена. Согледувајќи го овој недостаток, во оваа докторска дисертација предлагаме методологија за инженерство на атрибути од произволни временски серии. Во деловните системи друг чест тип на податоци се номиналните и категоријалните атрибути. Кога бројот на номинални атрибути е голем и нивната кардиналност е висока, тогаш традиционалните методи за нивна трансформација во нумерички се соочуваат со повеќе недостатоци. За надминување на ваквите предизвици е предложена трансформација од номинални во нумерички атрибути со помош на параметарот тежина на фактите и методот е генерализиран на повеќекласни проблеми.

По предобработката следува одбирањето на атрибути. Со одбирањето на атрибути се идентификуваат најрелевантните атрибути и овозможува алгоритмот за машинско учење да се посвети на тие аспекти од податоците кои се најкорисни за анализа и идно предвидување. Оваа фаза е од исклучителна важност особено

кога податочните множества имаат огромен број на атрибути. Многу проблеми во биоинформатиката и проблеми поврзани со обработка на текстуални документи содржат илјадници атрибути. Особено каракактеристични се проблемите со обработка на текст каде може да се случи податочното множество да има десетици па дури и стотици илјади атрибути. Ваквиот огромен број на атрибути наметнува сериозни предизвици за алгоритмите за машинско учење. За нивно надминување, од една страна може да се употребат филтрирачките методи за одбирање на атрибути кои се карактеризираат со едноставност и брзина. Од друга страна се обвиткувачките методи, кои се помоќни, но мемориски и временски покомплексни. Поради тоа предлагаме неколку нови хибридни методи за надминување на недостатоците на овие методи. Во контекст на Големи Податоци кои потекнуваат од брзината на генрирање, волуменот на податоците и разновидноста на изворите и типовите на податоци, традиционалните методи за одбирање на атрибути се неупотребливи. Тие генерално работат секвенцијално и само некои од нив се паралелизирани. Поради тоа во оваа докторска дисертација е предложена паралелна имплементација на неколку рангирачки и хибридни методи за одбирање на атрибутите, со што се овозможува нивна примена и кај Големи Податоци. Поради истите причини потребна е паралелизација на алгоритмите за машинско учење со цел да се овозможи обработка на Големи Податоци и градење на ефикасни предиктивни модели, дури и покрај предизвиците кои потекнуваат од волуменот и разноврсноста на податоците.

1.2 Хипотези на истражувањето

Во оваа докторска дисертација главна хипотеза е дека:

Примената на паралелно и дистрибирано пресметување кај големите податочни множества ќе овозможи извлекување на знаење од нив користејќи машинско учење.

Евалуацијата на оваа хипотеза во основа зависи од неколку подхипотези. Најпрво, потребни се методи за инженерство на атрибути кои се генерички и применливи за различни типови на одатоци, а од друга страна се лесно паралелизибилни. Поради тоа, *првата подхипотеза* гласи: *со автоматско инженерство на атрибути од произволни временски серии и номинални атрибути се овозможува генерирање на информативни атрибути.* Во Глава 3 се опишани методите кои подоцна се искористени за потврдување на оваа подхипотеза преку неколку студии на случај опишани во Глава 6.

Генерирањето на многу атрибути, од друга страна, бара нивно одбирање со цел да се добие концизно множество на репрезентативни атрибути кое овозможува алгоритмите за машинско учење да се забрзаат и да имаат помала мемориска комплексност. Поради тоа се издвојува *втората подхипотеза: рангирачките и*

обвиткувачките методи за одбирање на атрибути може да се комбинираат во хибридни методи за ефикасно добивање мали множества од информативни атрибути. Предложените хибридните методи се опишани во Глава 4 и тие преку неколку студии на случај опишани во Глава 6 ја потврдуваат оваа хипотеза.

Во евалуацијата на главната хипотеза клучна е *третата подхипотеза: користењето на генерички платформи и техники за дистрибуирано пресметување на компјутерски кластери овозможува паралелизација на алгоритмите за предобработка и класификација на податоците*. Методите на кои се темели оваа подхипотеза се опишани во Глава 5. За потврдување на оваа подхипотеза најпрво е потребно овозможување на рамномерна распределба на оптоварувањето на јазлите на кластерот при запишување и читање. Поради тоа во Глава 5.2 е предложен дизајн на примарните клучеви на табелите кај NoSQL базите на податоци со кои се овозможува рамномерна распределба на оптоварувањето на јазлите на кластерот при запишување и читање. Потоа, за инженерство на атрибути од временски серии се користат методите опишани во Глава 5.3. Исто така, во Глава 5.4 е предложен начин за генерирање на атрибути преку агрегација на Големи Податоци. Паралелизацијата на пресметката на информационата добивка е опишана во Глава 5.5. Паралелизацијата на хибридните и обвиткувачките методи за одбирање на атрибути е предложена во Глава 5.6. Оваа подхипотеза е евалуирана со помош на повеќе студии на случај опишани во Главите 6.9.1, 6.8 и 6.9.2, и според добиените резултати подхипотезата се потврдува.

1.3 Преглед на докторската дисертација

Во Глава 2 е даден вовед во концептите на надгледувано машинско учење и класификација, се опишани стратегиите за евалуација на предиктивните модели и е даден краток осврт на некои позначајни алгоритми за машинско учење.

Следно, во Глава 3 е направен преглед на литературата поврзана со техниките за инженерство на атрибути од временски серии и трансформација на номиналните атрибути во нумерички. Потоа е опишана *предложената генеричка методологија за инженерство на атрибути од временски серии* кои може да потекнуваат од произволни извори. Методологијата користи разновидни техники за генерирање нови временски серии и генерирање на атрибути од оригиналните и изведените временски серии: пресметување на разни статистички параметри за вредностите во временските серии, пресметување на хистограми со еднакво растојание и перцентили, трансформирање од временски во фреквентен домен со помош на брза Фуриева трансформација, пресметка на корелации и автокорелации и друго.

Исто така во Глава 3 детално е објаснета *предложената техника за трансформација на номинални и категоријални атрибути во нумерички* која се те-

мели на параметарот тежина на фактите. Предложеното подобрување на овој параметар овозможува негова употреба дури и кај повеќекласни проблеми за трансформација на номинални во нумерички атрибути. Дополнително, предложен е начин за негово пресметување дури и во гранични случаи кога предулсовите не се задоволени.

Понатаму, во Глава 4 се опишани различните типови на методи за одбирање на атрибутите кои постојат во литературата: филтрирачки, обвиткувачки и хибридни. Воочувајќи ги предностите и недостатоците на некои филтрирачки и обвиткувачки методи за одбирање, *предложени се неколку хибридни методи за одбирање на атрибути* кои ги надминуваат нивните недостатоци. Прво е предложен хибриден метод кој прави комбинација од филтрирачки и корелациски методи со цел брзо да се намали множеството на атрибути и потоа полесно да се определат меѓузависните атрибути. Потоа е предложен хибриден метод за брзо откривање на атрибути кои се осетливи на концептуални отстапувања. Веројатносната дистрибуција на ваквите атрибути е променлива во текот на времето, поради што изградените модели може да станат неупотребливи, бидејќи биле претренирани и соодветни само за тренинг множеството. Во оваа глава е предложен уште еден хибриден метод за брзо пребарување на просторот на множества на атрибути кој прво ги рангира атрибутите и потоа со помош на алчно диверзифицирано пребарување напред-назад наоѓа мали подмножества на информативни и независни атрибути. На крајот на Глава 4 е предложен уште еден хибриден метод за одбирање на атрибутите кој ги комбинира претходните два методи во еден.

Во Глава 5 е опфатена *паралелизацијата на алгоритмите за предобработка и класификација на податоците*. Прво се опишани најпопуларните платформи и концепти за дистрибуирано и паралелно пресметување: Hadoop, MapReduce Spark, HDFS и HBase. Потоа е направен преглед на литературата поврзана со паралелизација на секвенцијални алгоритми со помош на овие технологии. Во продолжение се дискутира за предложените дизајни на примерни клучеви на HBase табели со кои се овозможува рамномерно оптоварување на јазлите во кластерот за време на операции поврзани со пишување (пр. при запишување на податоци во базата, трансформации на податоците, генерирање на атрибути, итн), а истовремено овозможувајќи брз случаен пристап. Понатаму, се разгледани неколку методи за автоматско генерирање на атрибути со помош на агрегациски функции и е предложена нивна паралелна имплементација со помош на Spark. Следно е предложен начин за паралелизација на пресметката на информационата добивка со помош на Pig Latin, MapReduce или Spark. Понатаму во оваа глава се предлага метод за паралелизација на обвиткувачките методи за одбирање на атрибутите преку паралелна евалуација на различни множества на атрибути. Глава 6 завршува со кратка дискусија за учењето на класификациони модели во дистрибуирана околина, како и со предложен метод за дистрибуирање на едноставните класификациски модели на секој јазол со цел да се овозможи паралелна класификација на новите податоци.

Во Глава 6 се опишани повеќе студии на случај во кои успешно се употребени предложените методи за инженерство и одбирање на атрибути. Применета е информационата добивка за дискретизација и пресметка на оптимални прагови за поделба на интервалите на ризик во медицината. Со методологијата за инженерство на атрибути од временски серии е овозможено: препознавање на активности на пожарникари; предвидување на опасни концентрации на метан во рудници за јаглен; предвидување на сеизмички активности во рудници за јаглен; препознавање на активности во системи за амбиентално потпомогнато учење; и е овозможено потиснување на лажните аларми на мониторите во одделите за интензивна нега. Исто така е демонстрирана скалабилноста на паралелизираните алгоритми со помош на трансформација, агрегација и одбирање на атрибути кај Големи Податоци.

На крај, во Глава 7 се резимираат придонесот и резултатите од истражувањата опишани во оваа докторска дисертација. Наведени се насоки за идни истражувања и се посочени предизвици кои допрва треба да се надминат.

1.4 Постигнати резултати и објавени трудови

Во текот на изработката на оваа докторска дисертација беа објавени 3 научни трудови во меѓународни списанија со импакт фактор: [1] и [3] се објавени во списанието IEEE Access чиј импакт фактор е 3.2, [2] е објавен во списанието PLOS One чиј импакт фактор е 4.4, додека [4] е објавен во European Journal of Obstetrics & Gynecology and Reproductive Biology чиј импакт фактор е 1.662. Во зборници на меѓународни конференции беа објавени следниве 18 научни трудови кои директно се поврзани со темата на докторската дисертација: [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. Исто така, уште 10 научни трудови беа објавени во зборници на меѓународни научни конференции, кои иако користат алгоритми за предобработка и класификација, сепак не даваат директен придонес во потесната област од интерес на оваа дисертација: [23, 24, 25, 26, 27, 28, 29, 30, 31, 32].

При развојот на методите предложени во оваа докторска дисертација истите беа искористени за учество на повеќе меѓународни натпревари за податочно рударење, инженерство на атрибути и одбирање на атрибути, при што на четири од нив [50, 51, 52, 53] методите беа препознаени како иновативни и со висок потенцијал за практична употреба помеѓу стотици други предложени методи, поради што беа презентирани на соодвените конференции. На еден од натпреварите [51], благодарение на методот за инженерство на атрибути од временски серии, беше освоено трето место во конкуренција на стотина тимови од целиот свет.

Глава 2

Машинско учење

Надгледуваното машинско учење има за цел да научи одредена функција од означени тренинг податоци (инстанци). Кај надгледуваното машинско учење секоја тренинг инстанца е опишана со атрибути и за неа е позната ознаката односно излезната променлива. Со учењето на тренинг множеството се креира модел кој е способен да препознава нови инстанци и да ја предвидува нивната излезна променлива. Кога излезната променлива е елемент од некое конечно множество, тогаш станува збор за класификациски проблем. Градењето на класификациски модели врз основа на реални податоци се соочува со повеќе проблеми и некои од нив може да се надминат со инженерство и одбирање на атрибути.

Во оваа глава е даден краток осврт на методите за евалуација на предиктивните модели и се опишани некои позначајни алгоритми за надгледувано машинско учење.

2.1 Методи за евалуација на класификациските модели

Проценката на прецизноста на класификациските модели добиени со надгледувано учење е важна не само заради добивање претстава за идната прецизност на моделите, туку и заради изборот на моделот. Евалуацијата на ефикасноста на методите за предобработка на атрибутите најчесто се прави преку евалуација на класификациските модели кои се изградени после предобработката. Начинот на евалуацијата на класификациските модели може да влијае на толкувањето на придобивките од инженерството и одбирањето на атрибутите.

Идеалните модели би имале мала наклонетост кон тренинг податоците и мала варијанса, но во пракса никој начин на проценка на перфромансите не

може да биде точен цело време [54].

2.1.1 Тренинг и тест множества

Сите алгоритми за надлгедувано учење користат множество на податоци за градење на класификациски модели. Множеството што се користи за учење на моделот се нарекува тренинг множество. Со методот на задржување (houldout, анг.) обично се одделува посебно тест множество со кое се проценуваат перформансите на изградениот модел со помош на тренинг множеството. Ако се претпостави дека со повеќе тренинг примероци, алгоритмот подобро учи и гради подобри модели, тогаш овој метод за евалуација со тест множество се смета за песимистичен бидејќи само дел од сите достапни примероци се достапни за учење. Со зголемување на тренинг множеството за сметка на тест множеството се намалува наклонетоста на моделот (bias, анг.), но се зголемува варијансата, при што важи и обратното.

Проширување на овој метод на евалуација е со случајно одбирање (random subsampling, анг.) при што целата постапка се повторува и резултатите се усреднуваат. На овој начин може да се наруши претпоставката за независност на тренинг и тест примероците.

Главниот недостаток на овој метод е тоа што неефикасно се користат достапните податоци за градење на моделот бидејќи тест множество не се користи за учење на моделот. Кога податочните множества се големи, овој проблем и песимистичноста на пристапот се надминуваат.

2.1.2 Валидациски множества

Освен тренинг и тест подмножеството, може дополнително да се издвои и валидациско подмножество. Целта на ова подмножество е да помогне во учењето на хипер параметрите на некои класификациски алгоритми. Имено, повеќе класификациски алгоритми имаат дополнителни параметри кои може да влијаат на нивните перформанси. Во продолжени се дадени примери за алгоригмите опишани во Глава 2.2. Кај машините со носечки вектори (Support Vector Machines, SVM, анг.) тоа се параметарите C (цена на грешка) и γ (закривеност на хиперлиниите, во случај на Гаусов кернел). Кај наивниот Баесов класификатор тоа е функцијата за веројатност. Кај случајните шуми и екстремно случјаните дрва тоа е бројот на атрибути кои се достапни во секое од индивидуалните дрва за одлучување. Кај k-најблиските соседи тоа е бројот на соседи. Ако не се подесат овие хипер-параметри, тогаш може да не се добијат најоптималните перформанси и понекогаш разликата помеѓу модел со неоптимизрани и оптимизрани параметри може да биде огромна.

Валидациското множество служи токму за таа намена - оптимизрање на хипер-параметрите. Имено, со тренинг множеството се градат повеќе класификациски модели со различни вредности на хипер-параметрите. Потоа нивните перформанси се евалуираат со помош на валидациското множество. Вредностите на хипер-параметрите кои резултирале со најдобри класификациски модели се одбираат за најдобри и потоа се користат за градење на финалните класификациски модели. На крај од секој алгоритам постои по еден модел кој се евалуира на независното тест множество. Во случај кога алгоритмот нема хипер-параметри (пр. логистичка регресија), тогаш нема потреба од валидациско множество.

Валидациското множество е важно и за обвиткувачките методи за одбирање на атрибути. Имено, откако ќе се изгради класификациски модел тој се евалуира со помош на валидациското подмножество. Различни множества на атрибути се користат за учење на различни модели кои потоа се евалуираат со помош на валидациското множество. Множеството на атрибути што дава најдобри перформасни се смета за најдобро.

Главен недостаток на употребата на валидациски множества е дополнителното намалување на примероците кои се користат за учење на финалните модели, но во случај на големи податочни множества овој недостаток станува ирелевантен. Друг начин како може да се надмине овој недостаток е со привремено разделување на тренинг множеството на две подмножества, едно за учење и едно за валидација. Откако ќе се определат хипер-параметрите, тогаш повторно целото тренинг множество се користи за учење на финалниот модел. Недостаток на оваа постапка е тоа што ја зголемува наклонетоста.

2.1.3 Вкрстена валидација и изоставување на еден примерок

Вкрстена валидација претставува метод кој го дели целокупното податочно множество на k дисјунктни подмножества со приближно иста големина. Потоа моделот се тренира користејќи $k-1$ подмножества, а се евалуира со проестанатото подмножество. При вкрстена валидација со k групи (k-fold cross validation, анг.) целата постапка се повторува k пати и перформансите се агрегираат.

Дополнително може ова да прошири на тој начин што се води грижа за дистрибуцијата на инстанците во однос на целната класа да биде еднаква во секоја група. Овој метод се нарекува слоевита (stratified, анг.) вкрстена валидација.

Кога k е еднаков на бројот на инстанци во податочното множество тогаш се добива еден специјален случај на вкрстена валидација кој се нарекува изоставување на еден примерок (leave-one-out, анг.). Оваа постапка е соодветна кога вкупниот број на инстанци е мал, па не постои можност да се употреби методот со тренинг и тест множество.

2.1.4 Специјализирани поделби на множествата

Во некои случаи методите за евалуација треба да ја земат во предвид вредноста на некој атрибут при поделбата на множеството на подмножества. Често податоците може значително да се разликуваат за некои субјекти од други, поради што стандардните евалуации со тренинг-тест множества или вкрстена валидација да даваат премногу оптимистички проценки.

На пример, во медицинските системи може да има многу инстанци за секој пациент во рамките на множеството, но инстанците да се разликуваат значително во зависност од тоа за кој пациент се однесуваат. Ако се истренира модел со некоја случајна поделба на множеството (на тренинг и тест или на групи кај врстената валидација) тогаш многу веројатно е дека инстанци од секој пациент ќе се наоѓаат во секоја група. Поради тоа во евалуацијата на изградените модели ќе учествуваат инстанци за истите пациенти. Но, понекогаш е потребно да се направи модел кој ќе има добри генерализирачки способности, дури и за нови пациенти. Евалуацијата на претходниот начин може да биде премногу оптимистичка бидејќи во тест множеството многу од инстанците ќе бидат на веќе постоечки пациенти.

За да се надминат ваквите проблеми, односно да се добијат пореални процени на генерализирачките способности на алгоритмите, при поделбите на подмножества може да се вклучи информација за субјектот на кого се однесува инстанцата. Така, при поделбата на множеството на подмножества, инстанците од едни субјекти се доделуваат на тренинг подмножеството, од други субјекти во валидациско подмножество и од трети субјекти во тест подмножество.

2.2 Краток осврт на некои алгоритми за надгледувано машинско учење

Во оваа глава накратко се опишани неколку алгоритми за надгледувано машинско учење кои се употребуваат во оваа докторска дисертација. Изборот на опишаните алгоритми е направен врз основа на нивната популарност во научните и деловните кругови која се должи барем на некои од следниве карактеристики: предиктивна моќ, едноставност за употреба, едноставност за паралелизација и интерпретабилност на моделите.

2.2.1 Машини со носечки вектори

Алгоритмот машини со носечки вектори (Support Vector Machine, SVM, анг.) [55] е алгоритам за надгледувано машинско учење. Според оригиналната дефиниција тренинг и тест инстанците припаѓаат на една од две класи. Инстанците се наоѓаат во хипер-простор, дефиниран според нивните атрибути, и алгоритмот прави модел кој треба да научи да ги одделува инстанците од двете класи со што е можно поголема маргина. За повеќекласни проблеми овој алгоритам се проширува со пристапот еден-против-сите или сите-против-сите [56, 57].

Просторот на атрибути SVM го дели барајќи носечки вектори со најголеми растојанија помеѓу најблиските точки од двете класи. Нека тренинг множеството од k инстанци и n атрибути е претставено преку паровите (x_i, y_i), $i = 1, ..., k$, каде $x_i \in R^n$, така што $y_i \in \{-1, 1\}$. Треба да се најде решение на проблемот кој е изразен со (2.1) и (2.2) со цел да се генерира SVM класификациски модел врз основа на тренинг податоците.

$$\min_{w,b,\xi} \frac{1}{2} W^T W + C \sum_{0}^{k} (\xi i) \tag{2.1}$$

каде важи:

$$y_i(W^T \phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0 \tag{2.2}$$

Просторот се разделува со хипер-линии така што сите инстанци кои се на една страна од линиите припаѓаат на една класа, а сите инстанци од другата страна припаѓаат на другата класа. Ако не е можно да се најдат хипер-линиите што идеално го делат просторот врз основа на тренинг податоците, алгоритмот се обидува да го раздели просторот на тој начин така што класификациската грешка се минимизира. Бидејќи во општ случај класификациските проблеми не се линеарно сепарабилни, не е возможно да се најде хипер-линија што го дели просторот на начин на кој атрибутите од двете класи ќе може да се разликуваат со разумна класификациска грешка. Поради тоа во истражувањата и студиите на случај презентирани во оваа докторска дистертација користат SVM со Гаусов кернел, кој користи параметри C (цена) и γ (закривеност на хипер-линиите) со кои се трансформира векторскиот простор на атрибутите во повисока димензија каде делењето на просторот може да се направи со поголема прецизност. Трансофрмацијата се прави со кернел функција $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, дефинирана за Гаусов SVM како $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0$.

Иако SVM е многу моќен класификатор, кај големи податочни множества тренирањето трае подолго. Овој проблем ескалира кога треба да се направи оптимизација на параметрите на алгоритмот, што се препорачува бидејќи тоа ги подобрува предиктивните перформанси и го намалува претренирањето [12]. Кога се користат машини со носечки вектори, и кога податочното множество има раз-

новидни атрибути, пожелно е множествата да се нормализираат така што секој атрибут да има аритметичка средина 0 и стандардна девијација 1, со што може дополнително да се подобрат перформансите [15].

Оптимизацијата на параметрите C и γ може значително да ги подобри перформансите на класификациониот модел [12, 15]. Параметарот C ја дефинира казната (цената) за грешки при класификацијата. Зголемувањето на C ја зголемува казната и го намалува бројот на инстанци што може да се наоѓа во маргината на грешка. Помали вредности на C овозможува правење на поголема маргина на грешка кога се разделува хипер просторот. Параметарот γ е специфичен за Гаусовиот SVM и влијае на флексибилноста на хипер-линиите. За помали вредности на гама, линија што го разделува хипер-просторот е речиси линеарна, а за поголеми вредности станува позакривена. Преголемото зголемување на γ може да предизвика претренирање и заситување на моделот да одговара на тренинг податоците [58]. Во [12] е предложен алгоритам во две фази за пребарување на просторот на параметри. Прво, со експоненцијално растечки чекори се тестираат различни комбинации на вредности за C и γ и се наоѓаат добри вредности. Потоа, со пробување на помали промени на вредностите се прави фино подесување. Интервалот на вредности во кои се прави пребарувањето за двата параметри е $(10^{-6}, 10^{6})$, што значи дека иницијално се прават $12 \times 12 = 144$ комбинации. Пребарувањето може да се упрости ако се забележи дека со менување на некоја вредност значително се деградираат перформансите споредено со најдобрите тековни перформанси.

2.2.2 Дрва за одлучување

Во методите разгледувани во оваа докторска дистертација како и во студиите на случај не се користат дрва за одлучување експлицитно, но бидејќи тие се фундаментален алгоритам кој се користи во ансамблите од класификатори, тие се објаснети накратко. Дрвата за одлучување [59] се алгоритми за гредење предиктивни модели кои во случајот на класификација можат да мапираат инстанци опишани со множество на атрибути во конечно множество на класи. Дрвата за одлучување се тренираат (учат) со помош на делење на тренинг множеството на подмножества врз основа на целната класа. Процесот се повторува рекурзивно за секое подмножество сè додека или сите терминални јазли (листови во дрвото) содржат инстанци само од една класа или кога понатамошното делење не дава дополнителна вредност во предвидувањата. Овој процес од-горе-надолу на градење на дрвата за одлучување е убедливо наупотребуваниот. Различни алгоритми на градење дрва за одлучување користат различни метрики за евалуација на чистотата на примероците кои припаѓаат на одреден јазол (пр. ентропија, Џини индекс на нечистотија, итн.). Откако ќе се изгради дрво за одлучување, непознатите инстанци се класифицираат на тој начин што во секој јазол се прави проверка во која од следните гранки припаѓа инстанцата. Кога ќе се стигне до

лист на дрвото (терминален јазол) инстанцата се класифицира во класата што е претставена со листот.

2.2.3 Случајни шуми

Ансамблите од класификатори е концепт на комбинирање на индивидуални класификатори со цел да се подобрат нивните перформанси. Постојат многу начини за комбинирање на индивидуалните класификатори, од кои еден е предложен во [22]. Алгоритмот случајни шуми (Random Forest, RF, анг.) [60] е еден од најпознатите претставници на ансамблите од класификатори. Со него се градат повеќе дрва за одлучување кои гласаат за да се донесе конечно предвидување. Индивидуалните класификатори во ансамбалот имаат случајно подмножество со повторување од сите атрибути со предефинирана големина и дрвата не се поткаструваат. Притоа се користи усреднување во торба (bootstrap aggregating и bagging, анг.) со кој се подобрува стабилноста и прецизноста на алгоритмите, особено кога се користат случајни атрибути. Во текот на учењето, дел од тренинг примероците не се користат за учење на моделот, туку за евалуација и проценка на грешката. Кога се користат голем број на дрва за одлучување во ансамбалот на случајните шуми, тогаш генерализирачката грешка се намалува и предиктивниот модел на случаните шуми е стабилен и робустен. Некои од главните позитивни карактеристики на случајните шуми се:

- Робустни се на екстремни вредности (outliers, анг.) и шум.
- Тие се транспарентни, давајќи проценка на грешката, важноста и корелацијата на атрибутите.
- Лесно се паралелизираат.

Поради овие карактеристики, овој алгоритам се користи во експериментите и студиите на случај опишани во оваа докторска дисертација.

2.2.4 Екстремно случајни дрва

Екстремно случајни дрва (Extremely Randomized Trees, ERT, анг.) е друг алгоритам кој гради ансамбли од дрва за одлучување [61]. За разлика од случајните шуми, каде секое дрво за одлучување се гради според оптималните поделби (cut-point, анг.) на атрибутите земајќи ја во предвид целната класа, кај екстремно случајните дрва поделбите во секое дрво на одлучување се сосема случајни. Со зголемувањето на случајноста при градењето на ансамбалост целта е да се намали варијансата, со што би се подобриле генерализирачките способности. Друга разлика од случајните шуми е тоа што дрвата за одлучување во ансамбалот на екстремно случајните дрва се градат со помош на сите тренинг податоци. Бидејќи

поделбите на атрибутите во дрвата за одлучување се случајни а не оптимални, обично екстремно случајните дрва се пресметковно поефикасни од случајните шуми [10]. И екстремно случајните дрва се лесни за паралелизација и даваат информација за информативноста на атрибутите.

2.2.5 K-најблиски соседи

Алгоритмот K-најблиски соседи (k-Nearest Neighbors, kNN, анг.) [62] е метод за класификација и регресија. За класификација се користат k најблиски тренинг инстанци во просторот на атрибути за да се определи со мнозинско гласање на која класа припаѓа непознатата инстанца. Овој алгоритам е претставник на типовите на алгоритми кои учат мрзливо, односно функцијата се апроксимира локално и целокупната пресметка се одлага се додека не е потребна класификација, односно не се прави предиктивен модел претходно. И покрај едноставноста на овој алгоритам, сепак во многу случаи тој дава високи предиктивни перформанси. Во зависност од вредноста на k, при класификацијата се зема соодветниот број на соседи за мнозинското гласање, кое може да биде и тежинско, во зависност од оддалеченоста на непознатата инстанца од нејзините соседи. Главен недостаток на овој алгоритам е чувствителноста на локалната структура на податоците.

Во зависност од типот на податоците може се употребува различна метрика за растојание, но за нумерички типови на податоци најчесто се користи Евклидовото растојание. Изборот за k зависи од типот на податоци. Генерално, поголеми вредности на k го намалуваат ефектот на шум кај податоците, но прават границите помеѓу класите да бидат слични. Најчесто се прави оптимизација на k користејќи тернинг множество за градење на класификационен модел и евалуација со валидациско множество. Вредноста на k што дава најдобри валидациски перформанси се користи при предвудување на инстанците од тест множеството.

2.2.6 Наивен Баесов класификатор

Навните Баесови класификатори [63] се фамилија на едноставни веројатносни класификатори кои се темелат на примена на Баесовата теорема со наивни претпоставки дека атрибутите се независни едни од други. Овие класификатори се многу скалабилни и користат параметри чиј број линеарно зависи од бројот на атрибути во даден проблем. Тренирањето се прави со линеарно време, за разлика од итеративните пристапи кај многу други алгоритми кои се пресметковно поскапи. Во зависност од веројатносниот модел, наивните Баесови класификатори се тренираат различно, но еден од најпопуларните методи е тој на максимална веројатност (maximum likelihood, анг.). За номинални или дискретни променливи, со помош на броење на тренинг примероците во зависност од различните вредности на атрибутите и на целната класа се добиваат условните веројатности. Од друга

страна, кај континуални променливи еден популарен начин е да се претпостави веројатносната дистрибуција на атрибутите (пр. Гаусова нормална дистрибуција, Бернулиева дистрибуција, бимодална и други). Потоа за одбраната дистрибуција се учат параметрите. На пример, за Гаусовата нормална дистирбуција се наоѓа средната вредност и стандардната девијација на секој атрибут во рамките на тренинг множеството и потоа ова се користи во градењето на предиктивниот модел. И покрај едноставниот дизајн и упростените претпоставки, наивните Баесови клаисификатори успешно се употребуваат во индустријата. Тие се од интерес во оваа докторска дисертација поради лесната паралелизибилност, брзината и можноста да се утврди дали определен атрибут е корисен во моделот или не е, со негово додавање или бришење од множеството на атрибути.

2.2.7 Логистичка регресија

Логистичката регресија [64] е дел од категоријата на статистички модели кои се нарекуваат генерализирани линеарни модели. Од дадено множество на континуални, дискретни или номинални атрибути, со неа може да се предвидуваат дискретни исходи. Атрибутите во логистичката регресија може да бидат од било која форма и таа не прави претпоставки за дистрибуцијата на независните атрибути и тие не мора да се со нормална дистрибуција, да се линеарно поврзани или да имаат еднаква варијанса во рамките на секоја група. Логистичката регресија прави линеарен модел врз основа на трансформирана целна класа. Моделите што се прават со логистичка регресија се едноставни за толкување, и врз основа на нив може лесно да се направи проценка на важноста на атрибутите во моделот. Поради едноставноста на алгоритмот, логистичката регресија лесно се паралелизира. Друга корисна особина на овој алгоритам е можноста да се утврди дали определен атрибут е корисен во моделот или не со помош на негово додавање или бришење од множеството на атрибути.

Глава 3

Инженерство на атрибути

Со помош на алгоритмите за машинско учење се овозможува автоматска анализа на големи количества на податоци и нивните перформанси генерално се зголемуваат со искуствтото. Во целиот процес клучно е искуството кое се претставува со помош на податочни множества. Ретко кога податочните множества се во формат и облик кој е соодветен за директна примена на алгоритми за машинско учење. Напротив, најчесто податоците потекнуваат од повеќе извори. Пред да може да се употреби алгоритам за машинско учење односно да се моделираат податоците, најчесто претходат фазите на разбирање на бизнисот и податоците, предобработка и одбирање на атрибути, како што е впрочем препорачано во [39]. Овие фази одземаат многу време, бидејќи во голема мера зависат од природата на податоците кои се обработуваат и процесот при кој биле собирани. Во [65, 66] се опишани постапките за предобработка на податоците: отстранување на екстремните вредности (outliers, анг.), отстранување или моделирање на шумот, моделирање на вредностите кои недостасуваат, се прават трансформации на податоците со цел да се нормализира дистрибуцијата на вредностите на атрибутите или да се претстават во попогоден облик за обработка (најчесто во нумерички) итн. Во [67, 68, 69] се опишани пристапи кои ги анализираат постапките за предобработката и нивните ефекти врз понатамошното тренирање на различни алгоритми од повеќе домени на примена. Во [70] се објаснети повеќе примери од различни деловни домени кои многу често содржат мешани типови на податоци. Авторите препорачуваат кога не постои еднозначно природно мапирање на номинални во нумерички податоци, најдобро е истите да се обработуваат генерирајќи бинарни променливи за секоја номинална вредност. Слични препораки се дадени во [71, 72]. Кога станува збор за обработка на текстуални податоци и извлекување на знаење од нив, најчесто се употребува функцијата TF-IDF (term frequency-inverse document frequency), како што е опишано во [73] и [74].

Најчесто, од податочните множества потребно е да се генерираат атрибути

со помош инженерство на атрибути (feature engineering, анг.). Впрочем, инженерството на атрибути е клучно за примената на машинско учење и тоа претставува тежок, долготраен и скап процес кој бара експертско знаење. Кога инженерството на атрибути се прави рачно, тогаш тоа се состои од следниве чекори: разгледување и тестирање на атрибути, одлучување кои атрибути да се генерираат, генерирање на атрибути, евалуација на перформансите на моделот изграден со генерираните атрибути, подобрување на атрибутите и повторување на целата постапка сè додека не се добијат задоволителни перформанси. За да се поедностави овој процес на рачно генерирање атрибути, често се применуваат методи за автоматско учење и генерирање на атрибути. Тие вообичаено зависат од доменот на проблемот и типовите на податоци.

Во оваа глава е направен преглед на техниките за инженерство на атрибути од временски серии и трансформација на номиналните атрибути во нумерички кои се достапни во литературата. Потоа е опишана предложената генеричка методологија за инженерство на атрибути од произволни временски серии. Исто така, детално е објаснета предложената техника за трансформација на номинални и категоријални атрибути во нумерички.

3.1 Инженерство на атрибути кај временски серии

Во многу деловни процеси во текот на времето се генерираат податоци кои претставуваат временски серии. Некои од нив се генерираат со константна рата, а други во моментите на случување на некој настан. Карактеристични податоци од првиот тип се: финансиски и економски показатели, индекси на берза, курсни листи итн. Кај нив временското растојание помеѓу две вредности е константно. Вториот тип на податоци се генерираат во моментот на извршување на некој настан (пр. трансакција) и растојанието помеѓу настаните е променливо.

Благодарение на развојот на лесни и ефтини сензори, тие денес се распространети наоколу и служат за мерење на различни активности и појави. Поради тоа, во многу други области каде тие се застапени се генерираат податоци во вид на временски серии: метерологија, фабрики, рудници, медицина, екоинформатика, паметни градови, паметни околини, амбиентално помогнато живеење, разни паметни уреди, итн. Овие сензори генерираат мерења (пр. температура, забрзување, жироскопски мерења, ГПС локации и друго) кои често треба да се процесираат во реално време.

Анализата на временски серии најчесто се прави со цел на откривање на шаблони, кластерирање, класификација или откривање на правила [75]. Заради густината на податоците што се мерат од сензорите и природата на временските

серии, инженерство на атрибути и намалувањето на димензионалноста е процесот кој е неопходен за примена на алгоритми за машинско учење [76]. Вообичаено временските серии, особено кога доаѓаат од повеќе извори (пр. сензори) содржат шум и не се секогаш временски порамнети [77]. Друг предизвик при генеирањето на предиктивни атрибути е справување со остапувања на атрибутите и промени во нивната веројатносна дистрибуција, кои може да се резултат на отчитувања направени од различни сензори, податоци собрани за различни луѓе, деградација на осетливоста на сензорите со текот на времето итн. Овие карактеристики на реалните системи кои генерираат временски серии треба се земат во предвид при инженерството на атрибути, така што тие нема да влијаат значително на моделите за анализа на податоците.

Често, во зависност од доменот на проблемот типовите на атрибути кои се генерираат веќе се докажани како корисни во истиот или некој сличен домен. Ова често е субјективно и зависи од искуството на истражувачот. Со цел да се надмине ова, ние предлагаме метод кој прво систематски генерира нови временски серии изведени од оригиналните, со цел понатаму од нив да се генерираат разновидни информативни и робустни атрибути.

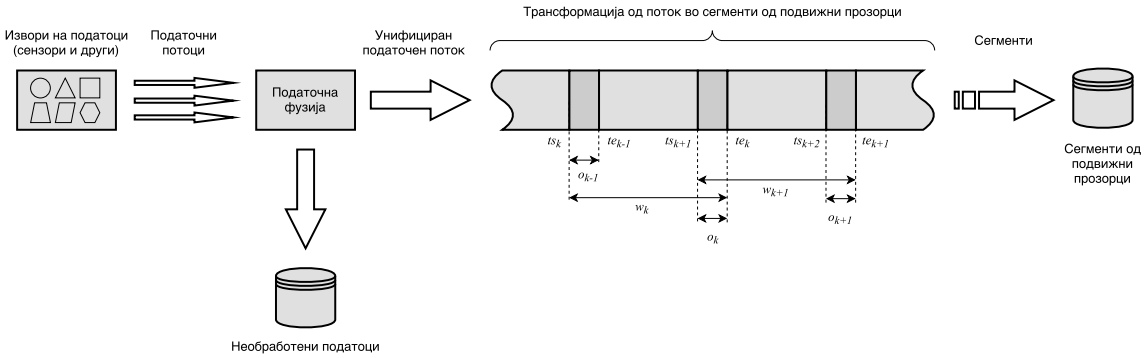
Во следните подглавја ќе бидат опишани постапките кои претходат на инженерството на атрибути од временски серии, како и предложената методологија за автоматско инженерство.

3.1.1 Фузија на податоци и сегментација

Пред да се започне со генерирање и одбирање на атрибути, податоците во временските серии треба да се сегментираат на соодветен начин. Во [78] е даден преглед на подготвителните чекори кои се потребни во системите за класификација на активности. Дополнително, во таа студија се дискутира влијанието на должината на подвижните прозорци во препознавањето на активности на луѓе.

Процесот на сегментирање на податочните потоци (data streams, анг.) со подвижни прозорци е прикажан на Слика 3-1. Фузијата на податоци се состои од парсирање, форматирање, временско порамнување и мапирање на податочните типови кај влезните потоци од податоци. Со фузијата сите податочни потоци се унифицираат во еден поток кој потоа се зачувува во база на податоци.

Иако во продолжение се дискутира за сензори и нивните мерења, истите методи може да се употребат за било кои извори на податоци. Различните сензори прават мерења со различна фреквенција. За секое мерење е познат временскиот печат (timestamp, анг.) t кога тоа било направено. Временскиот печат е неопходен за правилно временско порамнување на влезните податочни потоци и сегментирање на унифицираниот податочен поток во подвижни прозорци. Заради поедноставување, да претпоставиме дека сите мерења се чуваат во еден ентитет



Слика 3-1: Сегментација на податочни потоци со подвижни прозорци

R кој ги има следниве информации: временски печат t, измерена вредност v и идентификатор на сензорот s.

На Слика 3-1, k-тиот прозорец (w_k) се состои од сите податоци кои биле собрани во временскиот период $(ts_k, te_k]$. Множеството на мерења од i-тиот сензор што пипаѓаат на прозорецот w_k е означен со m_i^k, како што е преставено со (3.1). Исто така, интервалот $[ts_{k+1}, te_k]$ го дефинира сегментот во кој се преклопуваат прозорците w_k и w_{k+1}.

$$m_i^k = \sigma_{s=i \text{ and } t>ts_k \text{ and } t\leq te_k}(R) \tag{3.1}$$

Нека δ_i ја означува временската разлика помеѓу последователните мерења m_i на i-тиот сензор. Понатаму, нека μ го означува минималното временско растојание помеѓу последователните мерења на сите сензори, како што е дефинирано со (3.2). Нека a и b ги означуваат должината на прозорците и преклопувањето помеѓу соседните прозорци, соодветно, така што $a > b$ и $a, b \in \mathbb{N}^+$. Следува дека должината на подвижните прозорци може да се претстави како $w = a \times \mu$ и дека должината на сегментот на преклопување е $o = b \times \mu$.

$$\mu = \min_i \delta_i \tag{3.2}$$

Со цел да се определи бројот на примероци во еден подвижен прозорец (n_i) за i-тиот сензор, се користи (3.3), која прави заокружување нагоре и враќа цел број.

$$n_i = \lceil w/\delta_i \rceil \tag{3.3}$$

Во општ случај, различни сензори може да прават мерења со различна фреквенција. Ако сензорот l прави мерења многу ретко (т.е. има мала фреквенција), а

сензорот p мери многу често (има висока фреквенција), тогаш $\delta_l \gg \delta_p$. Следствено, исто така важи и $\delta_l \gg \mu$. Во некои случаи, $\delta_l \gg a \times \mu$ може дополнително да важи, што значи дека сензорите со ниска фреквенција може да не регистрираат мерење во некој временски прозорец. Но, дури и за таков сензор l, (3.3) ќе го заокружи бројот на мерења по прозорец на 1. Оваа равенка исто така имплицира дека сите сензори прават константен број на мерења во секој прозорец, што не мора да биде секогаш точно. Овие предуслови се воведени со цел да се гарантира дека во секој прозорец има барем едно мерење од секој сензор и дека бројот на мерења во еден прозорец е константен. Со ова значително се упростува автоматското инженерство на атрибути и дополнително гарантира дека сите сензори ќе бидат земени во предвид во секој подвижен прозорец. Од друга страна, ова ја нарушува (3.1). Поради тоа, овој предуслов се олабавува и се редефинира со релациона алгебра. За да се добијат n_i мерења што одговараат на подвижниот прозорец k, кој завршува во време te_k, се употребува изразот со релациона алгебра даден во (3.4). Овој израз ги одбира сите мерења од i-тиот извор во k-тиот прозорец во кој временскиот печат t е помал или еднаков на крајот на k-тиот прозорец te_k, и ги враќа последните n_i мерења.

$$dp_i^k = \sigma_{\text{rownum}() \leq n_i}(\tau_{t \text{ desc}}(\sigma_{s=i \text{ and } t \leq te_k}(R))) \tag{3.4}$$

Обично преклопувањето (o) е поголемо од минималното време помеѓу мерењата μ, но е од ист ред на големина. Должината на прозорците w обично е околу еден ред на големина поголема од μ. Ова евристичко правило е врз основа на повеќе истражувања како на пример: [79, 80, 51, 47].

За дадена должина на прозорецот w, должина на преклопувачкиот сегмент o и вкупно време на собирање на податоци T, бројот на тренинг примероци N може да се пресмета со (3.5).

$$N = \left\lfloor \frac{T - (w - o)}{w - o} \right\rfloor \tag{3.5}$$

3.1.2 Должина на подвижните прозорци

Должината на подвижниот прозорец (w) и должината на преклопувачките сегменти (o) треба да бидат дефинирани пред да се прави сегментација. Според прегледот презентиран во [81], големината на прозорците зависи од целната класа, односно активноста што треба да се препознае. На пример, активности како "одење" се пократки од "готвење". Помалите сензорски фреквенции или посложените активности наметнуваат подолги подвижни прозорци [78]. Кога се препознаваат предефинирани активности, користењето на прозорци со променлива должина соодветни за секоја активност може да биде од корист за перформан-

сите на системот. За да се овозможи ова, во секој момент треба да се прави сегментација со прозорци со различна должина. По генерирањето на атрибути од податоците во секој подвижен прозорец треба да се прават повеќе предвидувања, по едно за секој прозорец, кои понатаму треба да се агрегираат со цел да се направи конечно предвидување на активноста во одреден момент. Ваквиот пристап е посложен и поради тоа не го земаме во предвид во оваа дисертација. Наместо тоа, сметаме дека со систематско инженерство на атрибути може да се добијат атрибути кои ќе овозможат високи предиктивни перформанси. Резонираме дека должината прозорецот соодветна за најдолгата активност би била соодветна и за пократките доколку генерираните атрибути се робустни и информативни.

3.1.3 Методологија за систематско инженерство на атрибути од временски серии

Во ова поглавје е опишана предложената методологија за систематско генерирање на атрибути од временски серии, објавена во [1]. Многу пристапи прават фреквенциско филтрирање со цел да се отстрани шумот во мерењата или да се раздели сигналот на компоненти [40, 48, 49, 82]. На пример, кај проблемите со препознавање на активности со помош на акцелерометри, некои автори со ваквите методи на фрекевенциско филтрирање забрзувањето го разделуваат на гравитациско забрзување и на забрзување на телото на човекот [48]. Поради слични причини, некои пристапи прават мазнење (пеглање, smoothing, анг.) со подвижен просек пред генерирањето на атрибути [83, 84]. Бидејќи ориентацијата на сензорите е важна, овие филтрирачки техники се корисни [40]. Сепак, методологијата за систематско генерирање на атрибути од временски серии што ја предлагаме е поопшта и не е специјализирана за одредени типови на сензори. Всушност, изворите на податоци не мора воопшто да бидат сензори. Затоа, методологијата не применува техники за фреквенциско филтрирање. Тврдиме дека и без специјализирано филтрирање, методологијата сепак ќе генерира информативни атрибути кои ќе резултираат во робустни класификациони модели.

Кога податочните множества освен временските серии содржат и номинални атрибути, тогаш тие се трасформираат во номинални атрибути со помош на техниките опишани во Глава 3.2.

Моделирање на чувствителноста на промена со први изводи

По дефниција првите изводи ја опишуваат брзината на промена на некоја функција. За пресметување на прв извод од дискретен примерок, на пример вредности од некоја временска серија, може да се искористи следнава постапка.

За временска серија со вредности $v_i, 0 \leq i < n$ кои се добиени во време

$t_i, 0 \leq i < n$, може да се пресметаат $n-1$ први изводи со (3.6). Првите изводи повторно претставуваат временска серија од која понатаму може да се генерираат атрибути.

$$v_i^{'} = \frac{v_{i+1} - v_i}{t_{i+1} - t_i}, \quad \text{каде} \quad 0 \leq i < n-1 \tag{3.6}$$

Моделирање на сезоналноста на податоците

Често кај временските серии постојат сезонски компоненти кои може да се состојат од периодични, повторливи и генерално регуларни и предвидливи шаблони. Ова е особено видливо во деловни и финансиски податоци каде празниците, деновите од неделата, месецот од годината, тромесечието и слично може да имаат влијание на временските серии (пр. продажба, профит, курсна листа итн). Сезонските ефекти може да ги сокријат вистинските движења во временските серии и одредени несезонски карактеристики на податоците кои можеби се од интерес.

Постојат неколку главни причини зошто се анализираат сезонските варијации:

- Опишувањето на сезонскиот ефект овозможува подобро разбирање на неговиот допринос во временскте серии (пр. колку новогодишните празници ја зголемуваат продажбата на одреден тип на производи).
- Елиминирањето на сезонската компонента од временските серии може да помогне во изучувањето на други компоненти како циклични (пр. продажба во текот на летниот период) и нерегуларни варијации (пр. продажба на нов производ).
- Да се користи за градење на поквалитетни модели за предвидување на идни сезонски трендови.

Иако ваквите анализи се застапени при обработката на деловните податоци, истите пристапи може да се искористат и при моделирањето на произволни временски серии, како што се сензорски мерења. На пример, сезоналноста може да се однесува на личните карактеристики на луѓето чии акции се следат и препознаваат во системи со амбиентално помогнато живеење или на контекстот (пр. наутро, во текот на денот, навечер, итн)

Според [85], компонентите кои може да се присутни во временските серии се:

- Нерегуларна компонента е тоа што останува откако сезонската и тренд компонентата ќе се проценат и отстранат од временската серија. Оваа компо-

нента е резултат од краткорочните флуктуации што не се ниту систематски ниту предвидливи.
- Трендот се дефинира како долгорочните движења во временската серија без нерегуларни ефекти (како ефектите поврзани со календарот во финансиските податоци). Кај финанасиските податоци трендот е резултат на влијанија како раст на популацијата, инфлација на цените и генерални економски промени.

За моделирање на сезонските компоненти се користат следниве методи [85]:

1. Кај адитивниот модел на временска серија, сезонската компонента се дефинира со (3.7), каде S ја означува сезонската компонента, Y е измерената вредност на временската серија, T е трендот, C е цикличната вредност и I е нерегуларната вредност.

$$S = Y - (T + C + I) \tag{3.7}$$

2. Во мултипликативниот модел сезонската компонента се изразува како однос (или процент) како што е дадено во (3.8):

$$S = \frac{T \times S \times C \times I}{T \times C \times I} \times 100 = \frac{Y}{T \times C \times I} \times 100 \tag{3.8}$$

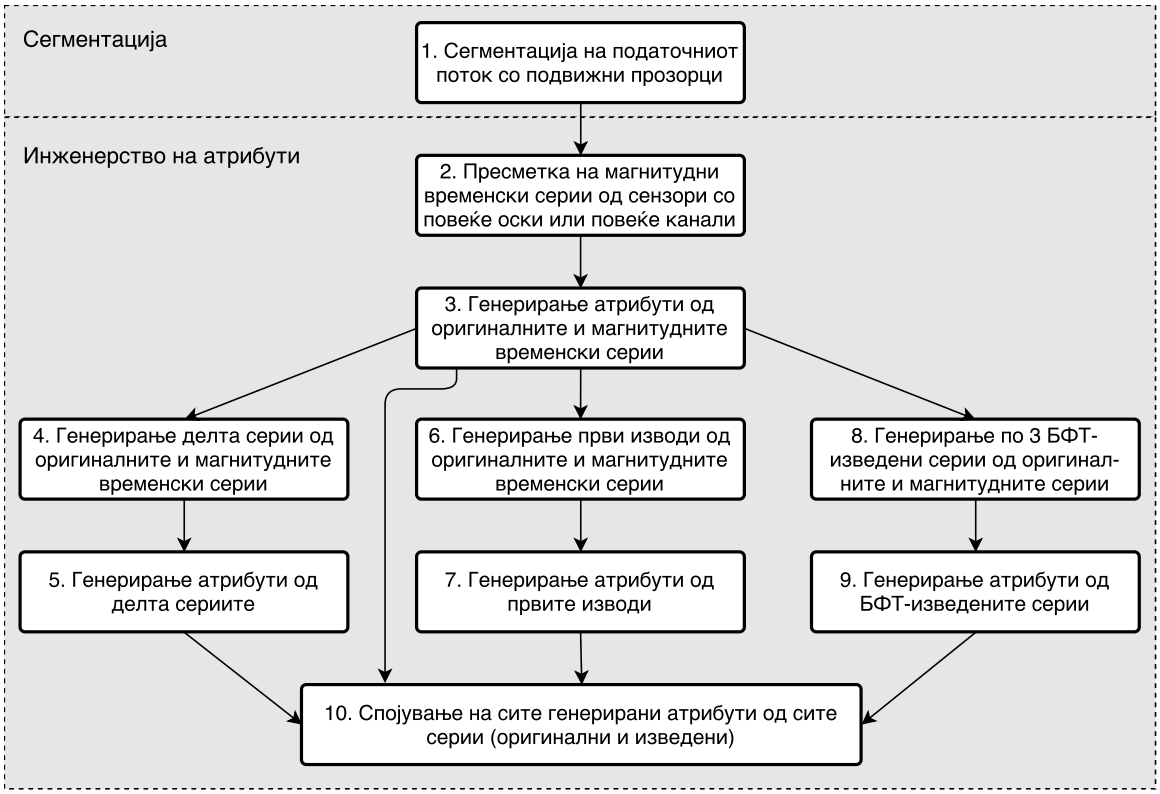
3. Десезоналираната временска серија ќе има само тренд (T), циклична (C) и нерегуларна (I) компонента и се изразува со (3.9) и (3.10) за адитивниот и мултипликативниот модел, соодветно:

$$Y - S = (T + S + C + I) - S = T + C + I \tag{3.9}$$

$$\frac{Y}{S} \times 100 = \frac{T \times S \times C \times I}{S} 100 = (T \times C \times I) \times 100 \tag{3.10}$$

Дополнитено постои и псевдо-адитивен модел, но тој не е искористен во оваа докторска дисертација, па затоа и не се разгледува во детали.

За сензорските податоци се смета дека адитивниот модел ќе биде посоодветен заради природата на податоците [15]. Дополнително, сметаме дека сезонската компонента на секоја временска серија во еден подвижен прозорец може да се моделира со аритметичката средина од вредностите во прозорецот. Токму поради тоа се воведени делта сериите, кај кои аритметичката средина се одзема од временските серии во рамките на еден прозорец, за што се дискутира подоцна во Глава 3.1.3.



Слика 3-2: Методологија за инженерство на атрибути од временски серии

Систематско генерирање на разновидни атрибути

Процесот на генерирање атрибути од временски серии е прикажан на Слика 3-2 и се состои од неколку чекори кои ги вклучуваат оригиналните временски серии и неколку типови на изведени временски серии.

Прво, во чекор 1, податочните текови се сегментираат во подвижни прозорци, како што е објаснето во претходното поглавје 3.1.1.

Потоа, во чекор 2, се генерираат магнитудни временски серии од сензорите кои прават мерења во повеќе оски (пр. жироскопи и акцелерометри) или во повеќе канали (пр. електро кардиоаграм - ЕКГ). Во случај на финансиски податоци, ова може да претставува податоци кои се добиваат од повеќе извори (пр. курсна листа на неколку берзи).

Од сите оригинални и магнитудни временски серии, чекор 3 ги генерира следниве типови на атрибути, кои се докажани како ефективни и предиктивни во повеќе натпревари поврзани со инженерство на атрибути од разновидни извори на податоци [51, 86]. Бројот на мерења во еден подвижен прозорец е означен со n.

- *Основни статистики*: минимум, максимум, опсег, аритметичка средина, хармониска средина, геометриска средина, модус, стандардна девијација, варијанса, искосеност (skewness, анг.), куртосис (kurtosis, анг.), однос сигнал-шум, енергија, енергија по примерок (energy per sample, анг.), со што се добиваат 14 атрибути по временска серија.
- *Хистограм со еднаква ширина (equal-width, анг.)* кој се добива со $\lceil log_2 n + 1 \rceil$ интервали, добиени со помош на правилото на Стурџес (Sturges, анг.) [87]. Хистограмот ја опишува веројатносната распределба на вредностите измерени во еден подвижен прозорец.
- *Атрибути добиени со перцентили*: прв квартал, медијана, трет квартал, интер-квартално растојание и некои други перцентили (5, 10, 20, 30, 40, 60, 70, 80, 90, 95), кои се употребени и во други студии [84, 88]. На овој начин од една временска серија се добиваат 14 атрибути. Слично на хистограмот, и перцентилите ја опишуваат веројатносната распределба.
- *Авто-корелација* од мерењата во еден подвижен прозорец [89, 90]. Нека τ го означува поместувањето на авто-корелацијата и нека τ се менува во доменот: $\tau \in [1, \lfloor \sqrt{n} \rfloor]$. За експоненцијално зголемување на вредноста на τ во тој домен, се пресметуваат класична авто-корелација и Пеарсонова корелација. Дополнително се пресметуваат двата типови на корелација користејќи ги вредностите од првата и втората половина на временскиот прозорец.
- *Пеарсонова корелација* помеѓу парови на временски серии [89, 90].
- *Коефициентите на линеарна и кдвадратна интерполација*. Линеарната интерполација е дефинирана со 2, додека квадратната со 3 коефициенти, со што се добиваат вкупно 5 атрибути [90].

Потоа, од оригиналните временски серии и нивните магнитуди, се генерираат нови временски серии: *први изводи*; *делта серии*, кои ги претставуваат релативните девијации од средната вредност на оригиналната серија во рамките на еден прозорец [15]; и сериите добиени со Брза Фуриева Трансформација (Fast Fourier Transformation - FFT) [91], во понатамошниот текст наречени *БФТ-изведени* временски серии: фреквенции, амплитуди и магнитуди [40, 80, 84].

Чекор 4 генерира *делта серии*, кои ги претставуваат релативните девијации од средната вредност на оригиналната серија во рамките на еден прозорец [15]. Прво се пресметува средната вредност $\overline{x}_i$ на мерењата x_i^j направени во рамките на еден подвижен прозорец на i-тата временска серија ($0 \leq j < n$, каде n е бројот на мерења во рамките на еден подвижен прозорец). Потоа се пресметуваат разликите $\Delta x_i^j = \overline{x}_i - x_i^j$ помеѓу оригиналното мерење x_i^j и пресметаната средна вредност $\overline{x}_i$. Притоа, средната вредност се пресметува во секој подвижен прозорец и за секоја временска серија посебно. Со ова, секое оригинално мерење x_i^j се мапира во нова вредност Δx_i^j. Овој чекор генерира толку делта временски серии колку што има оригинални и магнитудни временски серии.

Потоа од делта сериите, во чекор 5 се генерираат само атрибути кои се темелат на хистограми и перцентили, на ист начин како што се генерираат од

оригиналните временски серии.

Авто-корелациските атрибути од делта сериите се изоставуваат бидејќи тие се редундантни на авто-корелациските атрибути генерирани од оригиналните временски серии. Оваа редундантност е директна последица на дефиницијата на класична авто-корелација и Пеарсонова корелација. Атрибутите кои соодветсвуваат на интерполациските коефициенти исто така се изоставуваат бидејќи делта сериите претставуваат само линеарна транслација од оригиналните серии, така што и овие атрибути се редундантни. Од истите причини и повеќето основни статистики се редундантни, па затоа и тие се изоставуваат.

Слично на чекор 4, чекор 6 генерира *први изводи* од оригиналните и магнитудните временски серии [15]. Кога мерењата се прават со константна фреквенција, што е најчест случај, пресметката на првите изводи е тривијална бидејќи тие претставуваат разлика помеѓу последователните мерења во еден прозорец. Ако мерењата се прават со променлива фреквенција, тогаш во првите изводи треба да се земе во предвид и разликата на временските печати на мерењата. Од временска серија со n мерења во еден подвижен прозорец, временската серија на првите изводи има $n - 1$ вредности.

Чекор 7 генерира атрибути од временските серии на први изводи на сличен начин како и од оригиналните временски серии. Единствената разлика е тоа што авто-корелациските атрибути не се пресметуваат, поради причините што веќе се објаснети кај чекор 5.

Следно, чекор 8 ја пресметува брзата Фуриева трансформација (Fast Fourier Transformation - FFT, анг.), која во понатамошниот текст е означена со БФТ, од оригиналните и магнитудните временски серии, со што од секоја од нив се добиваат по три БФТ-изведени серии: фреквенции, амплитуди и магнитуди [40, 84].

Од секоја од БФТ-изведените серии, чекор 9 ги пресметува следниве атрибути: минимум, максимум, аритметичка средина, стандардна девијација, опсег, прв и трет квартал, меѓу-квартално растојание, медијана и 10-тиот, 40-тиот, 60-тиот и 90-тиот перцентил. Резонираме дека овие 13 атрибути ќе ја опишат веројатносната дистрибуција на вредностите во БФТ-изведените серии. Дополнително, од секоја оригинална и магнитудна временска серија се пресметува и спектралниот центроид, со што се добива уште еден атрибут од нив. Да резимираме, на крајот на чекор 9 се добиваат по $3 \times 13 + 1 = 40$ атрибути од фреквентиот домен на секоја оригинална и магнитудна временска серија.

Конечно, во чекор 10 се прави унија од сите атрибути генерирани на крајот на чекорите 2, 5, 7 и 9. Во оваа унија, потенцијално може да постојат редундантни и неинформативни атрибути. Во случај на голем број на оригинални временски серии, а како резултат на систематското генерирање на атрибути на секоја од нив, постои опасност од експлозија на атрибутите (feature explosion,

анг.), што може да резултира во огромен број на атрибути (десетици илјади или повеќе), поради што е неопходно одбирање на атрибутите. За надминување на овој проблем опишаната методологија за инженерство на атрибути се користи во комбинација со хибридниот метод за филтрирање на атрибути, опишан во глава 4.3.4.

3.2 Трансформација на номинални во нумерички податоци

Повеќето алгоритми за машинско учење може да се применуваат само на нумерички податоци заради примената на разни метрики за растојание. Многу деловни проблеми располагаат со хетерогени податоци и примената на алгоритми за машинско учење би била многу ограничена ако не се направи трансформација на атрибутите. Со трансформациите се добиваат на нумерички податочни множества, се менува дистрибуцијата на вредностите за подобро да одговара на хипотезата, итн.

Во зависност од податочните типови, различни трансформации на атрибутите се соодветни [92, 93]. Нумеричките податоци, без разлика дали се тоа нумерички или дискретни, може да се трансформираат на различни начини (пр. логаритамска трансформација, квадратна, реципрочна, квадратен корен, трет корен, експоненцијална трансформација итн) [92].

Од друга страна, трансформациите на номинални и категоријални податоци не се многу истражени [94], па затоа беа дел од предметот на истражување во оваа докторска дисертација.

3.2.1 Генерирање на прости променливи

Најчест начин за трансформација на номиналните атрибути е преку генерирање прости атрибути. Ова е едноставен начин кој не зависи од доменот на податоците, што се препорачува кога не постои мапирање од номиналните во нумерички вредности [70, 93, 72].

Со помош на оваа техника се генерираат n нови прости атрибути од секој номинален атрибут кој има n различни вредности, кога $n > 2$. Ако $n = 2$, тогаш се генерира само еден прост атрибут. Со (3.11) се пресметува вкупниот број на генерирани прости атрибути, каде v_i е бројот на различни вредности на i-тиот номинален атрибут. Последниот прост атрибут (v_i-ви) е линеарна комбинација од сите претходни, па затоа е редундантен и не се генерира.

$$z_{dummy} = \sum_{i=0}^{n} (v_i - 1) \tag{3.11}$$

Секој од генерираните атрибути може да има вредност 0 или 1, во зависност од тоа дали конкретната номинална вредност што ја претставува се појавила во оригиналниот номинален атрибут. Овој пристап прв пат е објавен во [95] и генерално се користел во регресиски анализи. Многу аспекти од употребата на прости променливи во регресиски анализи се дискутираат во [96]. Со текот на времето, оваа техника стана дел од повеќе софтверски пакети како стандарден чекор за трансформација на номиналните атрибути пред да се применуваат различни алгоритми за машинско учење. Кога бројот на номинални атрибути и бројот на различни вредности што може да ги имаат се мали, тогаш оваа трансформација дава добри резултати. Проблемите се случуваат кога податочното множество има многу номинални атрибути кои може да имаат голем број на различни вредности. Ова води до драстично зголемување на бројот на генерирани прости атрибути, со што се зголемуваат времето и потребната меморија за тренирање на алгоритмите за машинско учење. Овие проблеми се надминуваат со одбирање на атрибутите, сепак со ризик дека можеби некои корисни информации кои се содржани во отфрлените атрибути ќе бидат изгубени.

3.2.2 Тежина на фактите

Друга мерка за сличност на номинални атрибути е предложена во [97]. Со неа се дава поголема тежина на поретките вредности на атрибутите, притоа не правејќи претпоставки за дистрибуцијата на вредностите на атрибутите. Оваа метрика во [98] е применета за дефинирање на сличност помеѓу парови на објекти кај ненадгледувано учење.

Со параметарот тежина на фактите (Weight of Evidence, WoE, анг.), оригинално предложен во [99], може да се оценат доказите кои поддржуваат одредена хипотеза врз основа на расположливите податоци. Дополнително, овој параметар може да се искористи за трансформирање на номинални атрибути и примери за ова има во [100] и [92].

Бинарни проблеми

Во ова поглавје е опишана математичкиот модел за пресметување на параметарот тежината на фактите за бинарни класификациски проблеми [13, 20, 21].

Со (3.12) се дефинираа параметарот тежина на фактите на i-тата вредност на атрибутот A, каде N_i^A е бројот на инстанци што се означени како негативни

и P_i^A е бројот на инстанци што се означени како позитивни за i-тата вредност на атрибутот A. SN е вкупниот број на негативно означени примероци, PN е вкупниот број на позитивно означени примероци во тренинг множеството и n^A е бројот на различни вредности на атрибутот A.

$$WoE_i^A = ln\frac{N_i^A}{P_i^A} - ln\frac{SN}{SP} \tag{3.12}$$

Вредностите на SN и SP може да се пресметаат со (3.13) и (3.14), соодветно:

$$SN = \sum_{i=1}^{n^A} N_i^A \tag{3.13}$$

$$SP = \sum_{i=1}^{n^A} P_i^A \tag{3.14}$$

Информациона вредност

Користејќи ја дефиницијата на параметарот тежина на фактите, со (3.15) е дефинирана метриката информациона вредност [92] која ја оценува предиктивната способност на атрибутите. Иако оваа метрика може да се пресметува и за нумерички атрибути, сепак посоодветна е за номинални или дискретни вредности.

$$IV^A = \sum_{i=1}^{n^A} \left[\left(ln\frac{N_i^A}{SN} - ln\frac{P_i^A}{SP} \right) \times WoE_i^A \right] \tag{3.15}$$

Бинарни проблеми со незадоволени предуслови

Според (3.12) следува дека N_i^A и P_i^A треба да се различни од нула, но бидејќи тие претставуваат броења на инстанци, овие ограничувања се трансформираат во $N_i^A > 0$ и $P_i^A > 0$. Сепак, тие не се секогаш задоволени во податочните множества, па затоа употребата на овој параметар е ограничена. Начин на нивно надминување е предложен во [21, 20]. Различните типови на незадоволени предуслови [21] се опишани во продолжение:

Случај 1: *Бројот на позитивно и негативно означени инстанци е нула ($P_i^A = 0$ и $N_i^A = 0$, соодветно.* Ова е тривијален случај кога не постојат инстанци

со i-тата вредност на атрибутот A, иако оваа вредност припаѓа на доменот на вредности на оригиналниот атрибут. Во овие случаи претпоставуваме дека WoE_i^A е нула, што значи оваа вредност на атрибутот A нема влијание на трансформацијата.

Случај 2: *Бројот на позитивно означени инстанци е нула ($P_i^A = 0$) и бројот на негативно означени инстанци е поголем од нула ($N_i^A > 0$).* За да се примени (3.12), е предложено да се употреби вредноста $P_i^A = 1$ за позитивно означените инстанци. Истовремено, потребно е да се додаде соодветниот број на негативно означени инстанци, така што вкупниот однос на додадени позитивно и негативно означени инстанци е константен и ист како и во целото податочно множество (SN/SP). Во следните равенки бројот на вештачки додадени позитивно означени и негативно означени инстанци се претставени со $\delta p_i^A = 1$ и δn_i^A, соодветно. Овие вештачки "додавања" на инстанци само се применуваат во формулите за пресметка на тежината на факти. Со (3.16) и (3.17) се дефинира бројот на додадени инстанци, а предложената формула за пресметка на тежината на факти по додавањата е дадена со (3.21).

$$\frac{\delta p_i^A}{\delta n_i^A} = \frac{SP}{SN} \tag{3.16}$$

Со замена на $\delta p_i^A = 1$ во (3.16), δn_i^A може да се пресмета со (3.17):

$$\delta n_i^A = \frac{SN}{SP} \tag{3.17}$$

Сега P_i^A и N_i^A може да се променат, така што ќе ги вклучуваат вештачки додадените инстанци:

$$\Delta P_i^A = P_i^A + \delta p_i^A = 1 \tag{3.18}$$

$$\Delta N_i^A = N_i^A + \delta n_i^A = N_i^A + \frac{SN}{SP} \tag{3.19}$$

Така, наместо тежината на факти да се пресметува со (3.12) користејќи ги N_i^A и P_i^A, истата може да се пресмета со (3.20) која ги користи променетите вредности ΔN_i^A и ΔP_i^A, дефинирани претходно:

$$WoE_i^A = ln\left(\frac{\Delta N_i^A}{\Delta P_i^A}\right) - ln\left(\frac{SN}{SP}\right) \tag{3.20}$$

Ако се заменат (3.18) и (3.19) во (3.20), конечно се добива предложената формула за пресметка на тежината на фактите за овој случај на незадоволени предуслови:

$$WoE_i^A = ln\left(\frac{N_i^A \times SP + SN}{SN}\right) \tag{3.21}$$

Случај 3: *Бројот на негативно означени инстанци е нула ($N_i^A = 0$) и бројот на позитивно означени инстанци е поголем од нула ($P_i^A > 0$).* Слично на претходниот случај, предложено е вештачко додавање на една негативно означена инстанца, така што при премсетката на (3.12) ќе се земе дека $N_i^A = 1$. Исто така, се додава соодветен број на позитивно означени инстанци, за бројот на додадени инстанци да одговара на односот на позитивни и негативни инстанци генерално (SN/SP). Во овој случај бројот на додадени негативни инстанци е ($\delta n_i^A = 1$), па (3.16) може да се трансформира во (3.22):

$$\delta p_i^A = \frac{SP}{SN} \tag{3.22}$$

Сега P_i^A и N_i^A може да се променат да ги вклучуваат вештачки додадените инстанци:

$$\Delta P_i^A = P_i^A + \delta p_i^A = P_i^A + \frac{SP}{SN} \tag{3.23}$$

$$\Delta N_i^A = N_i^A + \delta n_i^A = 1 \tag{3.24}$$

Конечно, за овој случај на незадоволени предуслови, наместо да се користи (3.12), за пресметка на тежината на факти се користи (3.25) со помош на ΔN_i^A и ΔP_i^A, дефинирани претходно:

$$WoE_i^A = ln\left(\frac{SP}{P_i^A \times SN + SP}\right) \tag{3.25}$$

Предложената трансформација е осетлива на шум и во таков случај трансформираните вредности може да не соодветствуваат на реалниот релативен ризик. Сепак, шумот генерално претставува проблем во податочните множества и треба да се преземат мерки за негово отстранување или барем намалување уште пред да се прават трансформации на податоците.

Повеќе-класни проблеми

Едно од најограничувачките својства на параметарот тежина на фактите е тоа што е употреблив само за бинарни класификациони проблеми. Од друга страна, кај многу реални проблеми потребно е класифицирање во повеќе од две класи. За да се надмине овој проблем во [13] е предложена и евалуирана надградба на параметарот тежина на фактите врз основа на идеите изложени во [19]. Тоа е постигнато со претставување на повеќекласните проблеми како множество на бинарни подпроблеми. Потоа, за секој од новите бинарни подпроблеми се пресметува параметарот тежина на фактите. Овој пристап, уште наречен и еден-против-сите е употребен во [56] и [57] за генерализација на повеќе алгоритми за машинско учење кои иницијално поддржуваат само две класи (на пример машини со носечки вектори).

Алгоритмот 1 се повторува за секоја од класите. Со овој алгоритам кај податочно множество со k номинални атрибути и m класи се генерираат z_{woe} нови нумерички атрибути, како што е дефинирано со (3.26).

Алгоритам 1 Генерализација за повеќекласни проблеми на параметарот тежина на фактите со методот еден-против-сите

for $i = 1 \rightarrow m$ **do**

Привремено означи ги со $TempClass^1$ сите инстанци кои оригинално биле означени со $Class^i$.

Привремено означи ги со $TempClass^2$ сите инстанци кои оригинално биле означени со некоја класа различна од $Class^i$.

Пресметај ги вредностите тежина на фактите за секоја инстанца и за сите нејзини номинални вредности користејќи ги привремените ознаки $TempClass^1$ и $TempClass^2$.

Трансформирај ги сите k номинални атрибути користејќи ги пресметатните тежини на фактите во претходниот чекор, со што се генерираат k нови нумерички атрибути.

Додади ги генерираните k нумерички атрибути во трансформираното податочно множество.

Отстрани ги привремените ознаки на сите инстанци и врати ги оригиналните ознаки (класи).

end for

$$z_{woe} = \begin{cases} m \times n, & m > 2 \\ n, & m = 2 \end{cases} \tag{3.26}$$

Истиот алгоритам може да се примени за трансформација и на нумерички атрибути, но со оглед на тоа што веќе постојат многу трансформации за нумерички атрибути, истражувањето не се фокусира на ваквата потенцијална примена на оваа трансформација.

Вкрстена валидација кај повеќе-класни проблеми

Користењето на вкрстена валидација, опишано во Глава 2.1.3, е често употребуван начин за евалуација на перформансите на алгоритмите за машинско учење [54]. Постојат повеќе начини за правење на вкрстена валидација, од кои најчесто употребувана е вкрстената валидација со 10 групи [54]. Вообичаено вкрстената валидација се прави после предобработката на податоците, односно откако ќе се направат сите потребни трансформации. Овој пристап е соодветен кога трасформациите претставуваат функции, односно трансформираната вредност зависи само од оригиналната вредност што треба да се трансформира.

Спротивно на тоа, тежината на фактите дополнително зависи и од односот помеѓу инстанците во тренинг множеството, како што може да се забележи од (3.12), па затоа стандардниот пристап за вкрстена валидација не е соодветен. Бидејќи при вкрстената валидација односот помеѓу инстанците во тренинг множеството кое се користи во секоја итерација се менува, потребно е соодветно да се промени процесот на вкрстена валидација, што е прикажано со Алгоритмот 2. Користејќи го овој алгоритам, се гарантира дека при градењето на моделите се користат информации кои се присутни само во тренинг множеството. Резултатите од истражувањата со помош на овој алгоритам се изложени во [13].

Алгоритам 2 Трансформација на номиналните атрибути при вкрстена валидација со K-групи

Случајно размешај го податочното множество
Случајно додели група $Fold_i$ $(i = 1 \rightarrow k)$ на секоја инстанца во размешаното множество
for $i = 1 \rightarrow k$ **do**
Додели ги инстанците кои припаѓаат на групата $Fold_i$ на тест множеството $TestSet_i$.
Додели ги инстанците кои не припаѓаат на групата $Fold_i$ на тренинг множеството $TrainingSet_i$.
Пресметај ги тежините на факти, WoE_i, за сите инстанци во тренинг множеството $TrainingSet_i$.
Трансформирај ги сите вредности на сите номинални атрибути во $TrainingSet_i$ користејќи ги тежините на факти WoE_i, со што се добива трансформираното тренинг множество $TransformedTrainingSet_i$.
Трансформирај ги сите вредности на сите номинални атрибути во $TestSet_i$ користејќи ги тежините на факти WoE_i, со што се добива трансформираното тест множество $TransformedTestSet_i$.
Тренирај го моделот $Model_i$ со $TransformedTrainingSet_i$.
Валидирај го моделот $Model_i$ со $TransformedTestSet_i$, со што се добиваат резултатите $Results_i$.
end for
Агрегирај ги резултатите $Results_i (i = 1 \rightarrow k)$

Глава 4

Одбирање на атрибути

Одбирањето на атрибути е клучен процес кој претходи на примената на алгоритмите за машинско учење. Оваа фаза е од исклучителна важност особено кога податочните множества имаат огромен број на атрибути. Многу проблеми во биоинформатиката и проблеми поврзани со обработка на текстуални документи содржат илјадници атрибути. Особено каракактеристични се проблемите со обработка на текст каде може да се случи податочното множество да има десетици па дури и стотици илјади атрибути. Ваквиот огромен број на атрибути наметнува сериозни предизвици за алгоритмите за машинско учење. Преку идентификување на најветувачките атрибути, алгоритмите за учење се насочуваат на оние аспекти од податоците кои се најкорисни за анализа и предвидување. Во множествата на генерирани атрибути може да постојат нерелевантни и редундантни атрибути кои може да ги намалат предиктивните перформанси на класификационите модели, а притоа да го зголемат потребното време и мемориска комплексност за нивно градење. Дополнителна придобивка од одбирањето на атрибути е добивање попрости и покомпактни модели кои се поедноставни за толкување, визуализација и разбирање. Многу алгоритми за учење како невронски мрежи, дрва за одлучување, наивен Баесов класификатор и други, се соочуваат со ризик од деградација на перформансите во присуство на редундантни и нерелевантни атрибути, за што се изложени теоретски и емпириски докази во повеќе студи, меѓу кои се и [101, 102, 103].

Различните методи за одбирање на атрибути се фокусираат на различни цели или ги постигнуваат истите цели, но на различни начини. Обично одбирањето на најрелевантните атрибути е помалку оптимално отколку одбирањето на подмножество од атрибути кои се корисни за градење на добар предиктивен модел, особено ако помеѓу нив има редундантни атрибути. Во [103, 104] опширно е опишана разликата помеѓу релевантноста и корисноста на атрибутите. Во [105] се изложени повеќе препораки за одбирање на атрибутите и се претставени најчесто употребуваните методи. Алгоритмите за одбирање на атрибути се делат

на две главни типови: филтрирачки и обвиткувачки методи.

Во оваа глава накратко се опишани различните типови на методи за одбирање на атрибутите кои постојат во литературата и детално е даден опис на предложените хибридни методи за одбирање на атрибутите.

4.1 Филтрирачки методи

Методите од оваа група прават рангирање на атрибутите според некоја метрика. Овие методи се карактеризираат со едноставност, скалабилност и добра емириска издржаност. Заради едноставноста тие имаат многу помала мемориска и пресметковна комплексност и релативно лесно може да се употребуваат дури и кај големи податочни множества со илјадници атрибути. Подетален опис на ваква примена на овие алгоритми е даден во [106] и [107], додека во [108] е дадена детална емириска анализа на истите. Овие методи се независни од алгоритмот за машинско учење кој ќе се применува понатаму.

Дополнително, филтрирачките методи се делат на две подгрупи. Првата подгрупа се однесува на алгоритми кои тежински ги рангираат индивидуалните атрибути. Најчесто употребувани метрики за проценка на предиктивноста на атрибутите се: информационата вредност [92], информациона добивка [109, 110], однос на информациона добивка [109, 110], RELIEF [111, 112], ентропија [113] и други. Во [114] и [115] се претставени методи кои се темелат на постериорна веројатност.

Кај овие методи најголемиот проблем е тоа што тие ја земаат во предвид само индивидуалната корист од атрибутот во однос на целната класификација, но не можат да откријат редундантност и меѓу-зависност помеѓу одбраните атрибути [104]. Во втората подгрупа припаѓаат методи кои анализираат подмножества од атрибути врз основа на некоја метрика која ги опишува перформансите на подмножеството [116]. Токму од овој тип се методите кои се темелат на корелација [117, 118]. Имено, со овие алгоритми се бараат подмножества на атрибути кои имаат мала корелација помеѓу себе, а голема корелација со целната класификација.

4.1.1 Рангирање по информациона вредност

Рангирањето според информациона вредност (information value, анг.) [21, 92] зависи од пресметката на параметарот тежина на фактите [21]. Според оригиналната дефиниција, метриката информациона вредност е корисна за рангирање на атрибути кај проблеми со надгледувано учење каде целната класификаци-

ја е бинарна. Сепак, кај многу реални проблеми класификацијата е повекласна или повеќезначна. Во тие случаи, користејќи го проширувањето на параметарот тежина на фактите, предложено во [13], може да се пресметаат повеќе информациони вредности, по една за секој пар на целни класи. Потоа со различни типови на агрегација (пр. просек, тежински просек, минимум, максимум итн) овие информациони вредности може да се претстават со една вредност која би се искористила за рангирање и одбирање на атрибутите. Овој пристап е искористен за одбирање на атрибути кај повеќезначна класификација во [18].

4.1.2 Рангирање по информациона добивка

Информационата добивка (information gain, анг.) е синоним за Кулбек-Либлерова дивергенција (Kullback–Leibler divergence, анг.) и најчесто се употребува за рангирање на индивидуални атрибути, како што е опишано во [119, 120]. Исто така, информационата добивка може да се употребува за категоризација на текст, како што е опишано во [109].

Еден од најчесто употребуваните алгоритми за дискретизација на континуални атрибути [121] во основа ја користи информационата добивка. На сличен начин во [122] таа се користи за ненадгледувана дискретизација на континуални атрибути. Методот предложен во [123], кој ги дискретизира континуалните атрибути со помош на алгоритам кој користи информациона добивка, се употребува за подобрување на перформансите на дрвата за одлучување.

Овие трудови се многу влијателни и инспирираа друга примена на информационата добивка и ентропијата. Во [124] се предложени методи кои се темелат на филтри со честички (particle filters, анг.) и информационата добивка се користат за истражување, мапирање и локализација. Друга примена е за проширување на нотацијата на редукти кај груби множества (rough sets, анг.) [125], каде уште се употребува и за пресметување на минимални подмножества на атрибути, а притоа задржувајќи ја информацијата за целната променлива на задоволително ниво.

Во [4] информационата добивка и методот за дискретизација предложен во [121] се употребени за пресметка на прагови на интервалите (cut-off values, анг.) на различни параметри во медицината.

За да се пресмета информационата добивка, прво треба да се пресмета ентропијата $H(X)$ на податочното множество. Нека X го означува тренинг множеството и нека секоја инстанца во него x_i е во облик $(x_i^1, x_i^2, ..., x_i^k, y_i)$. Нека секој атрибут (т.е. колона) е дискретна случајна променлива чиј домен е множеството вредности $V^j, j = 1..k$. Нека множеството на можни ознаки (т.е. категории или класи) е L, така што $y_i \in L$. Тогаш ентропијата на множеството X може да се пресмета со (4.1), каде $p(l)$ е веројатноста за инстанцата x_i да биде означена со

класата l (т.е. $y_i = l$) која се пресметува со (4.2).

$$H(X) = -\sum_{l \in L} p(l) \log p(l) \tag{4.1}$$

$$p(l) = \frac{|\{x_i \in X | y_i = l\}|}{|X|} \tag{4.2}$$

Информационата добивка на j-тиот атрибут од податочното множество X може да се пресмета со (4.3), каде првиот множител во сумата ја претставува веројатноста j-тиот атрибут на некоја инстанца да има вредност v. Вториот множител во сумата во (4.3) ја означува ентропијата на подмножеството на инстанци што ја имаат вредноста v на j-тиот атрибут.

$$IG(X, j) = H(X) - \sum_{v \in V^j} \frac{\left|\left\{x_i \in X | x_i^j = v\right\}\right|}{|X|} H(\left\{x_i \in X | x_i^j = v\right\}) \tag{4.3}$$

Како што се гледа од (4.1), (4.2) и (4.3), пресметката на информационата вредност на сите атрибути се сведува на броење на инстанците по атрибут, вредност и класа. Откако ќе се избројат, може да се пресметаат веројатностите а потоа и информационата вредност. Во Глава 5.5 е опишана паралелна имплементација за пресметување на информационата вредност на секој атрибут во множеството.

4.1.3 Џини индекс на нечистотија

Џини индексот на нечистотија (Gini impurity index, анг.), слично на ентропијата и средната квадратна грешка, е мерка за нечистотија на множества. Таа е дефинирана како вкупното намалување на нечистотијата (тежински усреднето од веројатноста да се стигне до одреден јазол во дрвото за одлучување) усреднето од сите дрва во ансамбалот [126, 127, 60]. Таа е посебно корисна бидејќи веќе е имплементирана во повеќе софтверски пакети со алгоритми за машинско учење како sci-kit learn [128], Mahout [129], WEKA [71] и MLLIB [130], и се користи за проценка на важноста на атрибутите кај ансамбли од дрва за одлучување како кај случајни шуми [131, 60, 128] или екстремно случајни дрва [61, 128]. Во истражувањата опишани во оваа докторска дисертација овие алгоритми за машинско учење се користат повеќекратно, па тренирајќи модели воедно се добива и проценка на важноста на атрибутите. Ова е од особена важност бидејќи веќе постојат имплементации на овие алгоритми во паралелна средина [129, 130], па во случаите кога овие алгортми веќе се употребуваат не е потребна дополнително да се имплементираат алгоритми за проценка на важноста на атрибутите.

4.1.4 Одбирање на атрибути со корелациска анализа

Во [117] е опишан еден од најчесто употребуваните алгоритми за одбирање на атрибутите кој ги испитува перформансите на цели множества на атрибути од еднаш. Сличен пристап е опишан и во [118]. Имено, со овие алгоритми се бараат подмножества на атрибути кои имаат ниска корелација помеѓу себе, а висока корелација со целната класификација. Кога бројот на атрибути е мал, тогаш може да се пресметаат корелациите на многу парови на атрибути однапред и потоа ова да се искористи за водење на пребарувањето на просторот од можни множества на атрибути. За разлика од обвиткувачките пристапи, во овие пристапи не се евалуираат перформансите на одбраните множества од атрибути со тренирање на класификатори, туку само се разгледува корелацијата помеѓу одбраните атрибути, која треба да биде ниска, и нивната корелација со целната класификација, која треба да биде висока. Кај овие пристапи проблеми може да настанат кога бројот на атрибути е многу голем или ако постојат номинални атрибути што ја отежнуваат пресметката на корелацијата.

4.2 Обвиткувачки методи

Обвиткувачките (wrapper, анг.) методи наоѓаат подмножество на атрибути кои се корисни за дадениот класификациски проблем. Тие ги испитуваат перформансите на различни подмножества од атрибути со употреба на алгоритам за машинско учење [104, 132, 133]. При употребата на овие методи не се испитува индивидуалниот придонес на одредени атрибути туку на подмножеството на одбрани атрибути и притоа целиот процес е со карактеристики на црна кутија. Корисникот не знае зошто точно е одбрано конкретното подмножество на атрибути.

За во пракса да се употребува одреден обвиткувачки метод потребно е да се дефинираат следните работи:

- Како да се пребарува просторот на сите можни подмножества?
- Како да се евалуираат перформансите на алгоритмот за учење со цел да се води и насочува пребарувањето?
- Кој алгоритам за учење да се користи?

Доколку бројот на атрибути е мал, тогаш може да се испробаат сите комбинации, но ретко кога тоа е случај. Главен проблем на овие методи е големата пресметковна комплексност. Сепак, постојат повеќе техники на пребарување на просторот кои делумно го олеснуваат овој проблем. Од друга страна, огромна предност на обвиткувачките методи е нивната универзалност и независност од доменот на податоците.

Пристапот опишан во [134] користејќи генетски аглоритам одбира атрибути кои се погодни за тренирање на машини со носечки вектори. Во [135] е предложен и искористен генетски алгоритам за наоѓање на оптималното множество на атрибути кој користи динамички параметри на генетскиот алгоритам во зависност од епохата во која се наоѓа.

4.3 Хибридни методи

Освен претходните два типови на методи, постојат и комбинации од нив кои се нарекуваат хибридни методи. Целта на овие методи е преку комбинирање да се надминат недостатоците кои ги имаат филтрирачките и обвиткувачките методи. Во [136] е направена споредба помеѓу трите типови на методи за селекција на атрибути, додека во [137] се споредуваат различните хибридни пристапи и нивната примена за анализа на секвенци од гени. Авторите во [138] предлагаат пристап заснован на генетски алгоритам и взаемна информација. Во [139, 140] се предложени хибридни пристапи за одбирање на атрибути и истите се применуваат на проблеми со анализа на временски серии кај берзите. Одбирањето на атибути со помош на хибридни методи за дијагностицирање на болести е опишано во [141], додека во [142] е предложен метод кој е искористен за предвидување на протеински функции.

Во [143] е предложен метод за наоѓање на речиси оптималното множество на атрибути за големи податочни множества применувајќи алчна стратегија на пребарување на просторот на множества на атрибути. На истото податочно множество како, сепак пристапот со генетски алгоритам опишан во [135] дава подбори резултати, за сметка на значително зголемената временско-мемориска комплексност.

Ваквиот проблем е надминат во [144], каде е предложен хибриден обвиткувачки и филтрирачки метод за одбирање на атрибутите кај класификациски проблеми. Овој пристап прави рангирање на атрибутите со помош на филтрирачки метод за да се забрза пребарувањето на множествата од атрибути кај генетски алгоритам. Во студијата емпириски е покажано дека со хибридниот метод се добиваат подобри класификациски перформанси, се одбираат помали множества на атрибути и моделите се градат побрзо.

4.3.1 Комбинација од рангирачки и корелациски методи

За надминување на проблемите кај корелациските методи опишани во Глава 4.1.4, во [18] е предложен хибриден пристап кој прво ги рангира атрибутите и отстранува тие што не се предиктивни според информациската вредност, опи-

шана во Глава 4.1.1, и потоа на значително намаленото множество на атрибути прави одбирање со помош на корелациска анализа. Во пристапите со корелациската анализа во општ случај би било потребно да се пресмета корелацијата помеѓу сите парови на атрибути. Така, за n атрибути, би биле потребни пресметки на $\frac{n(n-1)}{2}$ коефициенти на корелација. Но, ако со филтрирачки метод се отстранат само половина од атрибутите, тогаш во втората фаза ќе треба да се пресметаат само $\frac{n(n-2)}{8}$ коефициенти на корелација, што е приближно една четвртина од оригиналниот број. Дури и кога не се пресметуваат сите парови на корелации однапред и наместо тоа се користи некој метод за пребарување на просторот кој ги пресметува корелациите само помеѓу потребните парови на атрибути, со намалувањето на бројот на атрибути се олеснува пребарувањето на просторот и полесно се откриваат оптималните подмножества.

4.3.2 Откривање на атрибути кои се осетливи на концептуални отстапувања

Во реалните проблеми, карактеристиките на податочните множества во текот на времето може да се менуваат поради што изградените модели може да станат неупотребливи. За поголема робустност на генерираните модели и спречување на претренирање (overfitting, анг.) корисно е да се пронајдат атрибутите кои се осетливи на примени на дистрибуцијата на вредности. Ова уште се нарекува концептуално отстапување (concept drift, анг.). На пример, некој атрибут може да има нормална дистрибуција во сите податочни множества, но да има значително различна средна вредност и стандардна девијација во тренинг множеството од валидациското и тест множеството. Дополнително, некој атрибут може да има целосно различна веројатносна дистрибуција во трите податочни множества, како на пример нормална, логаритамски-нормална, мулти-модална, униформна, итн. Овие промени на веројатносната дистрибуција често водат кон претренирање на моделите, поради што тие одговараат само на тренинг множеството [145].

За да може да се идентификуваат и отстранат ваквите атрибути, потребно е да се квантифицира нивната осетливост на концептуални отстапувања. Еден пристап за оваа намена е предложен во [145], кој подоцна успешно е употребен и во [146, 147]. Во оригиналната дефиниција ова е рангирачки пристап, кој ги рангира атрибутите по два критериуми.

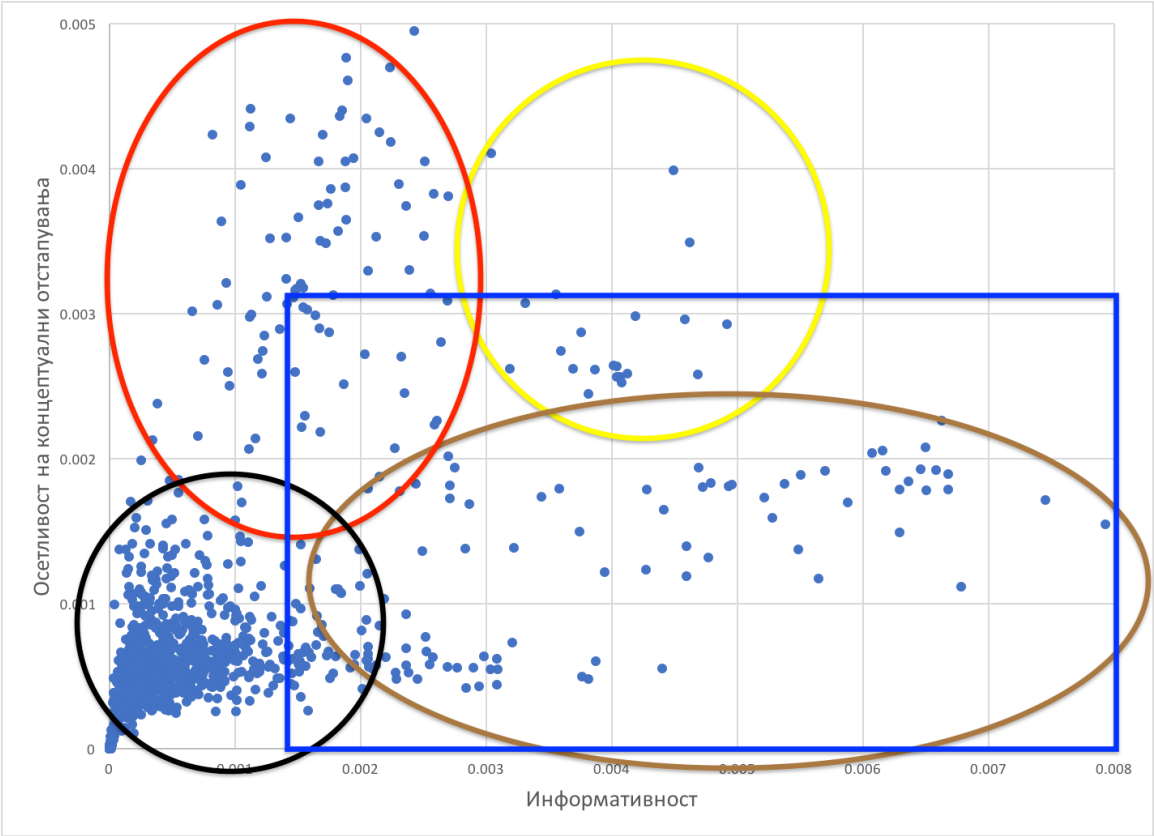
Алгоритмот за пребарување на просторот при одбирањето на атрибути со овој пристап е формализиран во [7] и е изложен во Изворниот код 4.1. Притоа тој е проширен во хибриден метод, кој после рангирањето на атрибутите прави алчно пребарување на просторот ги евалуира множествата на атрибути со обвиткувачки пристап. Најпрво се прави унија од тренинг множеството Xt и валидациско множеството Xv, при што уште се генерира и вештачка ознака (класа) ya. Оваа вештачка класа означува дали инстанцата потекнува од тре-

нинг или валидациското множеството (линии 2-3). Важно е евалуацијата да се изврши над валидациско множество кое ќе биде различно од тренинг и тест множествата. Потоа ја проценува информативноста на секој атрибут за предвидување на регуларната класа и дополнително информативноста за предвидување на множеството од кое потекнува инстанцата (променливите tr_all и td_all во линиите 4-5). Треба да се потенцира дека информативните атрибути кои се важни за предвидвување на множеството од кое потекнуваат инстанците се всушност многу осетливи на конецепутални отстапувања и тие треба да бидат отстранети [145]. Потоа, се пресеметуваат множество на прагови на информативност за двете класификации (линиите 6-8). Овие прагови се наоѓаат пресметувајќи 11 перцентили од листата на информативности на атрибутите (линија 7) и од листата на осетливости на концептуални промени (линија 8). На крај правејќи комбинации од пресменаните прагови се наоѓаат различни множества на атрибути (вкупно $11 \times 11 = 121$) за кои се тренира класификациски алгоритам и се одбира тоа множество на атрибути што дава највисоки перформанси.

```
1  def find_features(Xt, yt, Xv, yv):
2      Xa = merge(Xt, Xv)
3      ya = merge(zeros(len(yt)), ones(len(yv)))
4      fir = get_feat_importance(Xt, yt)
5      fid = get_feat_importance(Xa, ya)
6      p = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
7      tr_all = calc_percentiles(fir, p)
8      td_all = calc_percentiles(fid, p)
9      f_all = list(range(size(Xt, 1)))
10     best_tr = 0
11     best_td = 100
12     best_fs = f_all
13     best_score = get_score(Xt, yt, Xv, yv, f_all)
14     for ctr in tr:
15         for ctd in td:
16             cfs = [i in f_all if fir[i]>=ctr and fid[i]<=ctd]
17             cfs_score = get_score(Xt, yr, Xv, yv, cfs)
18             if fs_score > best_score:
19                 best_td = ctd
20                 best_tr = ctr
21                 best_fs = cfs
22                 best_score = cfs_score
23 return best_fs
```

Изворен код 4.1: Алгоритам за хибридно одбирање на атрибути според информативност и осетливост на концептуални отстапувања

За да се изврши овој начин на одбирање на атрибутите потребно е да се дефинираат метрика на евалуација (точност, прецизност, површина под крива-та на применикот и слично) на моделот, метрика за рангирање на атрибутите по информативност и алгоритмот за класификација. Првиот избор зависи од проблемот кој се обработува и која метрика е соодветна за него. За класификациският алгоритам пожелно е да се употребува брз алгоритам дури и кога бројот на атрибути е голем. Од извршените експерименти опишани во [6, 7, 10, 12, 15] заклучивме дека ансамблите од дрва за одлучување се најсоодветен избор за овој алгоритам бидејќи успешно може да се справат со голем број на атрибути. Ова не секогаш е случај дури и кај поедноставните алгоритми како логистичка регресија и k-најблиски соседи, кои може да се успорат до неупотребливост.



Слика 4-1: Информативност и осетливост на концептуални отстапувања на атрибутите генерирани за податочното множество SBHARPT

Генерално може да се случи некои атрибути иако се многу информативни да бидат отстранети затоа што се премногу осетливи на концептуални отстапувања и водат кон претренирање на моделот. На Слика 4-1 се претставени генерираните атрибути за податочното множество SBHARPT [1, 49] за кое се дискутира во повеќе детали во Глава 6.7. Координатата x ја означува информативноста, додека координатата y ја означува осетливоста на концептуални отстапувања на атрибутите. Идеалните атрибути имаат мала вредност на y, а голема вредност на x (кафена елипса). Атибутите кои се блиску до координатниот почеток не се информативни и најверојатно треба да се отстранат (црна кружница). Атрибутите кои имаат високи вредности на y се осетливи на концептуални промени и тие може да предизвикаат претренирање (црвена кружница). Сепак, постојат атрибути кои се информативни, но и осетливи на концептуални промени (жолта елипса) и за нив е потешко одлучувањето дали да останат или да се отфрлат. Заради тоа, со предложениот алгоритам, тестирајќи повеќе прагови прилагодени за секое податочно множество, се одбира подмножество од атрибутите кое дава најдобри перформанси на валидациското подмножество (плав правоаголник). Во рамките на одбраното подмножество можеби постојат редундантни атрибути и пожелно е тие да се отстранат, за што може да се употреби алгоритмот опишан во Глава 4.3.3.

4.3.3 Диверзифицирано пребарување напред-назад за одбирање на атрибути

Во [143] е предложен метод за наоѓање на речиси оптималното множество на атрибути за големи податочни множества применувајќи диверзифицирано пребарување напред-назад за одбирање на атрибути. Овој метод е хибриден и користи рангирање на атрибутите за да го насочи пребарувањето и потоа за секое множество атрибути што се испитува, се тренира класификациски модел кој се евалуира со валидациско подмножество.

Сепак, оригиналната имплементација на овој алгоритам е секвенцијална, поради што и покрај алчниот пристап во пребарувањето, перформасите на алгоритмот не се задоволителни поради високата цена на тренирање на класификатор за секое ново податочно множество. За надминување на овој проблем, во [1] овој алгоритам е паралелизиран, така што истовремено се испитуваат повеќе множества на атрибути. Дополнително, оваа имплементација користи евристика која дополнително го намалува просторот за пребарување.

Овде пребарувањето на просторот започнува со празно множество на атрибути во кое се додаваат атрибути еден по еден доколку ги подбруваат перформансите на тековно најдоброто множество на атрибути. Поради тоа, дури и кога вкупниот број на атрибути е голем, множествата на атрибути што се евалуираат се релативно мали и тие постепено растат сè додека има подобрување на перформансите.

Постапката најпрво започнува со рангирање на атрибутите, со цел прво да се земат во предвид поинформативните атрибути. Потоа во една итерација на изминување напред се додаваат атрибути, при што во паралелната имплементација повеќе податочни множества се евалуираат од еднаш. Атрибутите кои не помогнале кога последен пат биле додадени во одредено множество на атрибути повеќе не се земаат во предвид. Изминувањето напред завршува кога со додавање на атрибути не се постигнува подобрување на перформансите. Потоа започнува изминување наназад, односно отстранување на атрибути од најдоброто множество на атрибути. Со тоа се отстрануваат атрибутите кои можеби се редундантни на некои други. Ако атрибутот ги подобрил перформансите кога последниот пат бил отстранет од некое податочно множество, тој веќе не се зема во предвид. Откако сите атрибути ќе се земат во предвид за отстранување, тогаш завршува изминувањето наназад. Ако имало подобрување на перформансите во изминувањето напред или наназад, тогаш целата постапка се повторува. Инаку, пребарувањето завршува. Тоа може да се случи и ако се достигне максималниот број на множества што смее да се евалуираат, за да се ограничи времетраењето на пребарувањето.

Како и кај сите алгоритми кои прават обвиткувачко одбирање, и кај овој алгоритам треба да се определи кој алгоритам за класификација да се користи.

Алгоритмот треба да е осетлив на додавање или отстранување на еден атрибут од множеството, односно оваа промена треба да се рефлектира во перформансите на моделот соодветно ако атрибутот бил важен. Заради ова ограничување, ансамблите не се соодветни алгоритми бидејќи при градењето на моделите постапката е подложна на случајности, поради кои промената на множеството на атрибути може да биде нетранспарентна и занемарена. Од друга страна, наивниот Баесов класификатор и логистичката регресија се соодветни алгоритми, особено поради тоа што нивното тренирање и евалуација се брзи.

Изборот на класификациски алгоритам може да влијае на избраното податочно множество и може да се случи тоа да биде наклонето кон употребениот класификациски алгоритам. Како резултат на тоа, одбраното множество може да не биде глобално најоптимално, односно за други класификациски алгоритми да има други посоодветни множества на атрибути. Сепак, заради едноставноста на класификациските алгоритми што се користат во обвиткувачкиот метод за одбирање на атрибути, многу множества може да се евалуираат одеднаш и оваа постапка да трае многу кратко.

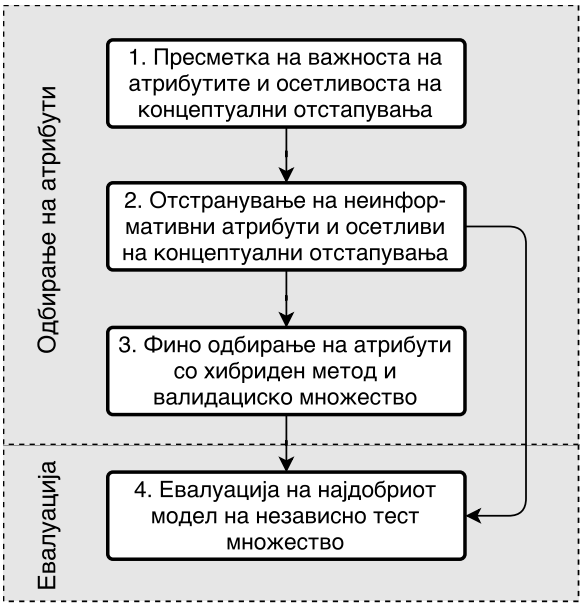
4.3.4 Хибриден метод со грубо филтрирање и алчно пребарување напред-назад

Методот опишан во оваа глава всушност само ги комбинира претходно опишаните хибридни методи во глава 4.3.2 и 4.3.3 како што е прикажано на Слика 4-2, и ваквата комбинација е искористена во [1].

Прв чекор е да се пресмета важноста на атрибутите. Ова може да се направи со било која метрика (пр. информациона добивка, Џини индекс, итн), но во имплементацијата опишана во [1] беше искористена проценката на важноста на атрибутите што ја дава алгоритмот екстремно случајни дрва користејќи 100 дрва. Причината за тоа е што при тренирањето на ваков класификатор, истовремено се гради модел кој може да се евалуира со валидациски или тест множества со што ќе се добие претстава за перформансите (време, точност, прецизност, итн) на целото множество на атрибути, а дополнително се добива и проценка на важноста на атрибутите. За иста намена може да се искористи и алгоритмот случајни шуми, кој кај големи податочни множества со многу атрибути е за нијанса побавен.

Во истиот чекор дополнително се пресметува и осетливоста на атрибутите на концептуални отстапувања, опишана во Глава 4.3.2.

Потоа, во следниот чекор се прави грубо одбирање на атрибутите користејќи го методот на отстранување на атрибутите кои се неинформативни или се многу осетливи на концептуални отстапувања со методот опишан во Глава 4.3.2.



Слика 4-2: Чекори на хибридниот метод за одбирање на атрибути со филтрирање и алчно пребарување

Потоа следува фино одбирање на атрибутите со хибридниот метод опишан во Глава 4.3.3. На крајот на оваа постапка се добива мало податочно множество кое не содржи редундантни атрибути.

Глава 5

Паралелизација

Во текот на последните години податоците кои ги собираат компаниите и разните организации се со значително зголемен волумен, па нивната обработка и анализа стануваат многу покомплексни. Традиционално, примената на алгоритми за предобработка и класификација на податоци не се соочува со предизвици кои потекнуваат од волуменот на податоците, туку тие сметаат дека базите на податоци кои веќе се употребуваат овозможуваат соодветни позадински функционалности како ефикасно запишување и читање на потребните податоци. Од друга страна, во контекст на Големи Податоци, примената на алгоритми за предобработка и машинско учење е условена најпрво од ефикасното запишување и вчитување на потребните податоци, а потоа изнаоѓање на соодветни начини за паралелизација на остатокот од обработката на податоците.

За ефикасна обработка на Големи Податоци може да се применат следниве пристапи: прилагодување на алгоритмите со разни оптимизации, користење на помоќен хардвер или паралелизација на алгоритмите и користење на кластер од компјутери. Првите два пристапи генерално може да дадат задоволителни резултати до одреден момент, но кога податоците доволно ќе се зголемат, тие веќе не се соодветни затоа што веќе не е можна обработка во реално време со задоволителна точност. Токму заради тоа, третиот пристап, кој го дистрибуира пресметувањето и чувањето на податоците на кластер од компјутери, е многу актуелен во денешно време. Иако оваа идеја не е толку нова, сепак од пред десетина години таа почна да се применува во повеќе производни системи.

Во оваа глава најпрво се разгледуваат постоечките архитектури кои ја олеснуваат паралелизацијата на алгоритмите. Понатаму се опишани предложените методи за ефикасно запишување на Големи Податоци. Во продолжение на оваа глава се изложени начините на паралелизација на некои алгоритми за инженерство и одбирање на атрибути.

5.1 Преглед на литературата

Инспирирани од пристапот на Google опишан во [148] во 2004 и [149] во 2005, многу други компании и проекти со отворен код тргнаа во оваа насока развивајќи различни дистрибуирани системи. Еден од најпопуларните такви системи е Apache Hadoop [150, 151], кој содржи алгоритми за дистрибуирана обработка, распоредување на ресурси и чување на големи податочни множества на компјутерски кластери. Листата на активни корисници [152], меѓу кои спаѓаат Yahoo, Facebook, Ebay, Adobe и други добро познати компании, покажува дека Hadoop е веќе зрел и добро воспоставен производ. Всушност, во [153, 154], каде се опишани најновите трендови поврзани со паралелизирана и дистрибуирана обработка на податоци, се споменуваат Hadoop и MapReduce како активни области на истражување и примена. Во [155] е наведен тековниот статус на паралелното пресметување и се дадени следниве заклучоци: и покрај многуте критики и спротивставени мислења паралелното пресметување е огромен успех за што сведочи примената во многу компании; постоечките имплементации на алгоритмите не се директно употребливи за паралелно пресметување; специјализираните хардверски забрзувања тешко остануваат релевантни во однос на генералните платформи и пристапи за паралелно пресметување. Во следните подглави е даден краток преглед на актуелните технологии за обработка на Големи Податоцина:Hadoop, HBase, MapReduce и Spark, како и на некои пристапи за пралелизација на алгоритми со помош на овие технологии.

5.1.1 Hadoop

Hadoop со помош на своите сервиси ги овозможува логистика и набљудување на процеси како распоредување, дистрибуција, комуникација и пренос на податоци, при што обезбедува редундантност на податоците и толеранција на грешки. Од повеќето сервиси кои се присутни во Hadoop, најзначајни се: YARN (MapReduce2), HDFS и HBase [156, 157]. YARN (уште еден навигатор на ресурси, Yet Another Resource Negotiator, анг.) [151] се грижи за распоредување, набљудување и управување со ресурси на MapReduce задачите. HDFS (Hadoop Distributed File System, анг.) [150] е дистрибуиран систем на датотеки кој е скалабилен и толерантен на грешки. Дизајниран е да овозможува скалирање на големи кластери од компјутери. Податоците обично се делат на големи блокови (типично 128 мегабајти), кои независно се реплицирани на повеќе јазли.

5.1.2 HBase: претставник на NoSQL базите на податоци

Иако постојат повеќе полемики и несогласувања во научната заедница и индустријата во однос на предностите и недостатоците на NoSQL базите на податоци во

однос на релационите, подобрата скалабилност на NoSQL базите е неоспорена.

HBase е нерелациона дистрибуирана база на податоци со отворен код која е инспирирана од BigTable на Google и работи на едно ниво над HDFS [158, 159, 160]. HBase е NoSQL (не-само-SQL) база на податоци во која табелите се дизајнираат во согласност со начинот на кој тие ќе се употребуваат. Ова овозможува едноставен дизајн и хоризонтално скалирање со додавање на повеќе јазли во кластерот по потреба. Секој од компјутерските јазли во кластерот има придонес во пресметковните и мемориските капацитети на целиот систем.

Дизајнот и употребата на HBase значително се разликува од системите за управување со релациони бази на податоци поради што и пристапот за имплементација на апликации кои го користат HBase е различен. При креирањето на табели треба да се земат во предвид начините на употреба (како ќе се запишува и чита од неа), очекуваниот волумен на податоци, па дури и дистрибуцијата на вредности на примарните клучеви. Перформансите на системот директно зависат од дизајнот на примарните клучеви на HBase табелите и тој е најважната одлука во правењето архитектура на системите [160, 158]. Причините за тоа се што различните региони во HBase опслужуваат различни опсези на примарни клучеви и се одговорни за редиците во тој опсег при што редиците се чуваат физички во растечки редослед според примарниот клуч. Кај релационите бази на податоци секогаш кога ќе се промени шаблонот на употреба на некоја табела може да се додаде дополнителен индекс или да се отстрани некој непотребен индекс, без притоа да се наруши достапноста на базата. Од друга страна, во HBase постои само индекс според примарниот клуч (аналогно на кластерираните индекси кај релационите бази на податоци), и тоа е единствениот начин да се пристапи директно до некоја редица. Ако треба да се пристапи според некоја друга колона, тогаш треба да се скенираат опсези на редици или дури целата табела. Ова ја потенцира важноста на дизајнот на примарниот клуч кај NoSQL базите како HBase.

Постојат повеќе техники за дизајн на примарни клучеви кои се оптимизирани за различни шаблони на пристап до табелата [160, 158]. Сепак, во литературата недостасува детална анализа на кои дизајни на примарни клучеви може да овозможат балансирано оптоварување кај сите јазли на кластерот, а истовремено овозможувајќи генеричност и употребливост за различни апликации.

Други NoSQL бази на податоци слични на Google Big Table и HBase се DynamoDB на Amazon [161] и Cassandra [162], така што концептите за пралелизација опишани во оваа докторска дисертација се употребливи со било која од нив.

5.1.3 MapReduce

MapReduce парадигмата за програмирање [148, 163] е клучна за дистрибуираната обработка и чување на податоци што го нуди Hadoop. Таа се состои од две фази: мапирање (map, анг.) и редуцирање (reduce, анг.). Првата фаза, мапирање, ги третира проблемите на обработка на податоци како прости проблеми за паралелизација делејќи ги податоците на дисјунктни множества кои може да се обработуваат паралелно и независно едно од друго. Фазата на редуцирање ги агрегира излезите од првата фаза па затоа мора да се изврши по нејзиното целосно завршување. Завршувањето на фазата на редуцирање го дава крајниот резултат. Во текот на фазата на мапирање може да се извршуваат различни операции како: читање, проекција, филтрирање, сортирање и слично. Излезот од оваа фаза е меѓурезултат кој се состои од парови клуч и вредност. Hadoop овозможува дистрибуирање на овие парови така што на исто место (ист јазол во кластерот) ќе се наоѓаат паровите со исти клучеви. Иако MapReduce моделот е рестриктивен, неговата едноставност го прави многу погоден за паралелизација на пресметките дури и на кластери со илјадници јазли.

Исклучителната едноставност на MapReduce моделот често бара програмирање на пониско ниво, дури и за овозможување на операции кои се многу едноставни во други програмски јазици. Со ова се зголемува времето за развој, почесто се прават грешки и може да ја ограничи оптимизацијата [164]. За олеснување на ваквите чести имплементации на слични функционалности е развиен скриптниот јазик Pig Latin. Овој јазик е на високо ниво и го олеснува текот на податоци, при што претставува компромис помеѓу SQL и MapReduce. Pig обезбедува концепти за манипулација на податоците слични на тие во SQL, при што се овозможува нивна интеграција во експлицитни текови на податоците. Pig програмите се компајлираат во секвенци од MapReduce програми кои потоа се извршуваат на Hadoop [165].

5.1.4 Spark

Инспирана од концептот на MapReduce, Spark парадигмата овозможува реискористување на работни множества од податоци во повеќе паралелни операции [166]. Примери за вакви апликации се итеративните алгоритми за машинско учење и алатките за интерактивна анализа на податоци. Концептот на издржливи дистрибуирани податочни множества (Resilient Distributed Datasets - RDD, анг.) е фундаментален за функционирањето на Spark [167]. RDD е основната апстракција во Spark и претставува немутабилна партиционирана колекција од елементи кои може да се обработуваат паралелно на различни јазли во кластерот. За да се овозможи ефикасност при реискористувањето на работните множества на податоци, Spark ги вчитува сите податоци во работната меморија каде се чуваат сè додека се потребни. Благодарение на извршувањето на операции врз податоци

кои се чуваат во работната меморија, некои програми во Spark се извршуваат за еден ред на големина побрзо отколку со помош на MapReduce [166, 167].

MLLIB е библиотека со отворен код за дистрибуирано машинско учење која се темели на Spark [130] и во неа се достапни некои алгоритми, како случајни шуми кои се употребуваат во оваа докторска дисертација.

5.1.5 Паралелизација на алгоритми

Паралелизацијата на алгоритмите воведува можност за повеќе потенцијални софтверски грешки кои често се поврзани со пристап до критични региони, комуникација и синхронизација помеѓу различни подзадачи. Поради тоа, пишувањето паралелни компјутерски програми е поголем предизвик отколку пишувањето секвенцијални програми.

Ефикасната евалуација на атрибутите за логистичка регресија е важна, особено кај големи податочни множества и токму тоа се истражува во [168]. Таму авторите предлагаат нова евристика за одбирање на атрибути нанапред (forward feature selection, анг.) што ги рангира атрибутите по проценетиот придонес што би го имале во перформансите на конечниот модел. Пристапот е тестиран на повеќе податочни множества кои се достапни на [169]. Дополнително, генерирани се нови податочни множества за кои однапред се познати коефициентите за логистичка регресија, со што лесно може да се евалуираат одбраните атрибути.

Во [170] е предложен алгоритам кој е заснован на анализа на варијансата. Со него се одбираат атрибути кои можат да ја објаснат варијансата на податоците. Имплементацијата на овој алгоритам е со помош на MPI (Message Passing Interface, анг.) [171]. Авторите во [172] предлагаат метод за одбирање на атрибути базиран на MapReduce моделот. Во секој јазол за време на фазата на мапирање се пресметува меѓусебната информација на атрибутите и со помош на евристика која го проценува степенот на придонес на секој поединиечен атрибут, се одлучува колку атрибути ќе бидат одбрани. Паралелна и дистрибуирана имплементација на SVM алгоритмот за машинско учење се предлага во [173] и истата се применува за скалабилно филтрирање на нерелавнтни пораки. Со помош на распоредувањето на пресметките на повеќе јазли, значително се намалува времето за тренирање на алгоритмот без притоа да има значителна деградација на предиктивните перформанси.

Еден поинаков пристап за обработка на големи податочни множества е предложен во [174]. Овде авторите со помош на паралелизирани и дистрибуирани пресметки прво се обидуваат да го намалат големото множество на мало репрезентативно подмножество кое понатаму ќе може да се обработува со стандардни (непаралелизирани) алгоритми.

Во [175] се предлага пристап заснован на MapReduce за дистрибуирано определување на подмножества на колони. Овде паралелизацијата се прави на тој начин што различни јазли имаат пристап до различни случајни подмножества на атрибути.

Еден обвиткувачки метод за паралелно одбирање на атрибути кој се одвива во две фази е предложен во [176]. Како останатите обвиткувачки пристапи, така и овде одбирањето се одвива во повеќе итерации. Во првата фаза, во секоја итерација се додаваат атрибути така што нивното додавање не ги влошува перформансите на моделот. Ова подразбира дека во секоја итерација моделот се евалуира со некој алгоритам за машинско учење. Потоа, во втората фаза, од множеството на одбрани атрибути се исфрлаат оние чие отфрлање не ги влошува перформансите на моделот. Исто така и во оваа фаза во секој чекор се евалуира моделот со алгоритам за машинско учење.

Во [177] е предложена паралелна имплементација на наивен Баесов класификатор со помош на MapReduce.

5.2 Дизајн на примарни клучеви на табели кај NoSQL бази на податоци

Во контекст на трајно чување на податоците во структуриран или полуструктуриран облик, HBase има повеќе предности. Имено, при запишувањето има поголеми можности кои податоци каде ќе се запишат, а со соодветен дизајн на табелите може да се чуваат повеќе верзии на податоците, за различни фамилии на колони од табалите да се дефинираат различни полиси на чување, репликација, големина на блоковски итн. Но, од аспект на паралелизација на алгоритми за предобработка и машинско учење, HBase има уште една предност, а тое е што нивото на паралелизам може полесно да се контролира на повисоко ниво.

Како што беше споменато во Глава 5.1.2, главниот предизвик при дизајнирањето на табелите во NoSQL базите на податоци е изборот на примарен клуч. Тој треба да овозможува брзо дистрибуирано запишување на податоците кое е рамномерно распоредено на сите јазли на кластерот. Монотоно растечките примарни клучеви (на пример кога се користи временски печат) предизвикуваат само еден од јазлите (тој што е задолжен за хронолошки последните редици) да мора да се справи со запишувањето, додека останатите јазли се неискористени. Во [157, 160] се препорачува избегнување на вакви примарни клучеви со помош на додавање на некој случаен или псевдо-случаен префикс во примарниот клуч. Во оваа глава се разгледуваат ограничувањата, предностите и недостатоците на некои дизајни предложени таму, но и се предлага нов дизајн кој овозможува надминување на некои од недостатоците кои ги имаат овие клучеви.

Со цел да се добијат примарни клучеви што ќе може лексикографски да се подредуваат, се прави дополнување однапред на броевите со нули, така што тие секогаш ќе имаат иста текстуална должина. Ова е важно бидејќи овозможува предвидливо подредување на редиците, индиректно помага во партиционирањето на табелите и овозможува откривање на региони од табелите кои се многу пооптоварени од другите во подолг временски период.

5.2.1 MD5 функција за хеширање

Алгоритамот MD5 (message digest, анг.), предложен во [178], на влез добива порака со произволна должина и на излез дава 128-bit "отпечаток" на влезот. Се смета дека е пресметковно невозможно да се генерираат две пораки кои имаат ист отпечаток или да се генерира порака што ќе има однапред зададен отпечаток. Иако MD5 алгоритамот иницијално е наменет дигитално потпишување, тој може да се употребува и за други намени. Една од нив е генерирање на хеш вредност од произволен аргумент. За дизајнирање на примарни клучеви на големи табели, предлагаме MD5 хеш вредностите да се користат како псевдо-случајни генератори. Бидејќи опсегот на вредности кои се генерираат со MD5 е многу голем, не е практично да се употребува директно за оваа намена, па затоа се пресметува модул од MD5 хеш вредноста, со што се добиваат вредности во предефиниран опсег. Деталната постапка како се прави ова е опишана во [11], каде е направена и детална анализа на дистрибуцијата на вредности добиени на овој начин.

Во [179] се предлага користење на MD5 хеш вредности за генерирање на примарни клчеви, но предложениот дизајн е специфичен за конкретен домен и недостасува анализа на дистрибуцијата на примарните клучеви и времетраењето на извршувањето споредено со други дизајни на примарни клучеви.

5.2.2 Предвремено партиционирање на HBase табели

При креирањето на HBase табелите може да се предефинираат точки на поделба (split points, анг.) со кои табелите се партиционираат предвеременo, така што секој регион (т.е. партиција) е задолжен за опслужување за одреден интервал на вредности на примарни клучеви.

Ако не се предефинираат точките на поделба, тогаш табелата е претставена само со еден регион. Кога овој регион ќе нарасне доволно, тогаш табелата автоматски се партиционира на два региона така што во секој од нив ќе припаднат приближно еднаков број на редици во моментот на поделбата. Ова е скапа операција која има неколку недостатоци. Прво, автоматски одбраната точка на поделба можеби е оптимална во моментот на поделба, но тоа не е гаранција дека ќе остане оптимална после некое време, со оглед на тоа што растот на податоците

и очекуваната дистрибуција на примарните клучеви не е земена во предвид при поделбата. Второ, после поделбата двата региони се опслужуваат од ист сервер на региони (Region Server) и тие треба рачно да се прераспределат на друг кој можеби е помалку оптоварен. Трето, моментот на поделба не може да се предвиди и тој може да се случи во време кога некои критични операции или апликации ја користат табелата.

Од друга страна, HBase овозможува предвремено партиционирање на табелите преку дефинирање на точки на поделба при креирањето на табелата. Ова е корисен начин за да се осигура дека табелата има соодветен број на региони кои се рамномерно распоредени на серверите на региони уште кога табелата е празна. Со предвременото партиционирање се гарантира дека иницијалниот товар е рамномерно распореден низ кластерот и овој метод секогаш треба да се применува доколку може да се претпостави дистрибуцијата на клучевите. Но, ако оваа претпоставка не е задоволена, тогаш постои ризик од креирање на региони кои многу малку ќе се користат поради нерамномерна дистрибуција. Ако изборот на точки на поделба е лош, тогаш различните региони ќе имаат различно оптоварување.

Ваквиот проблем ја потенцира важноста на предвидлива дистрибуција на примарните клучеви, што е главен фокус во [11]. Таму се дадени детални препораки како да се овозможи предвидлива дистрибуција за да се направи предвремено партиционирање и како да се одбере бројот на региони во однос на бројот на јазли во кластерот. За да се овозможи ова, се предлага користење на префикси во клучот кои овозможуваат рамонмерна дистрибуција. Во следниве подглави се опишани накратко типовите на префикси кои во детали се анализирани во [11].

Случаен префикс

Овој префикс претставува случаен цел број во даден интервал. Тоа што вредноста е ограничена овозможува поделба на интервалот на посакуваниот број на подинтервали, кои ќе одговараат на соодветни партиции на табелата. Ваквиот дизајн дава добри перформанси на запишување заради рамномерната распределба на случајните префикси. Од друга страна има исклучително лоши перформанси на случаен пристап при читање, бидејќи кога треба да се вчита некоја редица, случајниот префикс не е познат па мора да се направи скенирање на целата табела.

Префикс со модул од клучот

Овој дизајн претставува модул од единствениот клуч на редицата кој се додава на примарниот клуч како префикс. Очигледно овој дизајн е соодветен само

за редици чиј клуч е цел број, а дополнителните недостатоци беа воочени при експерименталната евалуација и се опишани во Глава 6.9.1.

Префикс со модул од MD5 хеш вредноста од клучот

Овој дизајн го надминува главниот недостаток на претходниот така што прво пресметува MD5 хеш вредност од клучот на редицата, и е применлив за клучеви од произволен тип, дури и за сложени клучеви составени од повеќе колони. Потоа пресметувајќи модул од добиената хеш вредност се добива префикс во даден опсег. Се покажува дека овој дизајн дава најдобри перормanси во однос на рамномерна распределба на товарот при запишување, но и овозможувајќи случаен пристап до конкретни редици.

Префикс со модул од MD5 хеш вредноста од целиот ред

Овој дизајн е сличен на претходниот кој пресметува хеш вредност од целиот ред, наместо само од примарниот клуч. Иако ова има ограничена реална примена, сепак е анализиран со цел да се направи евалуација на временските перформанси на MD5 функцијата за големи аргументи, како и тоа дали случајната распределба сеуште важи. Експериментите опишани понатаму даваат позитивен одговор на овие две парашања.

5.2.3 Читање од HBase табели

Откако некое податочно множество е зачувано во HBase табела, може да се продолжи со имплементација на алгоритми за предобработка и машинско учење. Со оглед на поделеноста на табелите на региони, при вчитувањето на податоци, автоматски се искористува принципот на локалност на податоците, па секој јазол иницијално ги вчитува и обработува податоците кои се снимени на него. Бројот на региони на HBase табелите го дефинира иницијалното ниво на паралелизам. Тоа понатаму може да се менува, но претставува скапа операција, бидејќи предизвикува трансфер на податоци помеѓу јазлите.

5.3 Паралелно инженерство на атрибути од временски серии

Инженерството на атрибути генерално е поедноставно за паралелизација отколку паралелизацијата на алгоритмите за машинско учење и одбирање на атрибути.

Причина за тоа е фактот што најчесто генерирањето на атрибути зависи само од оригиналните вредности на инстанците, но не и од вредностите на другите инстанци во податочното множество.

Кај инженерството на атрибути од временски серии настанува мало усложнување поради тоа што податоците иницијално се генерираат во оригинален облик, без да се сегментираат во подвижни прозорци. Поради тоа, потребна е соодветна архитектура и избор на технологии соодветни за Големи Податоци кои овозможуваат ефикасна, скалабилна и издржлива сегментација.

5.3.1 Генерирање на атрибути со податочен паралелизам

Ако податоците генерирани во еден временски прозорец се достапни на едно место, тогаш генерирањето на атрибути од секој временски прозорец може да се извршува паралелно. Овој начин е скалабилен бидејќи со хоризонтална фрагментација на податочното множество, различни јазли од кластерот може да обработуваат различни дисјунктни подмножества. Потребно е дефинирање на функција која од еден ред во податочното множество даден на влез, на излез ќе ги враќа сите генерирани атрибути. На овој начин, алгоритамот за инженерство на атрибути прикажан на Слика 3-2 во Глава 3.1, сите чекори од 2 до 10 ќе ги извршува во една функција, за секој ред од податочното множество посебно. Заради простиот паралелизам, оваа функција може да се повика многу пати паралелно на различни јазли обработувајќи различни редови истовремено. Единствен услов за да се направи ова е претходна сегментација на податоците во временски прозорци, за што се дискутира во продолжение.

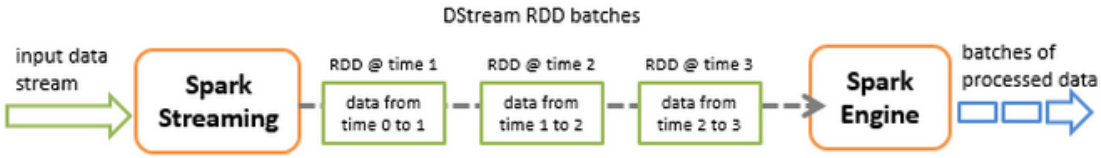
5.3.2 Сегментација на временски серии со Spark Streaming

Во последно време, еден начин на собирање на Големи Податоци што се генерираат со висока брзина е со користење на платформи за податочни потоци кои се дистрибуирани, скалабилни, реплицирани и отпорни на грешки, како што се Apache Kafka, Amazon Kinesis и други [180]. Со дефинирање на полиси на задржување (retention policy) може да се дефинира колку долго податоците ќе се задржат на податочниот тек по нивното генеирање. Податоците од текот може да се конзумираат со буткачки (push, анг.) или повлекувачки (pull, анг.) механизми [181, 36].

Еден од типовите на потрошувачи (consumers, анг.) на податочните потоци е технологијата Spark Streaming [1], која овозможува повлекување на податоците од податочните потоци на одредени временски интервали и преземање на податоци

[1] `http://spark.apache.org/streaming/`, посетено на 03.03.2017

Слика 5-1: Тек на податоци кај Spark Streaming



Слика 5-2: Сегментација на податочните потоци во временски прозорци со Spark Streaming

генерирани во определен временски интервал. Притоа е овозможено дефинирање на должината на временските прозорци, со што ќе се земат податоци кои биле генерирани во одреден интервал, и дефинирање на фреквенцијата кога се прави повлекувањето, што овозможува прилагодување на преклопувањето на соседните временски прозорци, концепти за кои се дискутираше во детали во Глава 3.1.1. На Слика 5-1 е прикажан генералниот тек на земањето податоци со помош на Spark Streaming и најчестите извори на податоци. Притоа, податоците може да бидат генерирани било каде (пр. софтверски дневници, сензори, кориснички уреди и друго) и по нивното запишување на платформите за податочни потоци, се прави едноставна податочна фузија.

По временската сегментација на веќе фузираните податоци со Spark Streaming, прикажана на Слика 5-2, податоците генерирани во одреден временски интервал од било кој извор се достапни како единствена целина над која може да се прават различни операции. После Spark Engine излегува множество на инстанци со податоците кои припаѓаат на еден временски прозорец (batch of processed data, анг.).

Со ова завршува фазата на сегментација и сегментираните податоци може да се чуваат во NoSQL база на податоци, HDFS, HBase, Amazon S3 итн. Ова овозможува извршување на алгоритми за инженерство на атрибути од сегментираните податоци подоцна во пасивен (offline) режим на работа со помош на програми имплементирани во MapReduce или класичен Spark.

Но, Spark Streaming може да се употреби за многу повеќе од само сегментација. Имено, после сегментирањето Spark Streaming може да продолжи со извр-

шување на функциите за инженерство на атрибути. На овој начин, освен сегментираните податоци без генерирани атрибути, може да има и втор излез од Spark Streaming, а тоа е множество со процесирани податоци кое ќе се зачува посебно од првиот излез. Така во Spark Streaming програмите може истовремено да се прави и сегментација и инженерство на атрибути, при што ќе се генерираат два излези.

Во системи пуштени во производство, со Spark Streaming може да се генерираат сите потребни атрибути или само одбрано множество на атрибути во реално време кои понатаму ќе искористат за класификација во реално време.

5.4 Инженерство на атрибути со агрегација на Големи Податоци

5.4.1 Преглед на секвенцијални пристапи во литературата

Инженерството кое вклучува агрегација на податоците, генерално е проследено со рачни процеси кои вклучуваат експертско знаење и бараат разбирање на бизнис правилата и деловните постапки кои водат до генерирање на атрибуите. Сепак постојат пристапи во литературата [182, 183, 184, 185, 186, 187], дури и производни системи [188], кои меѓу другото извршуваат генерални постапки кои се темелат на функции за агрегација за генерирање на атрибути. Ваквите пристапи овозможуваат автоматско генерирање на атрибути независно од доменот на податоците. Кај овие пристапи, генерално од различни колони на табелата се пресметуваат повеќе агрегациски функции, со што се добиваат многу потенцијално инфромативни атрибути.

За да се овозможи примена и автоматизирање на ваквите пристапи, агрегацијата на Големи Податоци е неопходна, поради што во оваа глава е анализрано едно често сценарио за екстракција, трансформирање и вчитување на податоци (Extract-Transform-Load, ETL анг.) кое воедно ги пресметува и потребните агрегации. ETL процесот вклучува и чекори кои се надвор од фокусот на оваа докторска дисертација, па затоа таквите чекори се споменати во предложената паралелна имплементација без да се објаснуваат во детали.

5.4.2 Паралелна агрегација на Големи Податоци во Spark

Постапката на екстрација, трансформирање и вчитување на податоци (Extract-Transform-Load, ETL анг.) е неопходна во градењето на податочни складишта. На Слика 5-3 се дадени чекорите на една варијанта на имплементирање на ETL

Слика 5-3: ETL и генерирање на атрибути со агрегација на Големи Податоци во Spark

и генерирање на атрибути со агрегација на Големи Податоци во Spark. Дел од чекорите кои вклучуваат генерирање на примарни и надворешни сурогат клучеви се неопходни кај ETL постапките во деловните процеси и градењето на податочни складишта, но не се од интерес за инженеството на атрибути. Текстот што е во загради и е искосен на Слика 5-3 ги претставува Spark наредбите кои се користат при имплементацијата на соодветниот чекор.

Чекор 1, ги вчитува и дистрибуира бизнис и сурогат клучевите на сите јазли на Spark кластерот со операцијата Broadcast во Spark. Традиционалниот начин за пополнување на надворешни клучеви во табелите на факти користи операции на спојувања (joins) и тој е неефикасен за Големи Податоци. Но, со однапред распоредување на мапирањето помеѓу бизнис и сурогат клучевите за секоја димензиона табела на секој јазол во структура на речник, оваа операција се сведува на едноставно мапирање од една вредност во друга. Со тоа е не е потребен дополнителен трансфер на податоците низ кластерот (освен иницијалниот broadcast) и го користи принципот на локалитет на податоците.

Во чекор 2 се вчитуваат сите нови податоци од Amazon S3. Може да се користи и било кој друг извор на податоци се додека тој е дистрибуиран и толерантен на грешки. На овој начин се овозможува различни јазли од кластерот да обработуваат различни податоци.

Чекор 3 генерира единствени монотоно растечки сурогат клучеви со помош на ZipWithUniqueId наредбата, која не предизвикува трансфер на податоците и го користи принципот на локалитет на податоците.

Во чекор 4 се прави трансформација на податоците, нивно прочистување, претварање во други типови, користење на нумерички трансформации и трансформации од номинални во нумерички типови, поставување на надворешни сурогат клучеви до димензиите, итн. Овој чекор ја користи наредбата MapPartitions во Spark со што се гарантира локално извршување на податоците, без да се прави трансфер низ мрежата.

Потоа во чекор 5 се прави групирање на податоците во иста група. Групирањето може да се прави по било која колона кој е од интерес. Кај деловни податоци ова може да биде по корисник, тип на услуга, временски период, итн.

Во чекор 6 податоците во секоја група се агрегираат, при што од различни колони се генерираат по повеќе атрибути со помош на агрегациските функции (пр. count, sum, min, max, или било која кориснички дефинирана функција за агрегација). При агрегацијата дополнително се задржуваат и неагрегираните податоци за да може да се зачуваат и во таков облик за понатамошна употреба.

Потоа во чекор 7 од агрегираните записи се прави множество на редици со наредбите Map или FlatMap, се генерираат примарни клучеви за секој ред на ист начин како и во чекор 3 и генерираните примарни клучеви им се доделуваат на

агрегираните редици. Дополнително, на неагрегираните редици се поставуваат надворешни клучеви кои покажуваат кон агрегираната редица која е генерирани од нив. Овој чекор ја користи и операцијата Persist со која се овозможува реискористување на резултатот од овој чекор повеќе пати, без да има има потреба од негово повторно пресметување.

Чекор 8 ги зачувува агрегираните редици на HDFS или било која соодветна база на податоци како HBase. Притоа секоја агрегирана редица меѓу другото, ги содржи и генрираните атрибути потребни за класификација и машинско учење. Дополнително, чекор 9 ги екстрахира неагрегираните редици и потоа во чекор 10 ги запишува на HDFS или во било која соодветна база на податоци.

Евалуациајта на предложената постапка со Големо Податочно множество е опишана во Глава 6.8.

5.5 Паралелно пресметување на информационата добивка

Во Hadoop повеќето од предизвиците поврзани со комуникација и синхронизација на процесите се веќе решени со помош на различни механизми и сервиси, така што кога алгоритмите се имплементираат за извршување на Hadoop (пр. во MapReduce, Pig Latin или Spark), програмерот не треба да троши голем труд за решавање на ваквите предизвици.

Паралелизацијата на пресметувањето на информационата добивка е направена според рамката предложена во [17]. На Слика 5-4 е прикажан текот на податоците при паралелното пресметување на информационата добивка [16, 14]. На почетокот, податоците се наоѓаат на HDFS. Првиот чекор ги вчитува податоците од HDFS во HBase табели кои се однапред партиционирани за сите јазли да бидат рамномерно оптоварени. Овој чекор е неопходен кога се програмира со MapReduce или Pig Latin. Кога се користи Spark, запишувањето на податоците во HBase само заради рамномерна распределба на оптоварувањето и заради контрола на паралелизмот не е потребно, бидејќи во него постојат поекспресивни наредби кои овозможуваат поголема контрола. Потоа, процесирањето на податочното множество за пресметка на информационата добивка се одвива во три фази, објаснети во следните подглави.

5.5.1 Пресметка на ентропија

Како што беше објаснето во Глава 4.1.2, дефиницијата на информационата добивка ја вклучува и ентропијата на целото податочно множество. Најпрво треба

Слика 5-4: Тек на податоците при паралелното пресметување на информационата добивка

да се пресмета бројот на инстанци по класа и потоа да се пресмета сумата на веројатности на класата. Ова е едноставен чекор и во општ случај не мора да се паралелизира, со оглед на тоа што неговата комплексност е $O(N)$, каде N е бројот на инстанци во множеството. Дополнително, ако само треба да се направи рангирање на атрибутите без да мора да се пресмета точната информациона добивка тогаш ентропијата може да се изостави од равенката (4.3). Поради тие причини овде не е опишана паралелизацијата на пресметката на ентропијата, иако тоа е направено во [14, 16].

5.5.2 Број на инстанци по атрибут, вредност на атрибут и класа

По пресметката на ентропијата, потребно е да се пресмета бројот на инстанци по атрибут, вредност на атрибутот и класата, како што може да се забележи од (4.3). Овој чекор во алгоритамот е пресметковно најскап. Скриптата за овој чекор во Pig Latin е претставена во Изворен код 5.1. На скриптата и се предаваат името на изворната табела, бројот на атрибути, индексот на класата, бројот на цифри (padding digits, анг.) итн. Најпрво се вчитува податочното множество од HBase табелата (линии 3 до 5). Секој ред во множеството е претставен со примарниот клуч и речникот r во кој клучеви се имињата на вчитаните колони на табелата, а вредности на речникот се вредностите на вчитаните колони за тековниот ред. Оваа репрезентација овозможува чување само на ненултите вредности во множеството. Потоа секој ред во множеството, означен со речникот r, треба да се разложи на торки од облик: *(индекс на атрибутот, вредност на атрибутот, класа, 1)*, односно *(feature index, feature value, class, 1)* во Изворен код 5.1. Ова се прави во линија 7 со кориснички-дефинираната функција *decode_sparse_row*. Ако множеството има M редици (инстанци) и N колони (атрибути), тогаш од секој ред ќе се генерираат N торки, вклучувајќи ги и нултите елементи (ќелии во табелата). Така, вкупно $M \times N$ торки се генерираат со овој чекор. Потоа овие торки се групираат по клучот *(индекс на атрибут, вредност на атрибут, класа)*, односно *(feature index, feature value, class)* во линија 9 до 11, и на крај бројот се запишува во друга табела (линија од 12 до 13). Изворниот код 5.1 се компајли-

ра во една MapReduce задача. Кога би се програмирала оваа постапка во Spark, повторно ќе се употреби истиот тек на податоците користејќи аналогни наредби (за секоја од употребените наредби во Pig Latin има соодветна во Spark). Бројот на мапирачки задачи ќе биде еднаков на бројот на региони на влезната табела, означена со *$table_dataset* во скриптата, додека бројот на редуцирачки задачи се поставува со параметарот *$parallel*, за кој постои слична функција и во Spark. Во Spark бројот на мапирачки задачи може да се менува програмабилно, дури и кога податоците не би биле зачувани во HBase.

```
1  register '$udf_path' using jython as UDFs;
2  set default_parallel $parallel;
3  pfdata_tmp = LOAD '$table_dataset' USING org.apache.pig.backend.hadoop.hbase.
4      HBaseStorage('r:*', '-loadKey=true') AS
5      (rowkey:tuple(prefix_padded:chararray, id_padded:chararray), r:map[]);
6  pfdata_short = FOREACH pfdata_tmp GENERATE
7      FLATTEN(UDFs.decode_sparse_row(r, $num_features, $num_features_digits,
8      '$feature_data_type', '$label'));
9  feature_value_class_counts_group = GROUP pfdata_short BY (feature_index, feature_value, class);
10 feature_value_class_counts = FOREACH feature_value_class_counts_group GENERATE
11     group as rowkey, SUM(pfdata_short.instance_count) as instance_count;
12 STORE feature_value_class_counts INTO '$table_feature_index_tmp'
13     USING org.apache.pig.backend.hadoop.hbase.HBaseStorage('r:instance_count');
```

Изворен код 5.1: Броење на инстанци по атрибут, вредност на атрибутот и класа со Pig Latin

5.5.3 Пресметување на информационата добивка

По завршувањето на претходниот чекор опишан во подглава 5.5.2, останува да се пресметаат веројатностите и ентропиите во (4.3), и потоа да се зачува резултатот. Овој чекор е втор по времетраење во однос на претходните и имплементацијата во Pig Latin за него е дадена во Изворниот код 5.2. Прво во линија 3 до 5 се вчитуваат торките *(индекс на атрибут, вредност на атрибут, класа, број на инстанци)*, односно *(feature index, feature value, class, instance_count)*, кои беа пресметани во претходниот чекор и сега се чуваат во табелата *$table_feature_index_tmp*. Оваа табела е соодветно партиционирана однапред уште пред почетокот на извршувањето на паралелизацијата за да се обезбеди рамномерно оптоварување на јазлите и при запишувањето (во претходиот чекор) и при вчитувањето во овој чекор.

```
1  register '$udf_path' using jython as UDFs;
2  set default_parallel $parallel;
3  feature_value_class_counts_tmp = LOAD '$table_feature_index_tmp' USING
4      org.apache.pig.backend.hadoop.hbase.HBaseStorage('r:instanceCount', '-loadKey=true') AS
5      (id:tuple(feature_index:chararray, feature_value:int, class:int), instanceCount:double);
6
7  feature_value_class_counts = FOREACH feature_value_class_counts_tmp GENERATE
8      flatten(id) as (feature_index, feature_value, class), instanceCount;
9  feature_index_group = GROUP feature_value_class_counts BY (feature_index);
10 feature_index_info_gain = FOREACH feature_index_group GENERATE
11     flatten(group) as feature_index_padded, flatten(UDFs.
12     calc_feature_info_gain(($entropy), group, feature_value_class_counts, ($num_instances)))
13     as info_gain:double;
14 STORE feature_index_info_gain INTO '$table_feature_index_info_gain'
15     USING org.apache.pig.backend.hadoop.hbase.HBaseStorage('r:ig');
```

Изворен код 5.2: Пресметување на информационата добивка со Pig Latin

Најкарактеристичниот дел од скриптата изложена во Изворен код 5.2 е во линијата 9. Овде сите торки се групираат по индекс на атрибутот. Кога Pig скриптата се компајлира во MapReduce задача, за време на мапирачката фаза индексот на атрибутот се емитира како клуч, а за време на редуцирачката фаза сите торки кои го имаат истиот клуч (во овој случај индексот на атрибутот) се групираат заедно на истиот јазол од кластерот. Идентична би била постапката што се одвива во позадина, кога ова би било имплементирано во Spark. Потоа во кориснички дефинираната функција *calc_feature_info_gain* на едно место го има бројот на инстанци за секоја вредност и класа по атрибут, па пресметката на информационата добивка со (4.3) е едноставна. На крај, резултатите се снимаат во HBase табелата $\$table\_feature\_index\_info\_gain$, за да се олесни нивното рангирање и евентуално одбирање на најдобрите од нив.

5.6 Паралелно пребарување на просторот кај обвиткувачки методи за одбирање на атрибути

Обвиткувачките методи за одбирање на атрибутите беа опишани во Глава 4.2. Откако ќе се дефинираат трите најважни работи: метрика за евалуација на множествата на атрибути, кој алгоритам за учење ќе се користи и стратегијата за пребарување на просторот со множества на атрибути, може се премине кон нивна паралелизација.

5.6.1 Претпоставки и ограничувања

Предложената паралелизација ги прави следниве претпоставки, па тие треба да се земат во предвид пред да се одлучи за паралелизација на некој обвиткувачки алгоритам за одбирање на атрибути:

- *Секој јазол може секвенциски да ги обработи сите редици од податочното множество и одредено подмножество на атрибути.* Ова е важно бидејќи секој јазол ќе извршува алгоритам за класификација со сите редици и некое подмножество на атрибути, а по извршувањето ќе ги измери перформансите на множеството на атрибути (време на извршување, предиктивни перформанси, број на атрибути).
- *Обвиткувачкиот метод може да дефинира листа на подмножества од атрибути кои треба да се евалуираат.* Ова е важно бидејќи тогаш нема потреба од синхронизација помеѓу паралелните процеси бидејќи секој од

нив може да работи независно евалуирајќи различни подмножества на атрибути независно од другите. Ваквото барање ја ограничува стратегијата на пребарување на просторот на множества на атрибути.

Воведувањето на овие претпоставки овозможува поедноставување на процесот на паралелизација. Заради едноставноста на пристапот за паралелизација, тој е применлив и на повеќејадрени машини и кај компјутерски кластери од повеќе машини. Воведувањето на овие претпоставки овозможуваат при користењето кластери од компјутери да се минимизира цената за комуникација и синхронизација, кои се воедно и поскапи и посложени во однос на истите процеси на една машина. Компјутерските кластери претставуваат хоризонтално скалирање бидејќи со додавањето на нови машини се зголемуваат пресметковните капацитети на системот.

Кога се користи една повеќејадрена машина, повеќе процеси или нитки може да се извршуваат истовремено. Со ова се овозможува вертикално скалирање, кое се овозможува со додавање повеќе процесори или меморија на една машина.

5.6.2 Чекори на паралелизација

Откако ќе се утврди дека податочното множество и алгоритамот за обвиткувачко одбирање на атрибути ги задоволува претходните претпоставки, тогаш може да се продолжи со нивна паралелизација.

Чекорите од кои се состои методот за паралелизација се:

1. Генерирај листа на множества на атрибути што треба да се евалуираат.
2. Распредели ја и дистрибуирај ја листата на јазлите од кластерот (т.е. процесите), така што секоја подзадача се состои од евалуација на едно подмножество на атрибути.
3. Секој процес ги извршува доделените подзадачи паралелно и независно од останатите процеси.
4. Чекај да завршат сите паралелни процеси.
5. Провери дали се задоволени терминалните услови?
6. Ако не се, врати се на Чекор 1.

За илустрација на методот за паралелизација во продолжение се опишани секој од чекорите применети на хибридните методи кои содржат обвиткувачко одбирање на атрибути опишани во Глава 4.3.2 и Глава 4.3.3, што е опишано и во [1].

Чекор 1 зависи од стратегијата за пребарување на обвиткувачкиот метод. На пример, кај методот за откривање на атрибути кои се осетливи на концепту-

ални отстапувања, опишан во Глава 4.3.2, множествата на атрибути кои треба да се евалуираат се добиваат со помош на рангирање на атрибутите по два критериуми. За рангирањето по двата критериуми може да се употреби паралелен алгоритам, како тој предложен во Глава 5.5. Потоа, користејќи различни прагови се дефинираат повеќе подмножества на атрибути, што одговара токму на овој чекор.

Слично, кај генетските алгоритми чекор 1 е дефиниран од единките во генерираната популација, каде една единка претставува едно множество на атрибути. Кај алчните методи за пребарување, како тој што е предложен во Глава 4.3.3, во чекор 1 се генерираат подмножества кои се поветувачки. На пример, тоа може да биде неколку најдобри подмножества на атрибути добиени до тој момент на кои се додадени еден или повеќе атрибути (пр. тие кои се индивидуално поинформативни).

Понатаму, во Чекор 2 оваа листа на подмножества се дистрибуира на паралелните процеси кои треба да ги евалуираат. Ако се користи Spark, дистрибуцијата ја прави програмата двигател (driver, анг.), така што секој од јазлите ќе добие по приближно ист број на множества на атрибути кои треба да се евалуираат. При користењето нитки и процеси за паралелизација на една машина, ова се прави во главната нитка (или процес), но логиката останува идентична.

Потоа, во Чекор 3, секој јазол го вчитува податочното множество со само тие атрибути кои се потребни (или целото множество од еднаш ако тоа е можно), го извршува секвенциски алгоритамот за класификација, и ги зачувува перформансите за соодветното множество на атрибути. Постапката се повторува додека секој јазол не ги евалуира сите множества на атрибути кои му се доделени.

Откако сите паралелни евалуации ќе завршат, програмата двигател ги агрегира перформансите на множествата од атрибути и одлучува дали алгоритамот ќе продолжи. Терминалноста може да биде резултат на надминување на бројот множества на атрибути кои е дозволено да се евалуираат или ако нема подобрување на предиктивните перформанси.

Ако извршувањето се врати на Чекор 1, тогаш следната листа на множества на атрибути ги зема во предвид перформансите од претходниот циклус.

Покрај овие предности на пристапот кои овозможуваат лесна паралелизација, постои еден недостаток, а тоа е што може да се случи еден некој процес или јазол да ја заврши работата многу побавно од другите. Поради тоа, другите ќе го чекаат да заврши пред да добијат нови задачи. Кај повеќејадрените системи, каде комуникацијата и синхронизацијата помеѓу нитките е поедноставна, може да се искористат механизми со семафори, така што процесите ќе добиваат нова задача во моментот кога ќе ја завршат тековната. Ова е во спротивност со пристатпот опишан во оваа глава, каде задачите се споделуваат однапред. Сепак, ваквиот пристап би овозможил само вертикална скалабилност, односно

користење само на една машина, што во општ случај не е доволно.

5.7 Краток осврт на паралелното извршување на класификациските алгоритми

Повеќе алгоритми за машинско учење веќе се паралелизирани со помош на MapReduce и Spark и се достапни во библиотеките Mahout [129] и MLLib [130], опишана подетално во [189]. Бидејќи главниот придонес на оваа докторска дисертација е во инженерството и одбирањето на атрибути, не беа предложени паралелни имплементации на алгоритмите за машинско учење. Сепак методот предложен за пресметување на информационата добивка може да се искористи итеративно за градење на дрва за одлучување, дискретизација на атрибути и градење на наивен Баесов класификатор.

Откако ќе се истренира класификационен модел, без разлика дали тоа е со помош на секвенциска или паралелна програма, треба да се утврди како ќе се употребува во производна околина. Генерално, кај дрвата за одлучување и ансамблите кои се темелат на дрва за одлучување, моделите се состојат од множество на правила. Кај наивниот Баесов класификатор за континуални променливи или логистичката регресија, моделите се уште поедноставни така што за секој атрибут во податочното множество има неколку параметри со кои се дефинира моделот. Дури и кај Големи Податоци, моделите на овие класификациски алгоритми се доволно мали за да може да се дистрибуираат на секој јазол од кластерот. Ова овозможува нивно извршување паралелно за секоја инстанца која треба да се класифицира. Како што функциите за генерирање атрибути може да се извршуваат паралелно, дури и во реално време со помош на Spark Streaming, така и истренираните класификациски модели може во реално време да класифицираат нови инстанци. Една технологија која го овозможува ова е Spark Streaming, како што беше опишано во Глава 5.3.2. Сепак, кај други класификациски алгоритми, како кај машините со носечки вектори и k-најблиските соседи, моделите не се толку портабилни и нивната дистрибуција на секој јазол од кластерот може да не изводливо. Некои пристапи за надминување на овие проблеми се споменати во Глава 5.1.5.

Глава 6

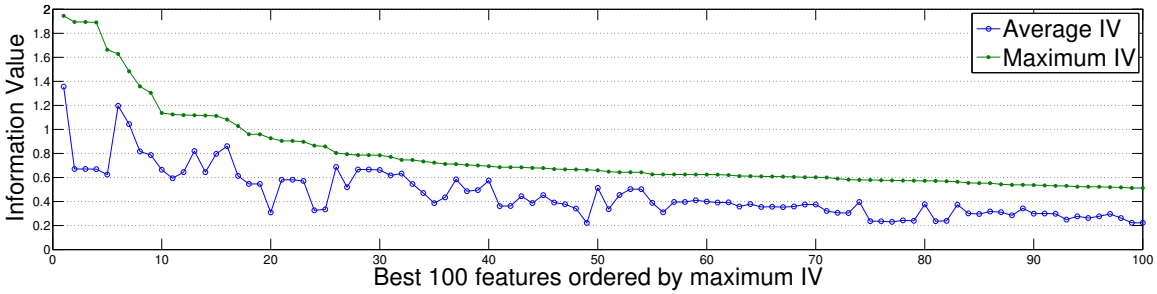
Студии на случај

6.1 Податочни множества со мешани типови на атрибути

Методот за трансформација на номинални во нумерички проблеми предложен во [21] и проширен на повеќекласни проблеми во [13] беше применет за неколку намени. Во [21] се претставени резултатите од обработката на податочно множество за оценување на кредитоспособноста на кредитобарателите. Со оглед на тоа што ова множество изобилува со номинални, нивната трансформација во нумерички беше неопходна за успешно градење на предиктивни модели. Слично, во [13] се обработени неколку податочни множества со мешани типови на атрибути со помош на оваа трансформација и е направена споредба со техниката за трансформација на номинални атрибути преку генерирање на прости атрибути. Еваулацијата покажа дека кога бројот на номинални атрибути и нивната кардиналност (вкупниот број на различни вредности што може да ги имаат) се мали, тогаш двете трансформации даваат слични резултати. Во такви случаи заради едноставноста се препорачува трансформацијата со генерирање на прости атрибути. Но, во случај кога податочното множество изобилува со номинални атрибути со голема кардиналност, алгоритмите за машинско учење даваат подобри резултати со предложената трансформација која го користи параметарот тежина на фактите.

Со помош на техниката тежина на фактите, во [18] номиналните атрибути беа трансформирани во нумерички. Користејќи ги тежините на фактите беше пресметана метриката вредност на информацијата на секој атрибут, што овозможи нивно рангирање и одбирање од над десет илјади на помалку од сто. Дополнителен предизвик во овој повеќезначен повеќекласен проблем беше агрегацијата на рангирањата по различните ознаки за да се добие единствено множество на атрибути. На Слика 6-1 е претставено рангирањето на овие атрибути според

максимална и просечна информациона вредност за податочното множество [50]. Се покажува дека некои атрибути иако имаат висока информативност за некоја класа, просечната информациона вредност им е ниска. Еден начин е атрибутите да се рангираат по некоја од метриките за агрегација и да се одберат одреден број на атрибути кои ќе се искористат за градење на еден општ класификациски модел соодветен за сите класи од повеќезначната класификација. Друг начин е да се одберат различни подмножества на атрибути, секое од нив соодветно за некој од подпроблемите.



Слика 6-1: Агрегација на информационата вредност кај повеќезначна класификација

Овој метод за рангирање искомбиниран со брз филтрирачки метод кој користи меѓукорелација, како што е опишано во Глава 4.3.1, овозможи добивање на мало репрезентативно множество на атрибити кое беше добиено многу ефикасно. Резултатите од предложениот метод и неговата иновативност овозможи тој да се најде во тесниот круг на финалисти на натпреварот Key risk factors for Polish State Fire Service: A Data Mining Competition at Knowledge Pit [50] (натпревар во податочно рударење за определување на клучните фактори на ризик кај полската противпожарна служба). За разлика од другите хибридни пристапи, кои користат генетски алгоритми [135] или алчно пребарување на просторот [143], овој метод за многу пократко време даде споредливи резултати, што е особено важно кај Големи Податочни множества, каде останатите обвиткувачки пристапи може и да станат неупотребливи.

6.2 Употреба на информационата добивка во медицината

Во [4] информационата добивка и методот за дискретизација предложен во [121] се употребени за пресметка на прагови на интервалите (cut-off values, анг.) на различни параметри во медицината. Имено, наоѓајќи интервали на вредности кои ја максимизираат информационата добивка, предложен е начин за наоѓањето на оптимални интервали на ризик од повреда на аналниот сфинктер при породувањето. Ова е нова примена на информационата добивка во медицината, каде до

сега интервалите на ризик на одредени параметри (пр. телесна маса, возраст и слично) во однос на некоја болест се генерирани врз основа на експертски знаења и искуства од областа. Применувајќи го методот за рангирање на атрибути врз основа на информационата добивка, пронајдени се оптималните интервали на повеќе параметри, така што пациентите чии параметри припаѓаат на ист интервал на вредности, имаат сличен ризик од одредена медицинска состојба, за разлика од остнатите пациенти.

Во [4] по откривањето на статистички значајните параметри со T-тест, беа пресметани праговите на итнервалите на слични вредности, како што е дадено во Табела 6.1. Почетната вредност на интервал 1 е минимумот, додека крајната вредност на интервал 2 е максимумот во анализираното податочно множество. Кога информационата добивка (колона ИД во Табела 6.1) е поголема, тогаш соодветниот параметар е поинформативен и покорисен за предвидување на целната класа (дали пациентот се здобил со повреда или не?). За некои параметри, на пример за времетраење на бременоста (ВБ), во првиот интервал до 280 денови 38% од пациентите имале повреда, а 62% немале. Но, во вториот интервал (над 280 денови ВБ), дури 90% од пациентите имале повреда, а само 10% немале. Откривањето дека токму 280 денови е прагот под кој и над кој значително се менува ризикот од повреда е постигнато со методот за надгледувана дискретизација кој се темели на максимизирање на информационата добивка. Со традиционалните статистички методи оптимизирање на интервалите на ризик не е можно. Вакviот модел овозможува откривање на оптималните интервали кои се базираат на максимизирање на информационата добивка, за разлика од традиционалните интервали на ризик, кои генерално се рачно дефинирани од експерти, па поради субјективноста на пристапот, понекогаш има спротиставени мислења.

Со предложената паралелизација успеавме да добиеме интервали кои се оптимални во однос на достапните податоци. Истите беа валидирани од медицински лица и потврдија дека примената на алгоритми од машинско учење носи бенефит. Паралелизацијата на алгоритамот за пресметка на информациона вредност е од особена важност, особено ако се има во предвид мноштвото медицински податоци кои се собираат секојдневно. Со нивна обработка може да се добијат посоодветни модели кои подобро ги опишуваат факторите на ризик. Денес, за жал тие податоци сè уште не се интегрирани во централизирани системи, туку се чуваат одделно во разни институции и медицински оддели, но сепак на светско ниво трендот на интегрирање е евидентен. Предложените паралелни имплементација на алгоритмите од машинско учење би можело да ги надмине пресметковните предизвици кои неминовно ќе се наметнат кога податоците од повеќе медицински извори ќе бидат интегрирани.

Табела 6.1: Интевали на сличен ризик за најважните параметри за предвидување на повреди на аналниот сфинктер.

		Интервал 1			Интервал 2		
Парам.	ИД	Интервал	Група 1	Група 2	Интервал	Група 1	Група 2
Воз.	0.043	[21,27.5)	83%	17%	[27.5,41]	53%	47%
Теж.	0.025	[55,85)	54%	46%	[85,130]	76%	24%
Вис.	0.021	[152,160)	38%	62%	[160,185]	62%	38%
БМИ	0.063	[18.4,28.3)	48%	52%	[28.3,42.8]	78%	22%
ВБ	0.222	[231,280)	38%	62%	[280,294]	90%	10%
ДБ	0.076	[46,50.5)	38%	62%	[50.5,57]	74%	26%
ТБ	0.066	[2260,3420)	42%	58%	[3420,4970]	72%	28%
ОГ	0.061	[31.5,36)	51%	49%	[36,41]	84%	16%

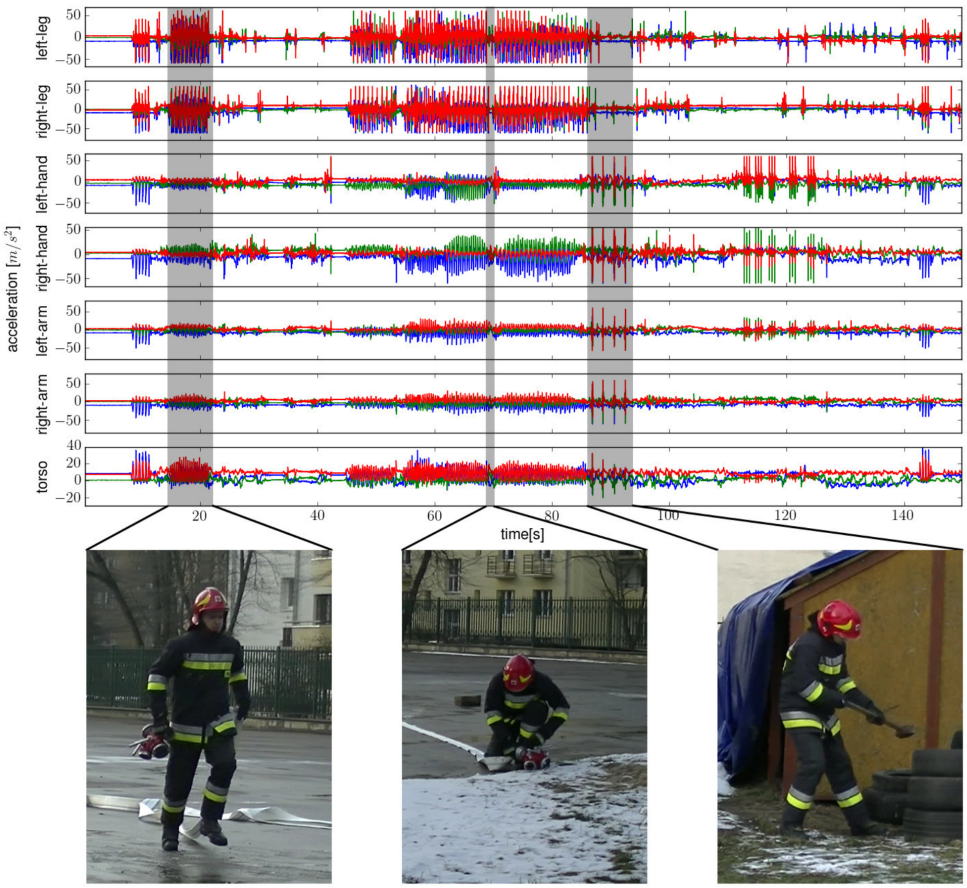
Парам.=Параметар, Воз.=Возраст на мајката, Теж.=Тежина на мајката,
Вис.=Висина на мајката, БМИ=Индекс на телесна тежина (Body Mass Index),
ВБ=Времетраење на бременеста, ДБ=Должина на бебе, ТБ=Тежина на бебе,
ОГ=Обем на глава, ИД=Информациона добивка,
Група 1 е групата на жени со повреда, Група 2 е контролната група.

6.3 Препознавање на активности на пожарникари

Сензорските мерења денес се користат за набљудување на различни процеси и целта е со анализа на мерењата да се подобрат тие процеси. Една таква примена е препознавањето на движењата и активностите на пожарникарите со помош на сензорски мрежи поставени на телото (body sensor networks, анг.). Опожарените места се сметаат за едни од попредизвикувачките околини во кои треба да се носат важни одлуки многу брзо поради динамичноста на средината [190]. Постојат повеќе иницијативи и научни проекти кои анализираат различни аспекти на ваквиот проблем [190, 191, 192]. Недостатокот на свесност за ситуацијата се смета за еден од главните фактори за незгоди на пожарникарите. Истражувањата презентирани во [190, 191, 192] имаат цел да ја зголемат безбедноста на пожарникарите преку следење на нивните кинематски и физиолошки услови за време на акциите за спасување од пожар. Ова може да води до развој на ефикасни системи за поддршка во донесувањето одлуки и да ја зголеми безбедноста на пожарникарите.

Токму поради тоа е организиран натпреварот за податочно рударење AAIA'15 Data Mining Competition [51]: Tagging Firefighter Activities at a Fire Scene (натпревар во податочно рударење за означување на активностите на пожарникарите на опожарени места). Целта на овој натпревар е преку инженерство на атрибути и градење робустни класификациски модели да се препознаат активностите на пожарникарите и позициите на телото. Податоците што се на располагање се собрани за време на тренинзи на осум пожарникари со помош на "паметна јакна" која содржи повеќе сензори поставени на телото со кои се следат активностите во реално време. Јакната се ги мери следниве типови на витални сигнали: електро-кардиограм (ЕКГ), пулс, рата на дишење и температура на телото. Дополнително, со фреквенција од 222 Hz се мерат движењата со помош на седум сетови од акцелерометри и жироскопи поставени на торзото, двете дланки, двете раце и двете нозе. Вежбите се снимени со видео камера заради рачно означување од страна на експерти. Пример од снимените сигнали и позициите на пожарникарите се дадени на Слика 6-2 [51]. Притоа, во секој момент е означена позицијата на пожарникарот (ползи, се движи, стои, се наведнува) и акцијата што ја прави (бара, манипулира со алат, не прави ништо, користи црево, се качува/слегува по скали, оди нагоре/надолу по степеници, оди, фрла/собира црево, удира со секира, пушта/сопира вода и други). Има вкупно 4 позиции и 16 активности, а вкупно 24 валидни комбинации на двете ознаки. Метриката за евалуација на решенијата на натпреварувачите беше тежински усреднета балансирана точност (balanced accuracy, анг.), при што првата ознака (позиција на пожарникарот) има тежина 1, а втората ознака (акцијата што се извршува) има тежина 2. Во тренинг и тест множеството има по четири пожарникари, но само за тренинг можеството се достапни ознаките.

Поради многуте сензори, податочното множество содржи 42 временски се-



Слика 6-2: Сензорски мерења и позиции на пожарникарите кај податочното множество AAIA'15

рии од акцелерометрите и жироскопите и уште 42 други нумерички вредности кои ги претставуваат виталните функции, но не претставуаат временски серии. Сегментацијата на временските серии е направена од организаторите, така што подвижните прозорци се долги 1.8 секунди. Проблемот е прилично сложен со оглед на сложеноста на позициите и активностите на пожарникарите, многуте временски серии и останати нумерички атрибути, тренирањето со податоци од едни пожарникари а тестирање со податоци од други, и тоа што постојат две ознаки (класи).

Во текот на овој натпревар беше развиен дел од методологијата за инженерство на атрибути, опишана во Глава 3.1. Со помош на оваа методологија, особено атрибутите кои се темелат на хистограми, а се генерирани од први изводи и делта серии, беше освоено трето место од повеќе од сто тимови [15, 51]. Беше постигнат краен резултат според метриката на натпреварот од околу 83%, што беше само за 1% послабо од победникот, но 2% подобро од четвртиот тим.

6.4 Предвидување на високи концентрации на метан во рудници

Друг пример на користење на сензори во продукциски системи е во рудниците. Генерално, рудниците претставуваат хазардна и опасна околина за работа. Рударите во рудниците за јаглен се соочуваат на различни ризици, како несреќи поврзани со експлозии на метан и карпи или долгорочни болести поради изложеноста на опасни гасови [193]. Рударството на јаглен е водечка причина за фатални повреди во САД [194]. Според National Institute for Occupational Safety and Health, ратата на фаталност на рударите во 2006 била 49.5 на 100,000 рудари, што е повеќе од 11 пати поголемо од ратата на фаталност во сите приватни индустрии.

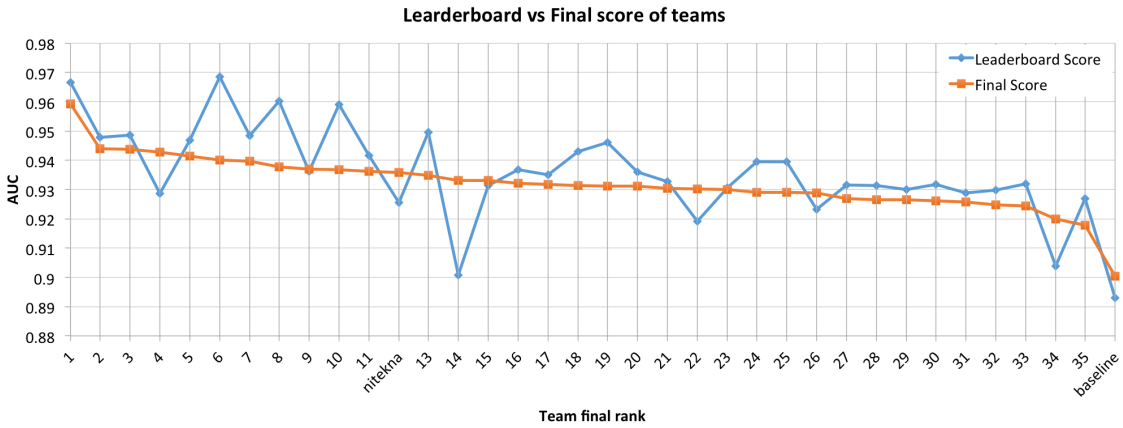
Вдишувањето на прашина од јаглен ја придизвикува болеста црни бели дробови кај рударите. Со цел да се обезбеди заштита на рударите, се користат системи за активно набљудување на производните процеси. Една од нивните основни примени е откривање на опасни концентрации на гас (особено метан) со цел да се спречат спонатани експлозии [195]. За таа цел, можноста да се предвидат опасни концентрации на гас во блиска иднина може да биде од поголема важност од самото следење на тековната состојба [191]. Во тој контекст е организиран и натпреварот IJCRS'15 Data Mining Competition [53]: Prediction of dangerous concentrations of methane in longwalls of a Polish coal mine (натпревар во податочно рударење за предвидување на опасни концентрации на метан во полски рудници за јаглен). Можноста да се предвидат опасни концентрации на гасови во рудниците може да спаси човечки животи, што ја потенцира важноста на оваа тема и примената на развиените алгоритми за инженерство и одбирање на атрибути.

Податочното множество за натпреварот е обезбедено од реални мерења во текот на неколку години од активен рудник за јаглен во Полска [53]. Целта е да се предвидат опасни концентрации на метан на три места во рудникот. Сечилото се движи во просторот каде се поставени мерачите на концентрација на метан. Колку е повисока струјата што ја троши сечилото, толку е поефикасно рударството, што теоретски значи дека повеќе метан се ослободува во воздухот. Ако се достигне некој предупредувачки праг на концентрацијата на метан, тогаш автоматски сечилото се исклучува, целиот рудник прекинува со работа и рударите се евакуираат. Ваквите прекини носат финансиски загуби за рудниците, но моментално се неопходни за да се заштитат рударите. Но, кога би можело со сигурност да се предвиди високата концентрација однапред, тогаш брзината на сечилото може да се намали превентивно, што теоретски ќе резултира со намалено ослободување на метан и ќе има повеќе време за вентилација. На овој начин исклучувањето на целата производна линија може да стане непотребно. Потенцијалната корист за рудниците е очигледна: ќе се спречат финансиски загуби без

да се зголемува ризикот за рударите.

Податочните множества од натпреварот се состојат од мерења на 28 сензори со фреквенција од 1 Hz и однапред се сегментирани во интервали од 10 минути. Во тренинг множеството постои преклопување на инстанците, додека во тест множеството нема преклопување. Резултатите се во вид на веројатносни предвидувања за појавување на висока концентрација на метан во следните 3 до 6 минути по крајот на интервалот од мерењата, за секое од трите мерни места посебно, а метрика за евалуација е површина под кривата на приемникот (AUC ROC).

Во текот на натпреварот беа евалуирани повеќе атрибути генерирани со методологијата за инженерство на атрибути, опишана во Глава 3.1. Беше откриено дека природата на овој проблем е таква што вредностите во временските серии се менуваат многу бавно и дека релевантни се само сензорите поставени на мерните места од интерес, додека останатите сензори низ рудникот не носат дополнителна информација. Исто така беа демонстрирани продобивките во однос на предиктивните перформанси од мрежно пребарување на параметрите за машини со носечки вектори опишани во Глава 2.2.1. Со овој метод беше спречено претренирање, што беше случај кај многу други тимови кои имаа силни перформанси при прелиминарната евалуација, но слаби при финалната, како што може да се забележи од Слика. 6-3 [12]. Со предложените атрибути и метод за оптимизирање на перформансите на класификаторите направивме огромен скок од 26 места од прелиминарното до финалното рангирање [53], поради се бевме меѓу финалистите и нашиот пристап беше поканет за презентација на конференцијата.



Слика 6-3: Прелиминарни (leaderboard) и конечни (final) резултати на натпреварот IJCRS 2015

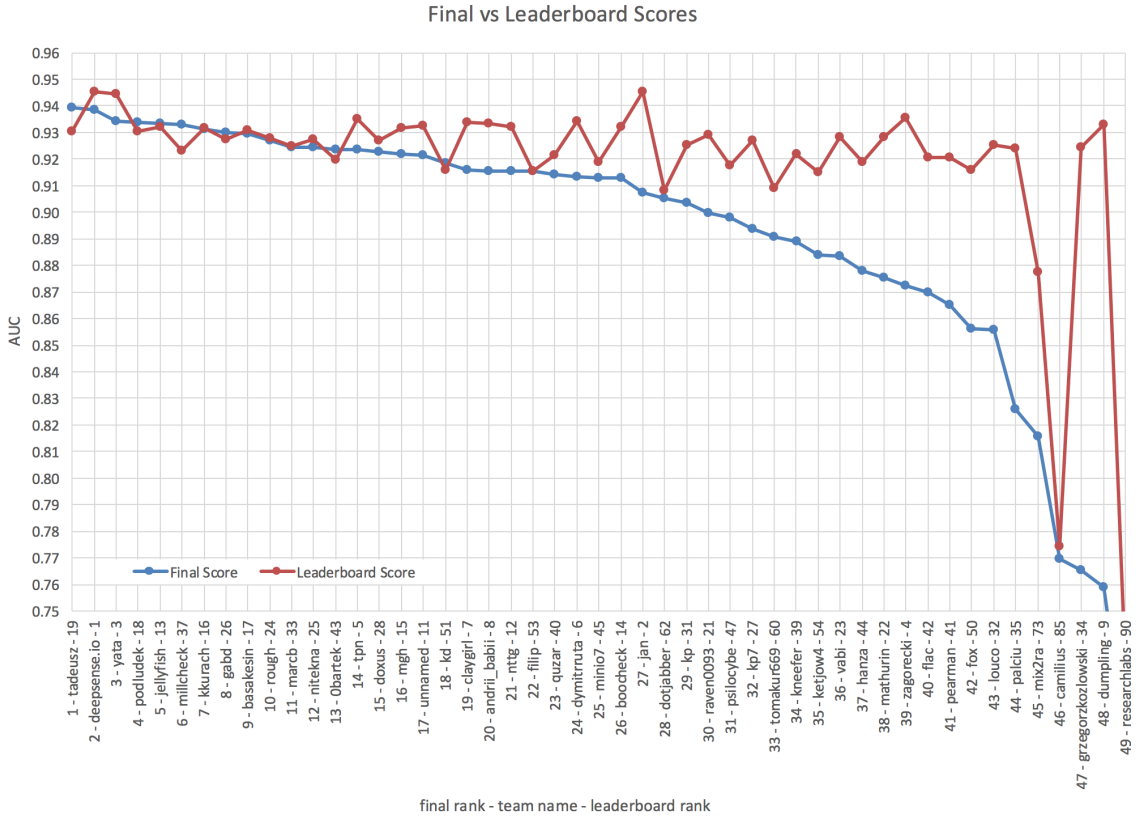
6.5 Предвидување на сеизмички активности во рудници

Претходниот пример покажува како може да се употребуваат сензорите во индустријата. Слично на него, на натпреварот AAIA'16 Data Mining Competition [52]: Predicting Dangerous Seismic Events in Active Coal Mines (натпревар за податочно рударење за предвидување на опасни сеизмички настани во аквтивни рудници за јаглен), се достапни реални мерења од различни сензори во период од неколку години. Целта на овој натпревар е да поттикне развој на издржлив модел за детектирање на период со зголемена сеизмичка активност која ги загрозува рударите кои работат во рудници за јаглен. Достапните податоци претставуваат агрегации на ниво на саат во рамките од 24 часа од повеќе сеизмички сензори кои ги бројат сеизмичките удари и ја мерат нивната енергија. Дополнително, постојат и номинални атрибути кои ги земаат во предвид последните евалуации од страна на експерти за сеизмичката состојба во рудникот. Моделите треба да предвидат дали во текот на 8 часа по 24-часовниот период кој се следи вкупната енергија на сеизмички удари надминува некое предефинирано алармантно ниво. Мотивацијата за анализа и развој на овие проблеми е тоа што ако успешно се предвидат сеизмичките активности, рударите може да бидат заштитени предвреме. Дополнително ова може да ја олесни анализата на причинско-последичната поврзаност помеѓу активноста во рудникот (минирање, копање, итн) и сеизмичките настани.

Во целото податочно множество постојат мерења од повеќе мерни места во повеќе рудници, па дополнителен предизвик претставува тоа што предиктивните модели треба да се изградат врз основа на податоци од едни рудници во некој временски период, а да се применливи и на други рудници во друг временски период. Предизвикувачки е и тоа што класната дистрибуција е многу небалансирана, и тоа што тренинг и тест множествата имаат различна класна дистрибуција.

Нашето решение на овој реален проблем, кое не користи никакво рачно моделирање или познавања од доменот на проблемот, е опишано во [7]. Во него се искористени повеќе методи кои се предложени при изработката на оваа докторска дисертација. За трансформација на номиналните атрибути (локации на работните места, експертските анализи и други) во нумерички се искористени методите опишани во Глава 3.2.1 и Глава 3.2.2. За инженерство на атрибути од временските серии (разните агрегации на ниво на час) е искористена методологијата предложена во Глава 3.1. За одбирање на атрибутите е искористен методот опишан во Глава 4.3.2 за отстранување на атрибути кои се осетливи на концептуални отстапувања, што се покажа како многу важно поради различнот временски период и различните рудници во тест множеството. Како резултат на експериментите, дополнително беше откриено дека ансамблите од повеќе модели

се помоќни отколку поединечните модели. Конечните резултати од натпреварот покажаа дека многу тимови имале претренирани модели што резултираше во драстични падови од прелиминарните до финалните резултати, како што може да се забележи од Слика 6-4. Ова ја потенцира ефикасноста на алгоритамот за одбирање кој успешно ги отстранува атрибутите кои се осетливи на концептуални отстапувања, што резултира во постабилни модели.



Слика 6-4: Прелиминарни (Leaderboard) и конечни (final) резултати на натпреварот AAIA 2016

6.6 Потиснување на лажни аларми на мониторите во одделите за интензивна нега

Мониторите во одделите за интензивна нега ги снимаат и следат виталните параметри на пациентите. Постојат повеќе системи за собирање на податоците [196] со многу осетливи сензори, што е потребно за точно идентификување на сите опасни нивоа на некои параметри. И покрај тоа, овие системи моментално имаат ограничени способности за обработка на необработените податоци. Поради тоа, носењето одлуки при дијагностицирањето се сведува на анализа на необработените податоци. Заради високата осетливост на сензорите, се случуваат многу

лажни аларми. Според [197], 72% до 99% од алармите се лажни, поради што кај медицинскиот персонал се јавува замор од алармите. Зголемениот број на аларми ја намалува брзината на реакција на медицинските лица, кои дури понекогаш свесно ги игнорираат алармите.

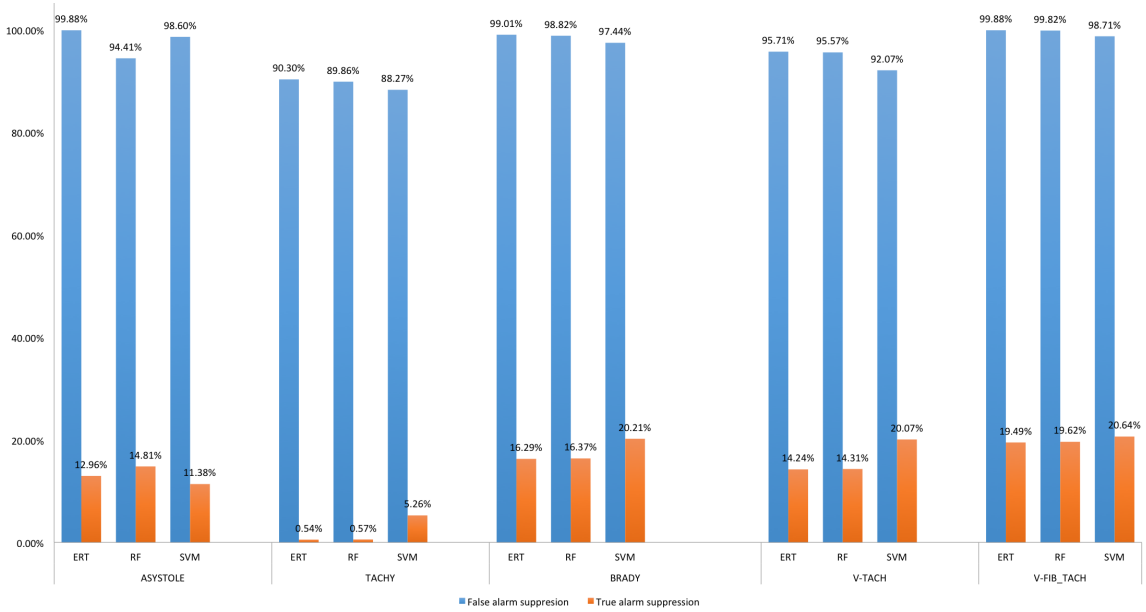
За да се поттикнат истражувањата поврзани со потиснување на лажни аларми од монитори во одделите за интензивна нега, рачно е означено подмножество од MIMIC II податочното множество [198, 196]. Алармите издадени од мониторите рачно се проверени од експерти и лажните аларми се означени, поради што ова податочно множество претставува "златен стандард" [199]. Подмножеството на рачно означени аларми е одбрано по два критериуми. Прво, треба да се случил критичен електро-кардиограм (ЕКГ) за аритмија. Второ, треба да се достапни асистолниот крвен притисок и барем еден ЕКГ канал за време на алармот. Пет типови на аларми се анотирани: асистолен (asystole, анг.), екстремна тахикардија (extreme tachycardia - TACHY, анг.), екстремна брадикардија (extreme bradycardia - BRADY, анг.), вентрикуларна тахикардија (ventricular tachycardia - V-TACH, анг.) и вентрикуларна фибрилација (ventricular fibrillation - V-FIB/TACH, анг.).

Постојат повеќе обиди за креирање на генерален модел за податочна фузија и за поддршка во донесувањето на одлуки со помош на големи количини медицински податоци, особено од монитори во одделите за интензивна нега [200]. Една од поважните области е потиснување на лажните аларми, и еден од пристапите за ова користи техники за процесирање на сигнали и податочно рударење [201]. Друг пристап опишан во [202] користи теорија на игри за одбирање најважните атрибути добиени со вејвлет трансформација од влезните временски серии. Во [203] авторите користат повеќе индекси за квалитет на ЕКГ сигналите пред случувањето на алармот.

Алгоритмите што се развиваат за потиснување на лажните аларми мора да ги обработуваат сигналите релативно брзо, така што алармот не смее да се случи подоцна од 10 секунди од настанот што го причинил [204]. Ова не е секогаш едноставна задача бидејќи количеството на податоци што треба да се обработи бара значителни пресметковни ресурси и време.

За инженерство на атрибути од временските серии во пристапот што беше предложен во [3] е искористена методологијата објаснета во Глава 3.1, додека за одбирање на атрибутите е искористен брз филтрирачки метод базиран на информационата добивка, како што е опишано во Глава 4.1.2. За градење класификациски модели се искористени алгоритмите случајни шуми, екстремно случајни дрва и машини со носечки вектори.

Процентот на потиснати лажни и вистински аларми со предложениот пристап за секој тип на аларм е даден на Слика 6-5. За алармите ASYSTOLE, BRADY, V-TACH и V-FIB/TACH беа потиснати од 95.5% до 99.9% од лажните аларми,



Слика 6-5: Потиснување на лажни аларми (false alarm) и потиснување на вистински аларми (true alarm) за секој тип на аларм и класификациски алгоритам.

но истовремено процентот на потиснати вистински аларми беше до 20%, што е неприфатливо. Сепак, за алармот TACHY беа потиснати 90% од лажните аларми додека само околу 0.5% од вистинските аларми беа погрешно потиснати. Овој резултат е доста ветувачки и може да води кон фино прилагодување на праговите на одлука и кај другите типови на аларми, така што ќе се намали процентот на потиснати вистински аларми.

Главна предност на предложениот метод е тоа што користи автоматско инженерство на атрибутите, со што се намалува времето за рачно креирање на атрибути за овој проблем. Со методологијата за инженерство на атрибути беше брзо откриено дека многу од другите сигнали освен артериският крвен притисок не се целосни и недостасуваат кај многу аларми, па затоа се неупотребливи. Исто така, беше откриено кои типови на атрибути се информативни. Интересно, некои атрибути беа информативни за еден тип на аларм, додека воопшто не беа информативни за други типови на аларми. Ваквите својства на атрибутите е многу тешко да се знаат однапред, што го потенцира главниот недостаток на рачното инженерство на атрибути. Од друга страна, со предложената методологија за секој тип на аларм автоматски беа одбрани најсоодветните атрибути. Ова е многу корисно бидејќи одбраните атрибути може да појаснат кои параметри се карактеристични за секој аларм. Исто така беше откриено дека сигналот кој ги содржи отчукувањата на срцето е непотребен бидејќи генерираните атрибути од асистолниот крвен притисок се многу поинформативни. Дополнително беше потврдено дека делта сериите се поинформативни отколку оригиналните временски серии на асистолниот крвен притисок, како и дека атрибутите кои се темелат на

хистограми и перцентили се поинформативни отоколку обичните статистички метрики.

Врз основа на изложените резултати беше заклучено дека методологијата за инженерство на атрибути од временски серии е разноврсна и применлива за секаков вид на сигнали. Во [3] се изложени и некои идеи како може да се подобрат перформансите, на пример со вклучување на други параметри од пациентите (пр. возраст, пол, медицинска историја, итн), вклучување на други сигнали (пр. ЕКГ), користење на колаборативно филтрирање за креирање на модели на пациенти со слични карактеристики, итн. Исто така може да дефинираат или искористат веќе постоечки метрики [205] за проверка на квалитетот на сигналот со цел да се одлучи дали да се користи моделот со машинско учење или не. Доколку сигналот не е кавалитетен, тогаш автоматското потиснување на сигналите може да се исклучи.

6.7 Препознавање на активности во амбиентално потпомогнато живеење

Зголемената возраст на граѓаните во развиените земји бара оптимизирање на здравствените системи. Трошоците за грижа на восрасните граѓани се зголемуваат, додека бројот на луѓе кои придонесуваат во здравствените фондови се намалува [206]. Според истиот извор инвестициите во медицински науки, технологии и техники за третман имаат потенцијал да ја намалат цената на здравствените услуги во среден и долг рок.

Во таа насока, амбиентално потпомогнатото живеење (АПЖ) е релативно нов тренд кој има за цел да овозможи нови производи, услуги и процеси транспарентно да им помагаат на луѓето (најчесто повозрасни) во нивните домови со цел да им се подобри квалитетот на живот [207]. Во студиите поврзани со АПЖ препознавањет на дневните активности на живеење (ДАЖ) со помош на сензорски мерења станува популарно [40, 41, 42, 43, 44] поради неколку причини. Прво, здравствените придобивки поврзани со физичката активност зависат од нејзиниот интенизитет, времетраење и фреквенција [208, 209, 210], па затоа е важно точно да се процени. Второ, промените во ДАЖ може да се показател на некоја медицинска која се развива пред да стане критична [211], па затоа препознавањето на ДАЖ е важно. Трето, препознавањето на човечките активности е важно за откривање на незгоди и непосредни итни случаи (пр. падови, губење свест, итн).

Генерално пристапите за препознавање на ДАЖ користат статистички методи и техники за процесирање на сигнали со кои се генерираат атрибути од необработените сензорски мерења. Повеќето пристапи користат рачно генерира-

ни атрибути врз основа на препораките од литературата и се приспособени за активностите што треба да се препознаваат [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]. Некои атрибути, како на пример различните статистички метрики, се користат во многу студии, додека други студии користат некои поспецифични атрибути. Поради тоа има одредена несигурност и незнаење за точно кои атрибути се навистина корисни за препознавање на активности. Друга забуна настанува поради тоа што некои студии користат повеќе сензори од други. Исто така, во некои студии се прави мазнење на сигналот и отстранување на шумот, а во други не.

Користејќи ја методологијата за инженерство на атрибути од временски серии, опишана во Глава 3.1, и хибридниот пристап за одбирање на атрибути, опишани во 4.3.4, во студиите [1, 2] се адресирани наведените проблеми. Предложените пристапи беа евалуирани на пет податочни множества поврзани со препознавање на ДАЖ и резултатите беа споредени со повеќе студии кои ги користат истите податочни множества. Освен придобивките кои потекнуваат од автоматското инженерство и одбирање на атрибути, независно од достапните сензори, целта на студијата [1] беше да се анализираат кои сензори и на кои локации на телото треба да се постават, како и кои атрибути треба да се генерираат. Со ваквите сознанија ќе се намалат интрузивноста и одбивноста на пациентите, затоа што ќе се користат помалку сензори, а воедно ќе се намали и цената на системите за АПЖ без да се загрозуваат перформансите. Токму овие својства се неопходни за успешно распростанување и прифаќање на системите за амбиентално потпомогнато живеење.

При правењето на експериментите, учесниците беа поделени на три подмножества за тренирање, валидација и тестирање. Податоците собрани од учесниците во тренинг множеството се користат за градење на класификациони модели, а тие од валидациското множество во обвитукувачкиот процес на одбирање на атрибути и фино подесување на параметрите на класификационите алгоритми. Сите финални модели откако ќе се одбеат атрибутите и ќе се подесат параметрите на класификаторите се градат со помош на унија од тренинг и валидациските подмножества. Тест множеството е целосно независно и служи само за проверка на перформансите на изградените модели.

Освен ваквата стратегија за евалуација, други студии користат и стратегија на изоставување на еден учесник (leave-one-subject-out, анг.) [40, 79, 80, 212] или вкрстена валидација со 10 групи [45, 47]. Од овие стратегии, таа со независно тест множество се смета за најпесимистична [54, 213], поради што таа беше одбрана во нашата студија [1].

Студијата опишана во [40] предлага хиерархиска класификација на ДАЖ врз основа на сензорски отчитувања. Податочното множество генерирано во таа студија се нарекува DaLiAc (Daily Living Activities) и истото е искористено и во студијата [1]. Притоа 19 учесници носат Shimmer сензори [214] поставени на десниот колк, градите, десниот рачен зглоб и левиот глужд. Во оваа студија се

анализирани следниве 13 активности: одење, станување, седнување, качување по скали, слегување по скали, лежење, трчање, возење велосипед во две нивоа на интензитет, миење чинии, чистење со правосмукалка, метење и скокање на јаже.

Во [45] е опишан дизајнот, имплементацијата и валидацијата на платформа за агилен развој на здравствени мобилни апликации и притоа е генерирано податочно множество кое се нарекува mHealth. Вкупно 10 учесници носат Shimmer2 сензори [214] поставени на градите, десниот рачен зглоб и левиот глужд. Дополнително се обезбедени и две-канални ЕКГ отчитувања. Има вкупно 12 активности од интерес: одење, станување, седнување, качување по скали, лежење, трчање, возење велосипед, наведнување, кревање на рацете, клекнување и скокање напред-назад.

Студијата [47] користи пет мобилни телефони поставени на пет различни локации: десиот џеб на фармерките, левиот џеб на фармерките, на каишот во правец на десната нога, на десната рака во висина на бицепсот, и десниот рачен зглоб. Ова податочно множество се нарекува FSP (Five Smart Phones) и се користи за утврдување кои локации на телото се најсоодветни за препознавање на вкупно 7 ДАЖ: одење, станување, седнување, качување по скали, слегување по скали, лесно трчање и возење велосипед.

Во [215, 48] е опишано податочното множество SBHAR (Smartphone-based Human Activity Recognition) кое содржи отчитувања од мобилен телефон кој се носи на половината. Освен активностите опфатени во ова множество, во неговата проширена варијанта SBHARPT [49] се вклучени и премините од една во друга активност. Вкупно 6 активности од интерес се анализирани: одење, станување, седнување, качување по скали, слегување по скали и лежење. Во проширеното множество SBHARPT се вклучени и уште 6 премини од една во друга позиција на телото (активност): стоење во седење, седење во стоење, седење во лежење, лежење во седење, стоење во лежење и лежење во стоење.

Во Табела 6.2 се дадени карактеристиките на податочните множества кои бе споменати претходно и се анализирани во детали во [1].

Влијанието на методите за одбирање на атрибути може да се забележи од Табела 6.3, Слика 6-6 и Слика 6-7. И методот кој ги отстранува атрибутите кои не се информативни или се осетливи на концептуални отстапувања, опишан во Глава 4.3.2, и методот кој прави диверзифицирано пребарување напред-назад со цел да се најде оптимално множество на атрибути, опишан во Глава 4.3.3, значително го намалуваат бројот на атрибути. Со првиот, бројот на атрибути е намален 3 до 5 пати, притоа и зголемувајќи ја прецизноста. Вториот дополнително ги намалува атрибутите и за сите податочни множества се одбираат до шеесетина атрибути, со што значително се забрзуваат алгоритмите (види Слика 6-8 и Слика 6-9).

Табела 6.2: Информации за податочните множества за амбиентално потпомогнато живеење.

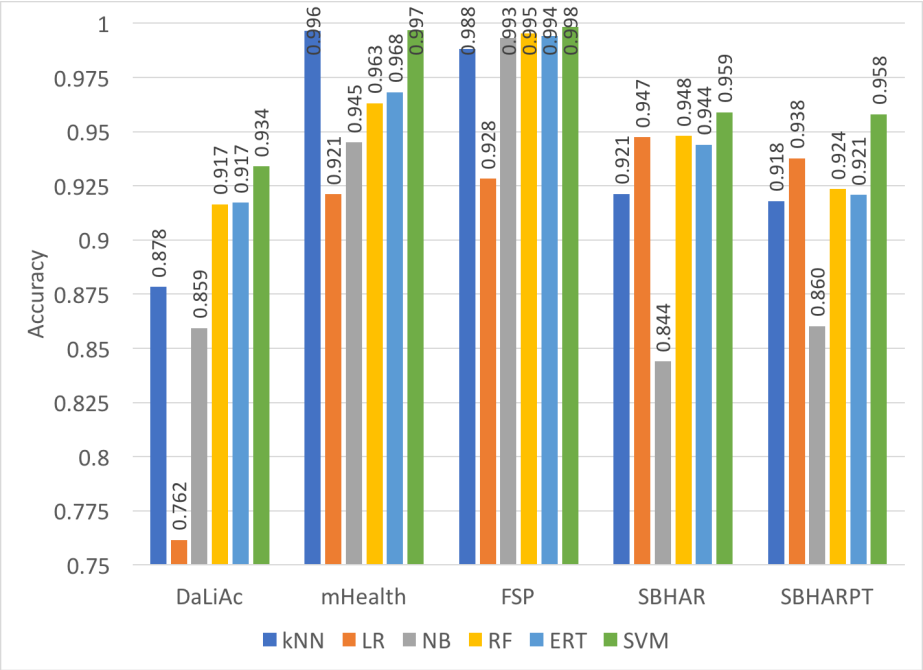
	Инстанци			Субјекти						Број на временски серии				
Множество	Трен.	Вал.	Тест	Трен.	Вал.	Тест	Фрекв.	Проз.	Прекл.	Ориг.	Магн.	Делта	ПИ	БФТ
DaLiAc	3356	2299	3150	7	5	7	204.8	5	2.5	24	8	32	32	96
mHealth	2662	1241	2549	4	2	4	50	4	3	23	8	31	31	91
FSP	5012	2506	5012	4	2	4	50	2	1	60	20	80	80	240
HAR	5132	2220	2947	14	7	9	50	2.56	1.28	6	2	8	8	24
SBHARPT	5468	2374	3192	14	7	9	50	2.56	1.28	6	2	8	8	24

Инстанци е бројот на сегменти со лизгачки прозорец; Трен., Вал. и Тест се тренинг, валидациско и тест подмножество, соодветно;
Фрекв. е фреквенцијата на отчитување во Hz; Проз. е должината на прозорецот во секунди; Прекл. е преклопувањето на соседните прозорци;
Ориг. е бројот на оригинални временски серии; Магн. е бројот на магнитудни временски серии; Делта е бројот на делта серии;
ПИ е бројот на временски серии на први изводи; БФТ е бројот на серии кои се темелат на брза Фуриева трансформација.

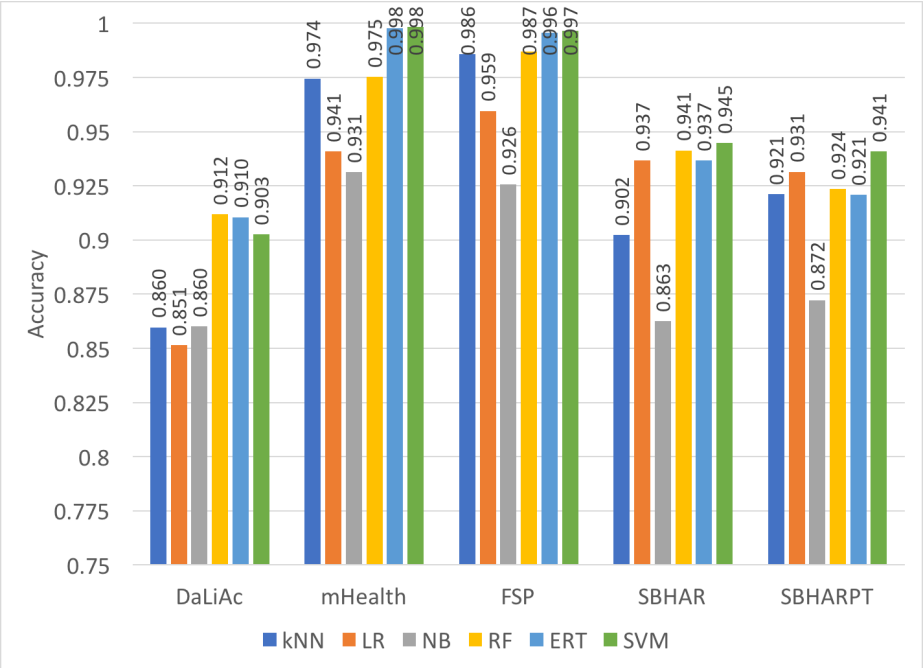
Табела 6.3: Број на атрибути и највисока прецизност по податочно множество и начин на одбирање на атрибутите.

	Без ОА		ОА Конц. Отс.		ОА Преб. Напред-Назад	
Множество	Атриб.	Прец.	Атриб.	Прец.	Атриб.	Прец.
DaLiAc	4871	0.916	1083	0.934	60	0.912
mHealth	3232	0.972	620	0.997	39	0.998
FSP	11418	0.996	2409	0.998	44	0.997
SBHAR	1415	0.922	316	0.959	47	0.945
SBHARPT	1236	0.925	475	0.958	60	0.941

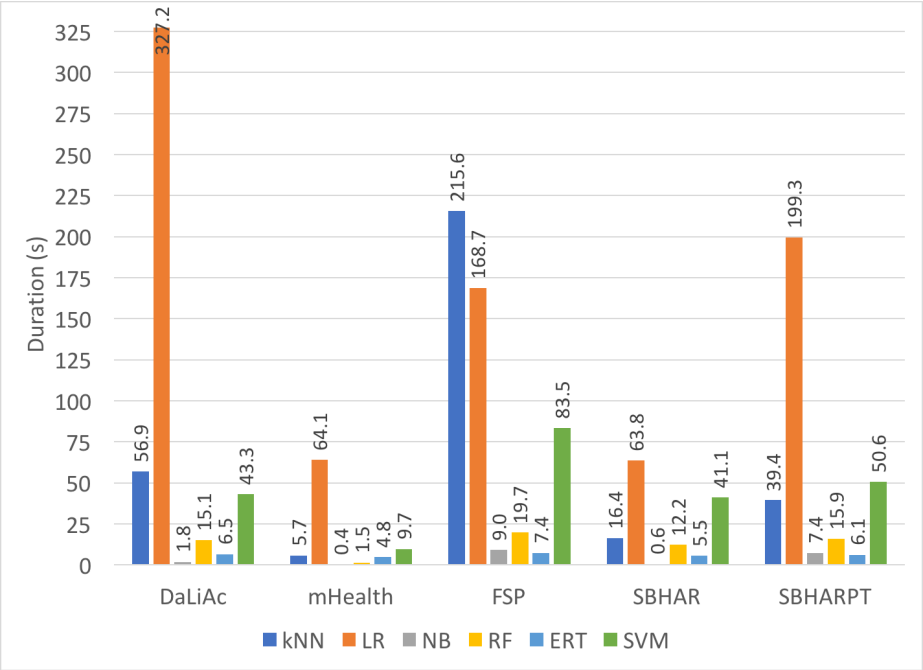
Атриб. е бројот на атрибути; Прец. е прецизност (accuracy) со најдоброто множество на атрибути; Без ОА е кога се користат сите генерирани атрибути без да се одбираат; ОА Конц. Отс. се однесува на множеството на атрибути добиено по одбирањето на атрибути кои не се осетливи на концептуални отстапувања; ОА Преб. Напред-Назад се однесува на множеството на атрибути добиено со хибридниот метод за одбирање кој прави диверзивификувано пребарување напред-назад.



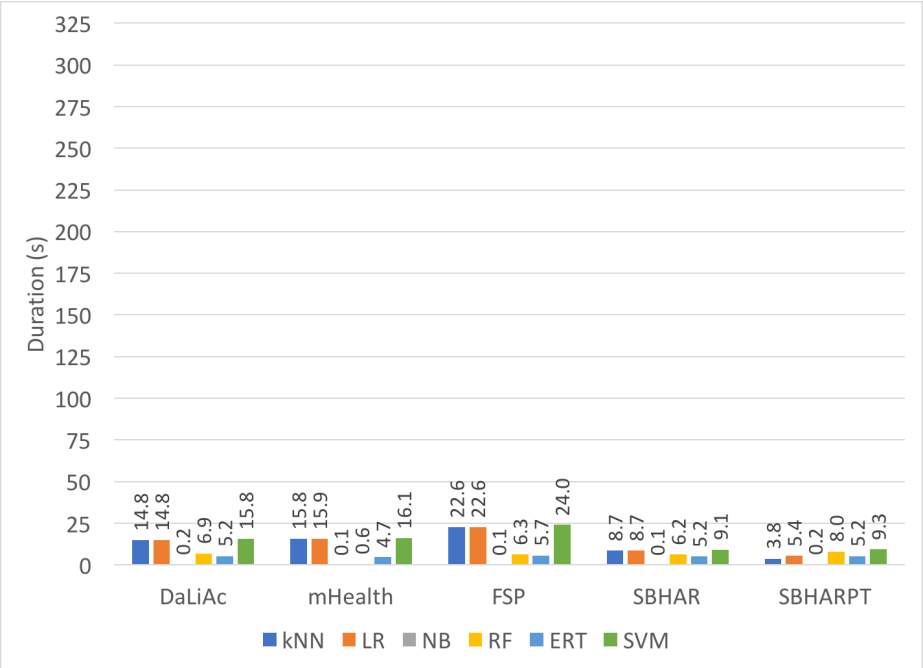
Слика 6-6: Прецизност по класификациски алгоритам и податочно множество со оптималното множество на атрибути добиено со отфрлање на атрибути кои не се информативни или се осетливи на концептуални отстапувања.



Слика 6-7: Прецизност по класификациски алгоритам и податочно множество со оптималното множество на атрибути добиено со методот кој прави диверзифицирано пребарување напред-назад.



Слика 6-8: Време по класификациски алгоритам и податочно множество со оптималното множество на атрибути добиено со отфрлање на атрибути кои не се информативни или се осетливи на концептуални отстапувања.



Слика 6-9: Време по класификациски алгоритам и податочно множество со оптималното множество на атрибути добиено со методот кој прави диверзифицирано пребарување напред-назад.

За да се утврди влијанието на различните типови на сензори и различните локации на поставување на сензорите на телото, за сите податочни множества се испитани перформансите добиени од секој сензор поединечно, но и комбинации од сите сензори. Притоа, постапките на одбирање на атрибути се повторуваат врз иницијалните множества од сите атрибути генерирани од временските серии на сензорите кои се евалуираат. На овој начин е овозможено искористување на најдобрите атрибути од секоја локација на сензори.

Од прикажаните резултати може да се заклучи дека и двата предложени хибридни методи за одбирање на атрибути се ефикасни за добивање на репрезентативни податочни множества. Со споредба на добиените резултати со оригиналните студии кои ги објавиле податочните множества покажано е дека методот за инженерство на атрибути е фунцкионален и резултира во подобри предиктивни перформанси отколку со разчно дизајнираните и одбрани атрибути. Со откривањето на мали множества на атрибути се овозможува генерирање на робустни класификациони модели кои ќе се извршуваат на уреди со мала пресметковна и мемориска комплекснот. Дополнително, преку евалуација на различни комбинации на локации и типови на сензори е откриено кои локации на сензорите се неопходни и кои локации се меѓусебно редундантни, за секое податочно множество посебно.

Исто така, со предложените методи за инженерство и одбирање на атрибути, покажано е дека само со сензорите вградени во паметен телефон и паметен часовник може да се постигне висока прецизност на препознавање на активности. Ова откритие е важно и охрабрувачко бидејќи овие два уреди се многу широко распространети и не се сметаат за премногу интрузивни. Поради тоа, тие може да се користат во системи за амбиентално потпомогнато живеење за препознавање на активности без притоа да има аверзија и одбивност, бидејќи луѓето веќе можеби и ги користат потребните уреди. Со ова може да се намалат трошоците за правење на системи за АПЖ бидејќи не е потребен специјализиран хардвер. Подетално изложени резултати и дискусија за резултатите од оваа примена на предложените методи е изложена во [1].

Табела 6.4: Информација за Големото податочното множество кое се агрегира за генерирање на атрибути

Параметар	Вредност
Формат на редиците во датотеките	JSON
Број на оригинални колони	17
Број на агрегирани колони	86
Број на останати неагрегирани колони (клучеви и сл.)	26
Број на S3 објекти	410K
Големни на на множеството (GB)	30
Број на редици	44M
Излезни неагрегирани редици	44M
Излезни агрегирани редици	2M

6.8 Агрегација на Големи Податоци

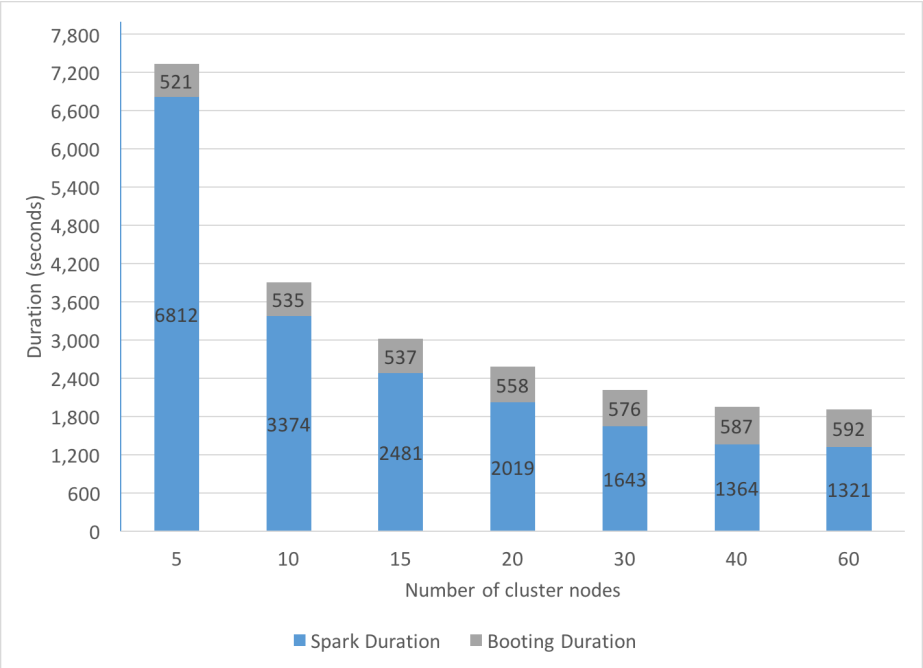
Во оваа студија на случај е направена евалуација на методот предложен во Глава 5.4 за генерирање на атрибути од Големи Податоци со помош на агрегација.

Кластерите користени во оваа студија на случај се извшуваа во облакот на Amazon Web Services (AWS), заради популарноста во индустријата и научната заедница. Од сите достапни типови на инстанци, само 'R3' и 'R4' се оптимизирани за мемориски-интензивни апликации и нудат најдобра цена по GB RAM меморија [216]. Бидејќи R4 немаат локален диск простор, тие не се соодветни за Spark кластери кои резултатите би ги запишувале во HBase или на HDFS. Исто така, големината на инстанци 'r3.large' не е соодветна за Spark додека 'r3.xlarge' нуди послаби мрежни перформанси во споредба со поголемите инстанци. Поради тоа за експериментите беше одбрана 'r3.2xlarge' инстанцата која има 61 GB RAM, 8 CPUs, 160 GB SSD и чини 0.665 USD на час во моментот на пишување на оваа дисертација.

Во Табела 6.4 се дадени информациите на податочното множество кое се однесува за корисници на видео услуги, слични на Netflix. Ваквото податочно множество се генерира дневно и постапката што е опишана во оваа студија на случај се повторува секој ден. Во експериментот кластерите имаа 5, 10, 15, 20, 30, 40 и 60 јазли со цел да се евалуира скалабилноста и забрзувањето.

Времетраењето на Spark програмата како и потребното време за стартување на кластерот (booting, анг.) се прикажани на Слика 6-10. Цената на извршување на целата програма е дадена на Слика 6-11. Очигледно, кластерите со 5 и 15 јазли се најефтини, со тоа што тој од 15 јазли работата ја завршува многу побрзо.

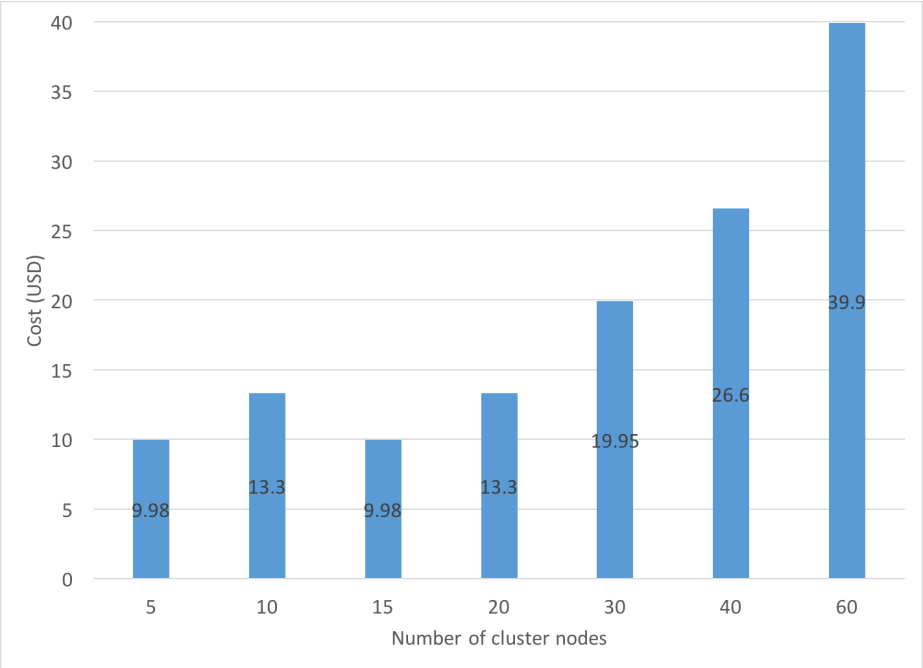
За сите конфигурации на кластерите, времето на стартување на кластерот не се наплаќа, но сепак треба да се земе во предвид при димензионирањето за



Слика 6-10: Времетрање на Spark програмата и времето за стартување на кластерите за агрегирање на Големи Податоци.

да се гарантира дека агрегацијата ќе се заврши во максималното предвидено време. Ова време многу не варира во зависност од бројот на јазли во кластерот, нешто што беше поврдено при многу извршувања на постапката опишана во оваа студија на случај. Сепак при статување на кластери со стотици јазли, времето вариираше повеќе, што треба да се земе во предвид кога се користат кластери по потреба (on-demand, анг.).

Времетраењето на Spark програмата значително се забрза кога бројот на јазли се зголеми од 5 на 10, намалувајќи се од 6812 до3374 секунди, како што може да се забележи од Слика 6-10. Причината за ова супер-линеарно забрзување е тоа што кога се користат само 5 јазли целото податочно множество не се собира во меморијата, поради што се прави запишување на дискот. Кај поголемите кластери, ова не е случај. Сепак како што бројот на јазли се зголемува, забрзувањето се намалува. На пример, со зголемувањето на бројот на јазли од 40 на 60, нема значително намалување на времето на извршување. Ова покажува дека зголемувањето на бројот на јазли треба да е во согласност со големината на податочното множество. За мали податочни множества или многу големи кластери, всушност паралелизацијата може да предизвика намалување на перформансите. Така, при извршување на истата постапка за ова податочно множество на кластери со 100 и 200 јазли, перформансите беа многу полоши отоколку кога се користат мали кластери до 10 јазли. Од ваквите експерименти може да се заклучи дека има физички лимит на колку брзо да се заврши агрегацијата, и воопшто некој процес на обработка на податоци.



Слика 6-11: Цена на кластерите за агрегирање на Големи Податоци.

Со оваа студија на случај покажавме дека може да се скалира инженерството на атрибути од Големи Податоци со помош на аграгација. Покажавме дека со употреба на соодветни технолгии, дури и кога податочните множества се големи, постапката може да биде завршена брзо и ефтино со користење на кластери во облак само во моментите кога тие се потребни.

6.9 Паралелно одбирање на атрибути кај Големи Податоци

Оваа студија на случај се однесува на два аспекти од анализата на Големи Податоци. Прво, потребни се ефикасни начини на запишување на податоците во NoSQL база на податоци. Второ, потребно е да се евалуира скалабилноста на предложената паралелна имплементација на филтрирачките методи за одбирање на атрибути.

6.9.1 Ефикасно запишување на Големи Податоци

Првиот чекор кој претходи на било каква анализа на Големи Податоци е нивното ефикасно запишување во NoSQL база на податоци што овозможува скалабилност. За таа цел беа тестирани различните дизајни на примарни клучеви на

HBase базата на податоци предложени во Глава 5.2. Експериментите од ваквата евалуација се објаснети во [11].

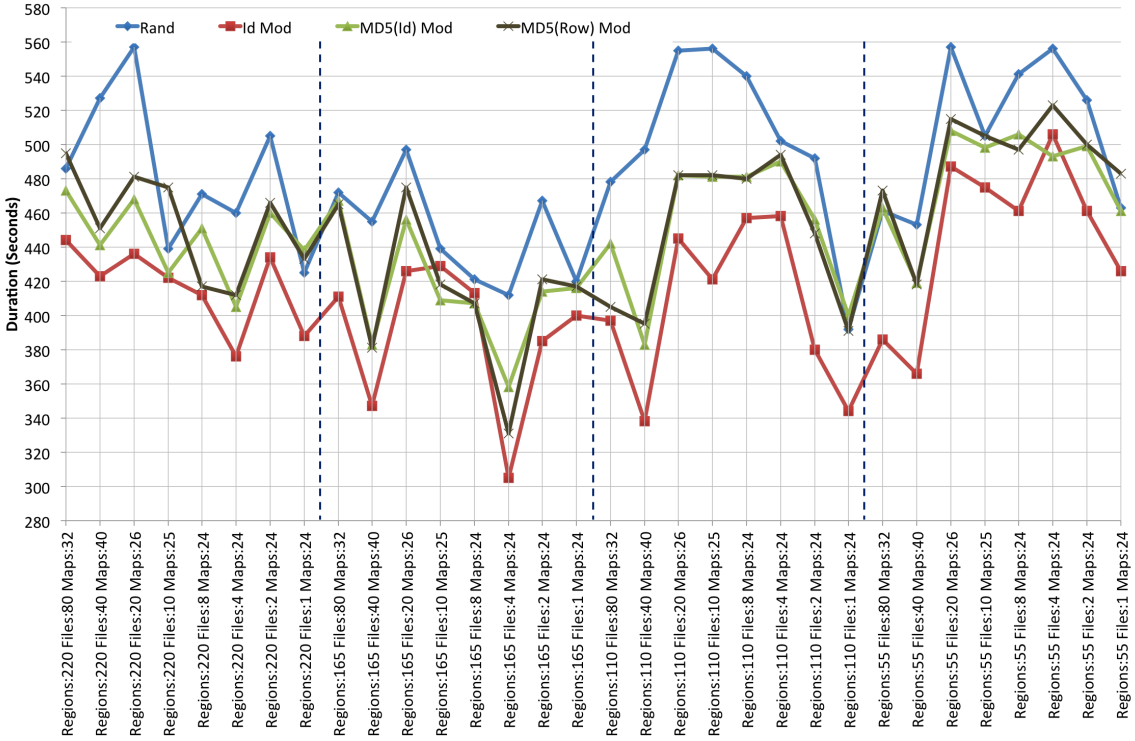
Бидејќи следната задача после вчитувањето е тестирање на паралелизацијата на алгоритмите за одбирање на атрибути, податочното множество кое со кое се вршени експериментите е означено и има голем број на колони. Имено, тоа е AAIA'14 [50] податочното множество, кое е ретка матрица од 50000 редици и околу 12000 колони со околу 0.9 % ненулти елементи. Бидејќи сепак бројот на редици е мал за да се смета за Големо податочно множество, тоа беше зголемено 80 пати хоризонтално, по што доби табела од 4 милиони редици и 12 илјади колони.

Компјутерскиот кластер кој беше употребен за експериментите е поставен локално на Факултетот за информатички науки и компјутерско инженерство при Универзитетот св. Кирил и Методиј во Скопје. Вкупно во кластерот кој се користеше за експериментите имаше 59 јазли, од кои 55 беа искористени за оваа евалуација, при што секој од нив е Intel Xeon Processor E5640 со 12M Cache, 2.66 GHz, 24 GB RAM, 4 јадра и 8 нитки. Експериментите започнаа со користење на 55 јазли и потоа бројот на јазли се намалуваше за да се испита нивното влијание врз скалабилноста.

Програмите за вчитување беа напишани во Pig Latin [164] и тие во позадина се компајлираа во MapReduce програми што парсираат, прочистуваат и трансформираат текстуални датотеки и ги запишуваат во HBase табели. За да се испитат екфектот на бројот на региони на HBase табелите врз паралелизмот, табелите беа однапред партиционирани за да имаат 1, 2, 4, 8, 16, 32, 55, 110, 165 и 220 региони. HBase автоматски ги дистрибуира регионите така што секој сервер на региони (HBase Region Server) опслужува еднаков број на региони по табела. Бројот на региони во овој експериментален дизајн го ограничува паралелизмот при запишувањето на податоците, додека паралелизмот при вчитувањето (број на мапирачки задачи) се одредува од дистрибуираноста на датотеката која се вчитува од HDFS, при шо освен големинатата на датотеката, немаме друга контрола. Во предложените дизајни на примарни клучеви делителот при пресметувањето на остатокот (модуло бројојт) е 1000000, со што се генерираат префикси во опсегот од 0 до 999999. Случајните броеви кои се користат кај случајниот префикс се во истиот опсег.

На Слика 6-12 е прикажано времето за вчитување на податочното множество во зависност од бројот на датотеки на кои е разделено, бројот на региони по HBase табела и во зависност од дизајнот на примарниот клуч. Како што може да се забележи, за сите дизајни на примарниот клуч најкраткото време на извршување е кога табелата е партиционирана на 165 ($55 \times 3 = 165$) региони, а деталната дискусија зошто тоа е така е дадена во [11]. Накратко, овој дизајн оптимално ги искористува јадрата на секој јазол, оставајќи едно слободно јадро за опслужување на оперативниот систем и три се користат за процесирање на

MapReduce задачите.

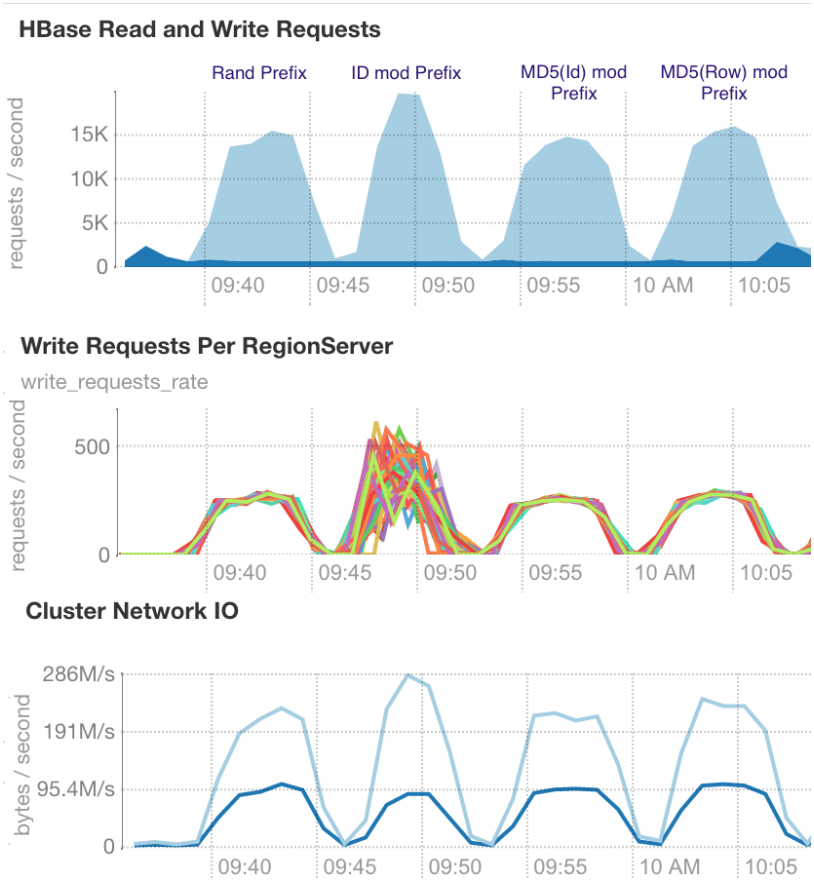


Слика 6-12: Перформанси на различните дизајни на примарни клучеви кај HBase кластер со 55 сервери на региони по број на партиции по табела и по број на HDFS датотеки.

Во однос на времето на извршување во зависност од дизајнот на примарните клучеви, од Слика. 6-12 може да се забележи дека случајниот префикс (Rand. prefix) дава најлоши резултати без разлика на бројот на HDFS датотеки во кои иницијално е зачувана. Забележително е дека префиксите со MD5 модул имаат слични перформанси (MD5(Id) Mod и MD5(Row) Mod), што покажува дека големината на аргументот на MD5 функцијата не влијае значително на времето на извршување. Ова го потврдува тврдењето дека MD5 функцијата може да се употребува за сложени клучеви кои се состојат од повеќе колони од произволен тип. Исто така се гледа дека префиксот со модул од ИД-то (ID mod) дава најдобри перформанси кај оваа конфигурација на кластерот.

Од графиците претсавени на Слика 6-13 се гледаат перформансите на кластерот при вчитување на множеството кога тоа се чува на HDFS во една датотека од 3 GB. Првиот график ги претставува бројот на барања за читање и пишување (HBase Read and Write Requests), вториот ги прикажува бројот на барања за пишување по сервер на региони (Write Requests per Region Server) и третиот го прикажува влезот-излезот на мрежата на кластерот (Cluster Network I/O). На секој од нив се дадени перформансите на различните дизајни на примарен клуч. Може да се забележи дека времето на извршување е слично (392, 344, 400 и 391

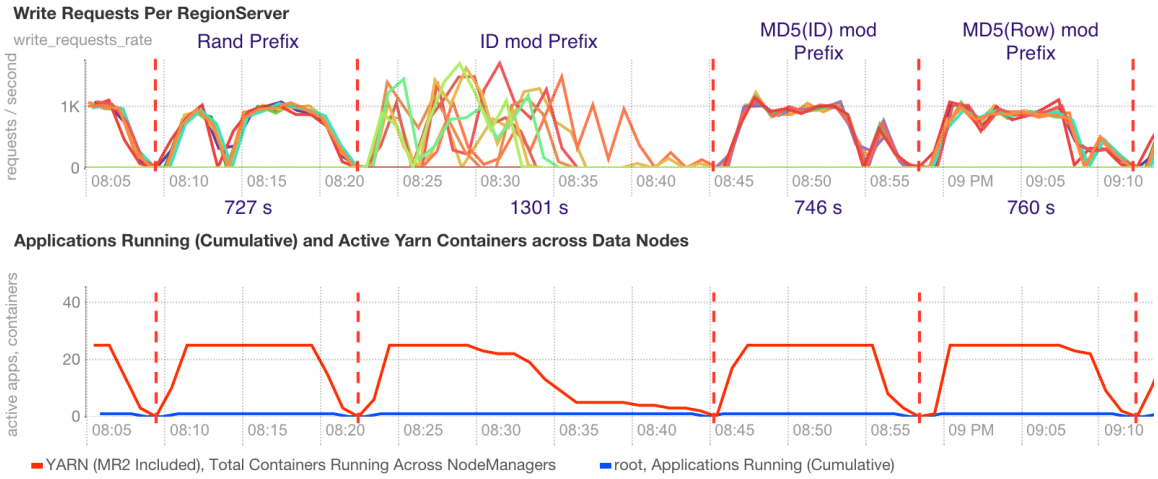
секунди). Префиксот со модул од ИД (ID modulo prefix) прави повеќе барања за читање и пишување во единица време и поголем мрежен сообраќај. Само за овој дизај на клучот активноста на серверите на региони не е рамномерна во секое време, што може да се забележи од вториот график на Слика 6-13. Поради тоа што овој дизајн генерира некогаш и неколку пати повеќе барања од другите, тој може да предизвика катастрофални последици и да предизвика пад на кластерот ако повеќе апликации работат одеднаш.



Слика 6-13: Перформанси на запишување на една датотека од 3GB во HBase табела со 220 региони дистрибуирани на 55 сервери на региони.

Друг недостаток на клучот со префикс со модул од ИД (ID modulo prefix) може да се забележи од Слика 6-14. Имено, бидејќи примарниот клуч ID е монотоно растечки број, и неговиот модул е монотоно растечки, што предизвикува само одредени региони да се активни во одредено време. Поради тоа секогаш има некои региони кои не работат ништо, поради што се зголемува времето на извршување. Со другите дизајни е постигната рамномерна распределба на оптоварувањето по региони и во текот на времето.

Во оваа фаза од студијата на случај беше евалуирана важноста на дизајнот на примарните клучеви во HBase табелите. Се покажаа предностите на однапред партиционирање на табелите со цел кластерот да има рамномерна распределба на



Слика 6-14: Перформанси на запишување во HBase табела со 8 сервери на региони и 8 региони по табела.

оптоварувањето. За да се овозможи тоа, примарниот клуч треба да има однапред позната распределба, но бидејќи тоа во општ случај не е можно, беа евалуирани дизајните на примарните клучеви предложени во Глава 5.2. Хипотезата дека со помош на MD5 хеширање може да се постигне рамномерна дистрибуција слична на случајната дистрибуција беше потврдена. Исто така беше покажано дека должината на аргументот на MD5 функцијата не влијае значително на времето на извршување. Предложениот префикс врз основа на модул од MD5 хеш вредноста овозможува пристап до случајни редици, што не е случај со слујачниот префикс. Од друга страна, се покажа дека користењето префикс со помош на модул од монотоно растечки клучеви е лоша опција заради непредвидливите перформанси и лошата распределба на оптоварувањето на јазлите. Во овие експерименти се покажува скалабилнста на предобработката на податочните множества кога се прават транформации кои зависат само од редот кој се обработува. Овој чекор на предобработка вклучува: прочистување на податоци, откривање на вредности што недостасуваат, претварање од еден во друг тип, дискретизација, генерирање на нови атрибути со помош на математички трансформации и друго.

6.9.2 Паралелна пресметка на информационата добивка

Во оваа глава се покажани резултатите од евалуацијата на паралелната имплементација за пресметување на информационата добивка предложена во [14, 16] и објаснета во Глава 5.5. За време на експериментите, кластерите не извршуваа други задачи. Поради тоа што табелите беа однапред партиционирани, паралелизмот кога се користи MapReduce и Pig Latin е исто така предефиниран. Симулирањето на помали кластери е преку дефинирање на табели со помал број на региони.

При евалуацијата се користени три кластери. Првиот, означен со *Amazon32*, работеше во облак на Amazon AWS. Вкупно имаше 32 јазли, секој од нив на m1.xlarge инстанца со 15GB RAM и 4 јадра. Од 32 јазли, ефективно само 8 ги извршуваа сите потербни Hadoop сервиси одеднаш (HBase Region Servers, HDFS Data Nodes и YARN) поради што е направена анализа на забрзувањето кога се користат најмногу 8 јазли. Останатите јазли во кластерот извршуваа само некои од трите потребни сервиси.

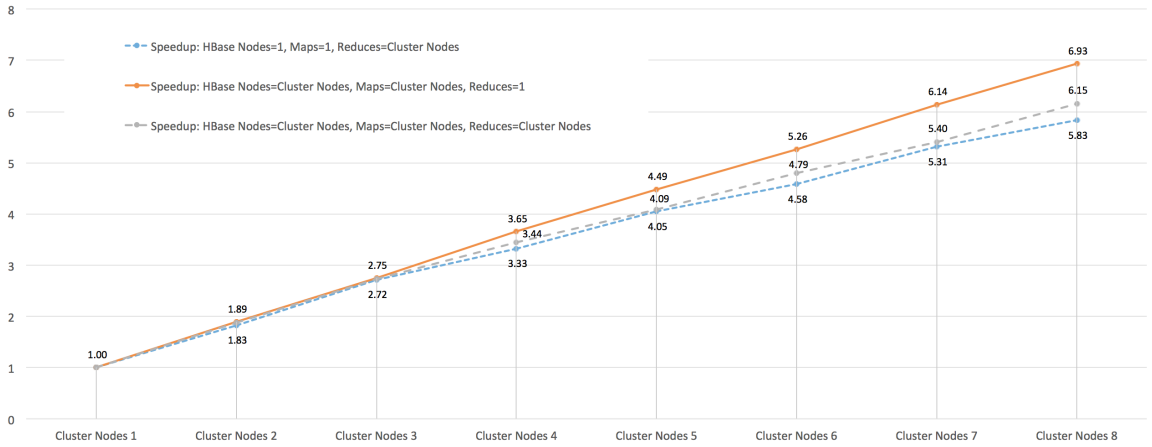
Вториот кластер, означен со *FCSE24*, беше поставен локално на Факултетот за информатички науки и компјутерско инженерство при Универзитетот св. Кирил и Методиј во Скопје. Вкупно во кластерот кој се користеше за експериментите имаше 24 јазли при што секој од нив е Intel Xeon Processor E5640 со 12M Cache, 2.66 GHz, 24 GB RAM, 4 јадра и 8 нитки. Експериментите започнаа со користење на 24 јазли и потоа бројот на јазли се намалуваше за да се испита нивното влијание врз скалабилноста. Третиот кластер ги имаше истите карактеристики како и вториот, со таа разлика што 54 јазли ги извршуваа сите потребни Hadoop сервиси.

Со оглед на тоа што чекорот од пресметката на информационата вредност опишан во Глава 5.5.2 е најсложен, ефектот од паралелизацијата е најголем кој овој чекор. Паралелизацијата на останатите чекори има мало влијание, па резултатите од неа не се опишани овде, но сепак се објавени во [14, 16].

Бидејќи главниот дел од пресметката се прави во мапирачките задачи од MapReduce или Spark пресметките, евалуирани се повеќе конфигурации на кластерите.

Најпрво, користејќи го оригиналното податочно множество AAIA'14 и кластерот Amazon32, кој се извршуваше во Амазон облакот, беше направена анализа на влијанието на бројот на јазли, бројот на мапирачки задачи и бројот на редуцирачки задачи. Тестирани се три варијанти на искористување на кластерот: сите достапни јазли да се користат за извршување на мапирачки задачи и да се извршува само една редуцирачка задача; сите достапни јазли да се користат за извршување и на мапирачки и на редуцирачки задачи; и само еден јазол да се користи за извршување на една мапирачка задача и сите достапни јазли да се користат за извршување на редуцирачки задачи. Забрзувањето во однос на извршувањето кога се користи само еден јазол со по една мапирачка и редуцирачка задача (т.е. секвенцијално ивршување) е дадено на Слика 6-15. Заклучокот е дека најдобро е да се користи колку што е можно повеќе мапирачки и само една редуцирачка задача. Вкупното извршување на кластерот со еден јазол е 4732 секунди, додека најбрзото решение користејќи 8 јазли е за 693 секунди со што се постигнува забрзување од 6.83.

За да се потврдат овие сознанија експериментите беа повторени на локалниот FCSE24 кластер. Тестирани се 1, 3, 5, 7 или 9 редуцирачки задачи во зависност од

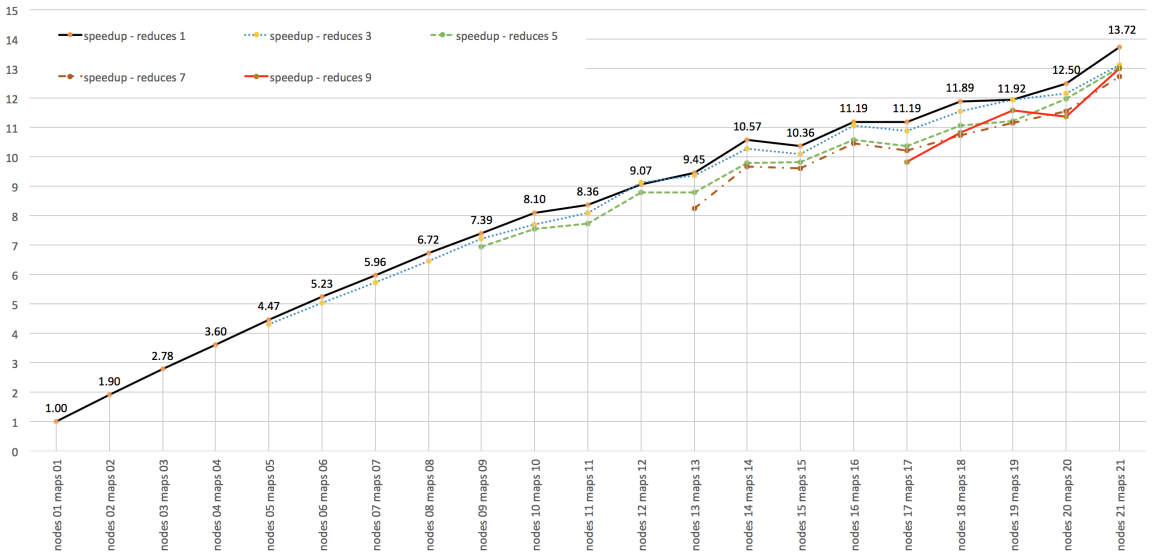


Слика 6-15: Забрзување во зависност од бројот на активни јазли и бројот на мапирачки и редуцирачки задачи на кластерот Amazon32

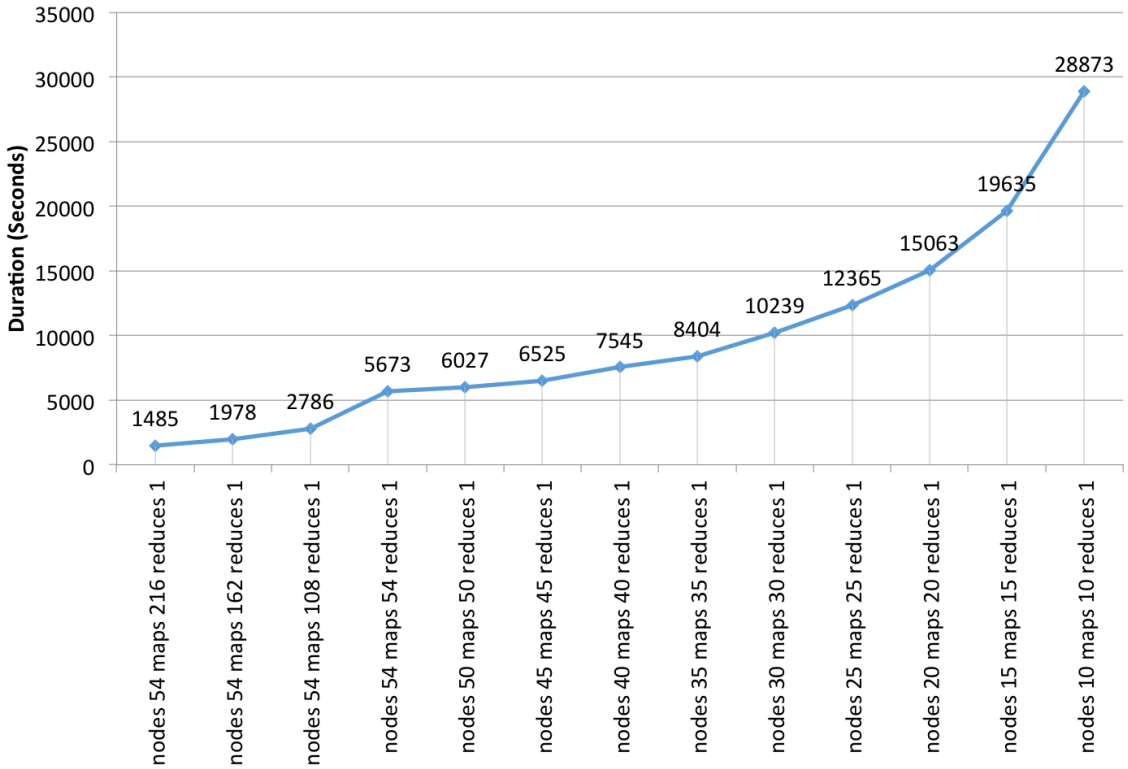
бројот на достапни јазли. Резултатите од овој експеримент се дадени на Слика 6-16 и покажуваат дека наголемо забрзување се постигнува секогаш кога се користи една редуцирачка задача. Секвенцијалното извршување на кластер со само еден јазол е 3637 секунди, додека најбрзото извршување е 265 секунди, постигнувајќи забрзување од 13.72 пати.

За да се утврди скалабилноста на решението, беше искористено 80-пати зголеменото податочно множество користено во експериментите во Глава 6.9.1. Резултатите од оваа евалуација кога се користат 54 јазли се дадени на Слика 6-17. Кога се извршуваат два, три или четири пати повеќе мапирачки задачи (108, 162, 216) од бројот на јазли во кластерот (54) со цел да се искористат јадрата на секој јазол пооптимално, перформансите се подобруваат. Вкупното време на извршување на овој чекор на кластер со само десет јазли е 28873 секунди, додека најбрзото извршување е за 1485 секунди кога се користат 216 мапирачки задачи и 54 јазли.

Со оваа студија на случај се покажува дека паралелното пресметување на информационата добивка е скалабилно на Големи Податоци. Со партиционирањето на табелите при нивното креирање е овозможено контролирање на нивото на паралелизам. Со извршување на алгоритамот на кластери ставени производство во облак и локално е покажана портабилноста на имплементацијата. За валидација на точноста на имплементацијата е направена споредба на пресметаната информациона добивка на атрибутите со други софтверски пакети.



Слика 6-16: Забрзување во зависност од бројот на активни јазли и бројот на мапирачки и редуцирачки задачи на кластерот FCSE24



Слика 6-17: Време на извршување за чекорот со броењето на инстанци во зависност од конфигурацијата на кластерот за податочно множество со 4 милиони инстанци и 12 илјади атрибути

Глава 7

Заклучоци и идна работа

Во оваа докторска дисертација главна теза беше дека примената на паралелно и дистрибирано пресметување овозможува забрзување и скалабилност на алгоритмите за предобработка и класификација на податоци. За потврда на ова тврдење беа имплементирани алгоритми за инженерство и одбирање на атрибути во паралелна околина и истите беа емпириски тестирани во повеќе студии на случај.

Во Глава 3 беше направен преглед на техниките за инженерство на атрибути од временски серии и трансформација на номиналните атрибути во нумерички кои се достапни во литературата. Потоа беше опишана предложената генеричка методологија за инженерство на атрибути од временски серии кои може да потекнуваат од произволни извори. Користејќи различни техники кои користат статистички метрики, хистограми, перцентили, брза Фуриева трансформација, корелација и интерполација, како и со генерирање на нови временски серии, предложената методологија овозможува генерирање информативни атрибути кои робустно го опишуваат проблемот од интерес. Исто така, детално беше објаснета предложената техника за трансформација на номинални и категоријални атрибути во нумерички со помош на параметарот тежина на фактите. Дополнително, беше предложено проширување на оваа трансформација со што е овозможена нејзина употреба и кај повеќекласни проблеми.

Во Глава 4 беа резимирани карактеристиките на различните типови на алгоритми за одбирање на атрибутите. Потоа беа предложени неколку хибридни методи кои ги надминуваат недостатоците на рангирачките и обвиткувачките методи. Најпрво, беше предложен хибриден метод кој прави комбинација од филтрирачки и корелациски методи со цел брзо да се намали множеството на атрибути и потоа полесно да се определат меѓузависните атрибути. Потоа беше предложен хибриден метод за брзо откривање на атрибути кои се осетливи на концептуални отстапувања. Веројатносната дистрибуција на ваквите атрибути е променлива во текот на времето, поради што изградените модели може да

станат неупотребливи бидејќи биле претренирани и соодветни само за тренинг множеството. Во оваа глава беше предложен уште еден хибриден метод за брзо пребарување на просторот на множества на атрибути кој прво ги рангира атрибутите и потоа со помош на алчно диверзифицирано пребарување напред-назад наоѓа мали подмножества на информативни и независни атрибути. На крајот на Глава 4 беше предложен уште еден хибриден метод за одбирање на атрибутите кој ги комбинира претходните два методи во еден.

Во Глава 5 беа опишани предложените пристапи за паралелизација на пресметките со користење на кластери од компјутери и искористување на повеќето јадра на процесорите. Прво беа опишани најпопуларните платформи и концепти за дистрибуирано и паралелно пресметување: Hadoop, MapReduce, Spark, HDFS и HBase. Потоа беше направен преглед на литературата поврзана со паралелизација на секвенцијални алгоритми со помош на овие технологии. Беа опишани предложените дизајни на примерни клучеви на HBase табели со кои се овозможува рамномерно оптоварување на јазлите во кластерот користејќи паралелизам на податоците. Со истиот пристап на паралелизам беше предложена имплементација за паралелното генерирање на атрибути со помош на агрегациски функции користејќи Spark. Во оваа глава беше предложен начин за паралелизација на пресметката на информационата добивка со помош на Pig Latin, MapReduce или Spark. Исто така, беше предложен и начин за паралелизација на обвиткувачките методи за одбирање на атрибутите преку паралелна евалуација на различни множества на атрибути.

Во Глава 6 беа опишани повеќе студии на случај во кои успешно беа употребени предложените методи за инженерство и одбирање на атрибути. Информационата добивка беше применета за дискретизација и пресметка на оптимални прагови за поделба на интервалите на ризик во медицината. Со методологијата за инженерство на атрибути од временски серии беше овозможено: препознавање на активности на пожарникари; предвидување на опасни концентрации на метан во рудници за јаглен; предвидување на сеизмички активности во рудници за јаглен; препознавање на активности во системи за амбиентално потпомогнато живеење; и беше овозможено потиснување на лажните аларми на мониторите во одделите за интензивна нега. Скалабилноста на паралелизираните алгоритми беше демонстрирана со помош на трансформација, агрегација и одбирање на атрибути кај Големи Податоци. Притоа, беа споредени постигнатите резултати во однос на останатите актуелни пристапи и беа анализирани постигнатите забрзувања со помош на паралелизацијата. Исто така, се дискутираше нивната значајност и придонес во облатите во кои беа применети.

Со евалуацијата на предложената методологија за инженерство на атрибути од временски серии беше покажано дека не се потребни специјализирани и рачно одбрани атрибути за да се пронајдат робустни и информативни атрибути. Напротив, со неа се добиваат подобри предиктивни перформанси и се прави значителна заштеда на времето потребно за дизајн, архитектура и имплементација

на системите. Преку евалуација низ повеќе студии на случај, беше потврдена првата подхипотеза која гласеше: автоматско инженерство на атрибути од произволни временски серии и номинални атрибути се овозможува генерирање на информативни атрибути кои овозможуваат робустна класификација.

Податочните множества со илјадници атрибути генерирани со помош на методологијата за инженерство на атрибути би биле премногу големи за директна примена на многу алгоритми за класификација. Во тој контекст, придобивките од предложените хибридни методи за одбирање на атрибути се истакнаа бидејќи овозможија брзо одбирање мали множества на атрибути кои воедно ги забрзуваат алгоритмите за машинско учење и ја зголемуваат портабилноста поради намалената мемориска и пресметковна комплексност. Со ова се потврди втората подхипотеза која тврди дека со помош на хибридни методи за одбирање на атрибути може ефикасно да се добијат мали множества од информативни атрибути.

Методологијата за инженерство на атрибути од временски серии и хибридните методи за одбирање на атрибути беа искомбинирани во автоматизиран процес кој не бара рачна интервенција за да изгради оптимални предиктивни модели со мали но многу информативни множества на атрибути. Во споредба со други пристапи во литратурата кои се специјализирани за одредена намена, предложената методологија даде помали множества на атрибути со кои се забрзаа алгоритмите за машинско учење и притоа се подобрија предиктивните перформанси.

Со паралелизацијата на повеќе алгоритми за предобработка и класификација се овозможи нивна примена на Големи Податоци. Третата подхипотеза тврди дека со користење на генерички платформи и техники за дистрибуирано пресметување на компјутерски кластери се овозможува паралелизација на алгоритмите за предобработка и класификација на податоците. Оваа хипотеза беше потврдена преку неколку студии на случај во кои беа употребени технологиите Hadoop, HDFS, HBase, Spark, Pig Latin и MapReduce за паралелизација на инженерството на атрибути, дистрибуираното запишување на Големи Податоци и паралелно одбирање на атрибути.

Пристапот за паралелизација на алгоритмите преку паралелизација на податоците (data parallelism, анг.) е соодветен и многу ефикасен за најголем дел од алгоритмите за предобработка, особено за инженерство на атрибути, овозможувајќи одлична скалабилност. Истиот пристап е соодветен и за рангирачките алгоритми за одбирање на атрибути. Сепак за некои од обвиткувачките и хибридните методи овој пристап не е секогаш применлив. Уште поголем предизвик е паралелизацијата на алгоритмите за машинско учење и класификација со помош на паралелизам на податоците.

За некои алгоритми за машинско учење можеби паралелизамот на моделите (model parallelism, анг.) би бил посоодветен. На пример, пристапите за паралелизација на градењето модели со машини со носечки вектори користат паралелизам

на податоците. Како резултат на тоа, само дел од податочното множество е достапно на секој јазол. Поради ова, локалните модели треба да се агрегираат за да се направи конечно предвидување, што од друга страна предизвикува намалување на перформансите во однос на случајот кога сите податоци се достапни за учење и градење на моделите. Вакви примери има многу и допрва останува да се утврди дали градењето на моделите може да се паралелизира без деградација на предиктивните перформанси. На овој начин сеуште само малку алгоритми се паралелизирани и ова претставува интересна можност за идна работа.

Референци

[1] E. Zdravevski, P. Lameski, V. Trajkovik, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, and N. Garcia, "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering," *IEEE Access*, vol. 5, pp. 5262–5280, 2017.

[2] E. Zdravevski, B. Risteska Stojkoska, M. Standl, and H. Schulz, "Automatic machine-learning based identification of jogging periods from accelerometer measurements of adolescents under field conditions," *PLOS ONE*, vol. 12, no. 9, pp. 1–28, 09 2017, accepted for publication. [Online]. Available: https://doi.org/10.1371/journal.pone.0184216

[3] P. Lameski, E. Zdravevski, S. Koceski, A. Kulakov, and V. Trajkovik, "Suppression of intensive care unit false alarms based on the arterial blood pressure signal," *IEEE Access*, vol. PP, no. 99, pp. 1–7, 2017, in press.

[4] K. Drusany Starič, P. Bukovec, K. Jakopič, E. Zdravevski, V. Trajkovik, and A. Lukanović, "Can we predict obstetric anal sphincter injury?" *European Journal of Obstetrics & Gynecology and Reproductive Biology*, vol. 210, pp. 196–200, mar 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0301211516310910

[5] P. Lameski, E. Zdravevski, A. Kulakov, and V. Trajkovik, "Cloud based architecture for automated weed control," in *Proceedings of the 17th IEEE International Conference on Smart Technologies IEEE EUROCON 2017*. IEEE, July 2017, in press.

[6] A. Zdravevska, A. Dimitrievski, P. Lameski, E. Zdravevski, and V. Trajkovik, "Cloud- based privacy preserving recognition of complex activities for ambient assisted living in smart homes," in *Proceedings of the 17th IEEE International Conference on Smart Technologies IEEE EUROCON 2017*. IEEE, July 2017, in press.

[7] E. Zdravevski, P. Lameski, and A. Kulakov, "Automatic feature engineering for prediction of dangerous seismic activities in coal mines," in *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 6. IEEE, Sept 2016, p. 251–254. [Online]. Available: https://fedcsis.org/proceedings/2016/pliks/152.pdf

[8] A. Dimitrievski, E. Zdravevski, P. Lameski, and V. Trajkovik, "A survey of ambient assisted living systems: Challenges and opportunities," in *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sept 2016, pp. 49–53.

[9] A. Dimitrievski, E. Zdravevski, P. Lameski, and V. Trajkovik, "Towards application of non-invasive environmental sensors for risks and activity detection," in *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sept 2016, pp. 27–33.

[10] E. Zdravevski, P. Lameski, A. Kulakov, and V. Trajkovik, "Performance comparison of random forests and extremely randomized trees," in *Proceedings of the 13th Conference for Informatics and Information Technology (CIIT 2016)*. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, April 2016.

[11] E. Zdravevski, P. Lameski, and A. Kulakov, "Row Key Designs of NoSQL Database Tables and Their Impact on Write Performance," in *Proceedings - 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016*. IEEE, feb 2016, pp. 10–17. [Online]. Available: http://ieeexplore.ieee.org/document/7445308/

[12] P. Lameski, E. Zdravevski, R. Mingov, and A. Kulakov, "SVM parameter tuning with grid search and its impact on reduction of model over-fitting," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing: 15th International Conference, RSFDGrC 2015, Tianjin, China, November 20-23, 2015, Proceedings*, Y. Yao, Q. Hu, H. Yu, and J. W. Grzymala-Busse, Eds. Cham, Switzerland: Springer International Publishing, 2015, pp. 464–474.

[13] E. Zdravevski, P. Lameski, A. Kulakov, and S. Kalajdziski, "Transformation of nominal features into numeric in supervised multi-class problems based on the weight of evidence parameter," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015, pp. 169–179. [Online]. Available: http://dx.doi.org/10.15439/2015F90

[14] E. Zdravevski, P. Lameski, A. Kulakov, S. Filiposka, D. Trajanov, and B. Jakimovski, "Parallel computation of information gain using hadoop and mapreduce," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015, pp. 181–192. [Online]. Available: http://dx.doi.org/10.15439/2015F89

[15] E. Zdravevski, P. Lameski, R. Mingov, A. Kulakov, and D. Gjorgjevikj, "Robust histogram-based feature engineering of time series data," in *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, Sept 2015, pp. 381–388. [Online]. Available: http://dx.doi.org/10.15439/2015F420

[16] E. Zdravevski, P. Lameski, A. Kulakov, B. Jakimovski, S. Filiposka, and D. Trajanov, "Feature ranking based on information gain for large classification problems with mapreduce," in *Proceedings of the 9th IEEE International Conference on Big Data Science and Engineering*. IEEE Computer Society Conference Publishing, August 2015, pp. 186–191.

[17] E. Zdravevski, P. Lameski, A. Kulakov, S. Filiposka, and D. Trajanov, "Simplifying parallel implementation of algorithms on hadoop with pig latin," in *Proceedings of the 12th Conference for Informatics and Information Technology (CIIT 2015)*. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, April 2015.

[18] E. Zdravevski, P. Lameski, A. Kulakov, and D. Gjorgjevikj, "Feature selection and allocation to diverse subsets for multi-label learning problems with large datasets," in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 2. IEEE, 2014, pp. 387–394. [Online]. Available: http://dx.doi.org/10.15439/2014F500

[19] E. Zdravevski, P. Lameski, and A. Kulakov, "Advanced transformations for nominal and categorical data into numeric data in supervised learning problems," in *Proceedings of the 10th Conference for Informatics and Information Technology (CIIT 2013)*, I. Mishkovski and S. Ristov, Eds. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, 2013, pp. 142–146.

[20] E. Zdravevski, P. Lameski, and A. Kulakov, "Towards a general technique for transformation of nominal features into numeric features in supervised learning," in *Proceedings of the 9th Conference for Informatics and Information Technology (CIIT 2012)*. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, April 2012.

[21] E. Zdravevski, P. Lameski, and A. Kulakov, "Weight of evidence as a tool for attribute transformation in the preprocessing stage of supervised learning algorithms," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 181–188.

[22] E. Zdravevski, A. Kulakov, S. Kalajdziski, and D. Davcev, "Probabilistic predictions of ensemble of classifiers combined with dynamically weighted majority vote," in *Proceedings of Artificial Intelligence and Applications 2011 (AIA 2011)*. IASTED, February 2011, pp. 236–244.

[23] P. Lameski, E. Zdravevski, V. Trajkovik, and A. Kulakov, *Weed Detection Dataset with RGB Images Taken Under Variable Light Conditions*. Cham: Springer International Publishing, 2017, pp. 112–119, accepted for publication. [Online]. Available: https://doi.org/10.1007/978-3-319-67597-8_11

[24] A. Alla, E. Zdravevski, and V. Trajkovik, "Framework for aiding surveys by natural language processing," in *2017 ICT Innovations web proceedings*, Sept 2017, accepted for publication.

[25] R. Mingov, E. Zdravevski, and P. Lameski, "Application of russian language phonemics to generate macedonian speech recognition model using sphinx," *ICT Innovations 2016, Web Proceedings*, 2016.

[26] P. Lameski, E. Zdravevski, and A. Kulakov, "Unsupervised weed detection in spinach seedling organic farms," in *Proceedings of the 24th International Electrotechnical and Computer Science Conference ERK 2015*, September 2015.

[27] P. Lameski, E. Zdravevski, A. Kulakov, and D. Gjorgjevik, "Plant species detection based on leaf contours," in *Proceedings of the 12th Conference for Informatics and Information Technology (CIIT 2015)*. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, April 2015.

[28] P. Lameski, D. Kulakov, E. Zdravevski, and A. Kulakov, "Tumor detection in manually selected regions of mri images," *ICT Innovations 2014, Web Proceedings ISSN 1857-7288*, pp. 183–190, 2014.

[29] B. Dikovski, P. Lameski, E. Zdravevski, and A. Kulakov, "Structure from motion obtained from low quality images in indoor environment," in *Proceedings of the 10th Conference for Informatics and Information Technology (CIIT 2013)*. Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, 2013.

[30] P. Lameski, E. Zdravevski, R. Mingov, and A. Kulakov, "Comparison of local image descriptors for plant identification from leaf image," in *Proceedings of the 2013 ICMER Conference*, 2013.

[31] M. Angelovski, P. Lameski, E. Zdravevski, and A. Kulakov, "Application of bci technology for color prediction using brainwaves," *ICT Innovations 2012, Web Proceedings ISSN 1857-7288*, p. 253, 2012.

[32] P. Lameski, E. Zdravevski, A. Kulakov, and D. Davcev, "Architecture for wireless sensor and actor networks control and data acquisition," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011, pp. 1–3.

[33] H. Chen, R. Chiang, and V. Storey, "Business intelligence and analytics: From big data to big impact," *MIS Quarterly: Management Information Systems*, vol. 36, no. 4, pp. 1165–1188, 2012.

[34] A. McAfee, E. Brynjolfsson, D. Boyd, K. Crawford, and S. Lohr, "Critical Questions for Big Data," *Information, Communication & Society*, vol. 15, no. 5, pp. 1–5, 2012.

[35] C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, aug 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0020025514000346

[36] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," *IEEE Access*, vol. 2,

pp. 652–687, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6842585/

[37] C. Dobre, F. Xhafa, and F. Xhafa, "Parallel Programming Paradigms and Frameworks in Big Data Era," *International Journal of Parallel Programming manuscript*, 2014. [Online]. Available: http://cipsm.hpc.pub.ro/Joomla/articles/2013{_}17.pdf

[38] H. Wang, Z. Xu, H. Fujita, and S. Liu, "Towards felicitous decision making: An overview on challenges and trends of Big Data," *Information Sciences*, vol. 367-368, pp. 747–765, nov 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0020025516304868

[39] C. Shearer, "The crisp-dm model: the new blueprint for data mining," *Journal of Data Warehousing*, vol. 5, no. 4, pp. 13–19, 2000.

[40] H. Leutheuser, D. Schuldhaus, and B. M. Eskofier, "Hierarchical, multi-sensor based classification of daily life activities: Comparison with state-of-the-art algorithms using a benchmark dataset," *PLoS ONE*, vol. 8, no. 10, p. e75196, 10 2013. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0075196

[41] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors (Basel, Switzerland)*, vol. 16, no. 4, p. 426, 04 2016. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4850940/

[42] E. Fortune, V. Lugade, S. Amin, and K. Kaufman, "Step detection using multi- versus single tri-axial accelerometer-based systems," *Physiological measurement*, vol. 36, no. 12, pp. 2519–2535, 12 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4838513/

[43] M. Munoz-Organero and A. Lotfi, "Human movement recognition based on the stochastic characterisation of acceleration data," *Sensors*, vol. 16, no. 9, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/9/1464

[44] T. Bastian, A. Maire, J. Dugas, A. Ataya, C. Villars, F. Gris, E. Perrin, Y. Caritu, M. Doron, S. Blanc, P. Jallon, and C. Simon, "Automatic identification of physical activity types and sedentary behaviors from triaxial accelerometer: laboratory-based calibrations are not enough," *Journal of Applied Physiology*, vol. 118, no. 6, pp. 716–722, 2015. [Online]. Available: http://jap.physiology.org/content/118/6/716

[45] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *BioMedical Engineering OnLine*, vol. 14, no. Suppl 2, pp. S6–S6, 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4547155/

[46] S. Chernbumroong, S. Cang, A. Atkins, and H. Yu, "Elderly activities recognition and classification for applications in assisted living," *Expert Syst. Appl.*, vol. 40, no. 5, pp. 1662–1674, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2012.09.004

[47] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors (Basel, Switzerland)*, vol. 14, no. 6, pp. 10 146–10 176, 06 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4118351/

[48] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy efficient smartphone-based activity recognition using fixed-point arithmetic," *Journal of Universal Computer Science*, vol. 19, no. 9, pp. 1295–1314, may 2013. [Online]. Available: http://www.jucs.org/jucs_19_9/energy_efficient_smartphone_based

[49] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754 – 767, 2016. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0925231215010930

[50] A. Janusz, A. Krasuski, S. Stawicki, M. Rosiak, D. Slezak, and H. S. Nguyen, "Key risk factors for polish state fire service: A data mining competition at knowledge pit," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, Sept 2014, pp. 345–354.

[51] M. Meina, A. Janusz, K. Rykaczewski, D. Slezak, B. Celmer, and A. Krasuski, "Tagging firefighter activities at the emergency scene: Summary of aaia'15 data mining competition at knowledge pit," in *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, Sept 2015, pp. 367–373.

[52] A. Janusz, M. Sikora, Ł. Wróbel, and D. Ślezak, "Predicting Dangerous Seismic Events: AAIA16 Data Mining Challenge," in *Proceedings of FedCSIS 2016*. IEEE, 2016, in print September 2016.

[53] A. Janusz, M. Sikora, Ł. Wróbel, S. Stawicki, M. Grzegorowski, P. Wojtas, and D. Ślęzak, "Mining data from coal mines: IJCRS'15 data challenge," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, ser. Lecture Notes in Computer Science, Y. Yao, Q. Hu, H. Yu, and J. W. Grzymala-Busse, Eds. Springer International Publishing, 2015, vol. 9437, pp. 429–438. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-25783-9_38

[54] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143. [Online]. Available: http://dl.acm.org/citation.cfm?id=1643031.1643047

[55] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[56] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *The Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.

[57] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1005336

[58] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data Mining Techniques for the Life Sciences*, ser. Methods in Molecular Biology, O. Carugo and F. Eisenhaber, Eds. Humana Press, 2010, vol. 609, pp. 223–239. [Online]. Available: http://dx.doi.org/10.1007/978-1-60327-241-4_13

[59] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[60] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[61] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10994-006-6226-1

[62] D. T. Larose, "k-nearest neighbor algorithm," *Discovering Knowledge in Data: An Introduction to Data Mining*, pp. 90–106, 2005.

[63] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.

[64] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, ser. Wiley Series in Probability and Statistics. United States: John Wiley & Sons, 2013, vol. 398.

[65] D. Pyle, *Data preparation for data mining*. San Francisco, Calif: Morgan Kaufmann Publishers, 1999.

[66] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[67] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, and Z. Chen, "Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 3, pp. 320–333, March 2006.

[68] M. Munk, J. Kapusta, and P. Švec, "Data preprocessing evaluation for web log mining: reconstruction of activities of a web visitor," *Procedia Computer Science*, vol. 1, no. 1, pp. 2273 – 2280, 2010, {ICCS} 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050910002565

[69] N. M. Nawi, W. H. Atomi, and M. Rehman, "The effect of data pre-processing on optimized training of artificial neural networks," *Procedia Technology*,

vol. 11, no. 0, pp. 32 – 39, 2013, 4th International Conference on Electrical Engineering and Informatics, {ICEEI} 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212017313003137

[70] M. Hofmann and R. Klinkenberg, Eds., *RapidMiner: data mining use cases and business analytics applications*, ser. Chapman & Hall/CRC data mining and knowledge discovery series. Boca Raton: CRC Press, 2014, no. 33.

[71] I. H. Witten, *Data mining: practical machine learning tools and techniques*, 2nd ed., ser. Morgan Kaufmann series in data management systems. Amsterdam ; Boston, MA: Morgan Kaufman, 2005.

[72] T. W. Miller, *Modeling techniques in predictive analytics: business problems and solutions with R.* Upper Saddle River, New Jersey: Pearson Education, Inc, 2014.

[73] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, 2004. [Online]. Available: http://dx.doi.org/10.1108/00220410410560582

[74] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, pp. 13:1–13:37, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1361684.1361686

[75] T. chung Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164 – 181, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197610001727

[76] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 12:1–12:34, Dec. 2012. [Online]. Available: http://doi.acm.org/10.1145/2379776.2379788

[77] B. Hu, Y. Chen, and E. Keogh, "Classification of streaming time series under more realistic assumptions," *Data Mining and Knowledge Discovery*, pp. 1–35, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10618-015-0415-0

[78] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/4/6474

[79] L. Bao and S. S. Intille, *Activity Recognition from User-Annotated Acceleration Data.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–17. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24646-6_1

[80] S. J. Preece*, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 3, pp. 871–879, March 2009.

[81] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, Third 2013.

[82] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, Jan 2006.

[83] A. M. Khan, A. Tufail, A. M. Khattak, and T. H. Laine, "Activity recognition on smartphones via sensor-fusion and kda-based svms," *International Journal of Distributed Sensor Networks*, vol. 10, no. 5, p. 503291, 2014. [Online]. Available: http://dx.doi.org/10.1155/2014/503291

[84] J. Lasek and M. Gagolewski, "The winning solution to the aaia'15 data mining competition: Tagging firefighter activities at a fire scene," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015, pp. 375–380. [Online]. Available: http://dx.doi.org/10.15439/2015F418

[85] A. G. Barnett and A. J. Dobson, *Analysing Seasonal Health Data.* Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010.

[86] A. Janusz, M. Sikora, Ł. Wróbel, S. Stawicki, M. Grzegorowski, P. Wojtas, and D. Ślęzak, *Mining Data from Coal Mines: IJCRS'15 Data Challenge.* Cham: Springer International Publishing, 2015, pp. 429–438.

[87] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[88] P. Siirtola and J. Röning, "Recognizing human activities user-independently on smartphones based on accelerometer data," *International Journal of Artificial Intelligence and Interactive Multimedia*, vol. 1, no. 5, pp. 38–45, 2012.

[89] H. Martín, A. M. Bernardos, J. Iglesias, and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," *Personal and Ubiquitous Computing*, vol. 17, no. 4, pp. 675–695, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00779-012-0515-4

[90] A. Zagorecki, "A versatile approach to classification of multivariate time series data," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 5. IEEE, 2015, pp. 407–410. [Online]. Available: http://dx.doi.org/10.15439/2015F419

[91] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms.* Springer-Verlag, 1993, pp. 69–84.

[92] R. Anderson, *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation.* Oxford: Oxford University Press, 2007.

[93] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD*

Explor. Newsl., vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[94] E. Tuv and G. Runger, "Scoring levels of categorical variables with heterogeneous data," *Intelligent Systems, IEEE*, vol. 19, no. 2, pp. 14–19, Mar 2004.

[95] D. B. Suits, "Use of dummy variables in regression equations," *Journal of the American Statistical Association*, vol. 52, no. 280, 1957.

[96] M. A. Hardy, *Regression with dummy variables*, ser. Sage university papers series. Newbury Park: Sage Publications, 1993, no. no. 07-093.

[97] D. W. Goodall, "A new similarity index based on probability," *Biometrics*, vol. 22, no. 4, pp. pp. 882–907, 1966. [Online]. Available: http://www.jstor.org/stable/2528080

[98] C. Li and G. Biswas, "Unsupervised learning with mixed numeric and nominal data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 4, pp. 673–690, Jul 2002.

[99] I. J. Good, *Probability and the Weighing of Evidence*. C. Griffin and Co., London, UK, 1950.

[100] E. P. Smith, I. Lipkovich, and K. Ye, "Weight-of-evidence (woe): Quantitative estimation of probability of impairment for individual and multiple lines of evidence," *Human and Ecological Risk Assessment: An International Journal*, vol. 8, no. 7, pp. 1585–1596, 2002. [Online]. Available: http://dx.doi.org/10.1080/20028091057493

[101] H. Almuallim and T. G. Dietterich, "Learning with many irrelevant features," in *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'91. AAAI Press, 1991, pp. 547–552. [Online]. Available: http://dl.acm.org/citation.cfm?id=1865756.1865761

[102] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 245 – 271, 1997, relevance. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370297000635

[103] P. Langley, *Elements of machine learning*. San Francisco, Calif: Morgan Kaufmann, 1996.

[104] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1-2, pp. 273–324, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0004-3702(97)00043-X

[105] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[106] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional word clusters vs. words for text categorization," *J. Mach. Learn. Res.*, vol. 3, pp.

1183–1208, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944969

[107] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *NIPS*, vol. 12, 2000, pp. 668–674.

[108] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944974

[109] C. Lee and G. G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Inf. Process. Manage.*, vol. 42, no. 1, pp. 155–165, Jan. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ipm.2004.08.006

[110] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.

[111] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the Ninth International Workshop on Machine Learning*, ser. ML92. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, pp. 249–256. [Online]. Available: http://dl.acm.org/citation.cfm?id=141975.142034

[112] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *Machine Learning: ECML-94*, ser. Lecture Notes in Computer Science, F. Bergadano and L. De Raedt, Eds. Springer Berlin Heidelberg, 1994, vol. 784, pp. 171–182. [Online]. Available: http://dx.doi.org/10.1007/3-540-57868-4_57

[113] T. Jebara and T. Jaakkola, "Feature selection and dualities in maximum entropy discrimination," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 291–300. [Online]. Available: http://dl.acm.org/citation.cfm?id=2073946.2073981

[114] A. Vehtari and J. Lampinen, "Bayesian input variable selection using posterior probabilities and expected utilities," *Report B31*, 2002.

[115] A. Y. Ng and M. I. Jordan, "Convergence rates of the voting gibbs classifier, with application to bayesian feature selection," in *In 18th International Conference on Machine Learning*. Morgan Kaufmann, 2001.

[116] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection a Data Mining Perspective*. Boston, MA: Springer US, 1998. [Online]. Available: http://dx.doi.org/10.1007/978-1-4615-5725-8

[117] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[118] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *ICML*, vol. 3, 2003, pp. 856–863.

[119] T. M. Cover and J. A. Thomas, *Elements of information theory*, 2nd ed. Hoboken, NJ: Wiley, 2006.

[120] C. Shang, M. Li, S. Feng, Q. Jiang, and J. Fan, "Feature selection via maximizing global information gain for text classification," *Knowledge-Based Systems*, vol. 54, no. 0, pp. 298 – 309, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705113003067

[121] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, 1993, pp. 1022–1029.

[122] J. Dougherty, R. Kohavi, M. Sahami *et al.*, "Supervised and unsupervised discretization of continuous features," in *Machine learning: proceedings of the twelfth international conference*, vol. 12, 1995, pp. 194–202.

[123] U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Machine Learning*, vol. 8, pp. 87–102, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00994007

[124] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters." in *Robotics: Science and Systems*, vol. 2, 2005, pp. 65–72.

[125] D. Ślezak, "Approximate entropy reducts," *Fundam. Inf.*, vol. 53, no. 3-4, pp. 365–390, Aug. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=2371245.2371255

[126] K. J. Archer and R. V. Kimes, "Empirical characterization of random forest variable importance measures," *Computational Statistics and Data Analysis*, vol. 52, no. 4, pp. 2249–2260, 2008. [Online]. Available: www.elsevier.com/locate/csda

[127] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," *Advances in Neural Information Processing Systems 26*, pp. 431–439, 2013. [Online]. Available: https://papers.nips.cc/paper/4928-understanding-variable-importances-in-forests-of-randomized-trees.pdfhttp://media.nips.cc/nipsbooks/nipspapers/paper{_}files/nips26/281.pdf

[128] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[129] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Greenwich, CT, USA: Manning Publications Co., 2011.

[130] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, Jan. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2946645.2946679

[131] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[132] J. Leng, C. Valli, and L. Armstrong, "A wrapper-based feature selection for analysis of large data sets," in *International Proceedings of Computer Science & Information Tech*, 2012.

[133] P. Yang, W. Liu, B. Zhou, S. Chawla, and A. Zomaya, "Ensemble-based wrapper methods for feature selection and class imbalance learning," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Springer Berlin Heidelberg, 2013, vol. 7818, pp. 544–555. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37453-1_45

[134] H. Frohlich, O. Chapelle, and B. Scholkopf, "Feature selection for support vector machines by means of genetic algorithm," in *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, Nov 2003, pp. 142–148.

[135] A. Zagorecki, "Feature selection for naive bayesian network ensemble using evolutionary algorithms," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, Sept 2014, pp. 381–385.

[136] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 74–81. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.658297

[137] P. Saengsiri, S. Wichian, P. Meesad, and U. Herwig, "Comparison of hybrid feature selection models on gene expression data," in *Knowledge Engineering, 2010 8th International Conference on ICT and*, Nov 2010, pp. 13–18.

[138] J. Huang, Y. Cai, and X. Xu, "A hybrid genetic algorithm for feature selection wrapper based on mutual information," *Pattern Recognition Letters*, vol. 28, no. 13, pp. 1825 – 1844, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865507001754

[139] C.-L. Huang and C.-Y. Tsai, "A hybrid sofm-svr with a filter-based feature selection for stock market forecasting," *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 1529 – 1539, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417407006069

[140] M.-C. Lee, "Using support vector machine with a hybrid feature selection method to the stock trend prediction," *Expert Systems with Applications*, vol. 36, no. 8, pp. 10 896 – 10 904, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409001560

[141] J. Xie and C. Wang, "Using support vector machines with a novel hybrid feature selection method for diagnosis of erythemato-squamous diseases," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5809 – 5815,

2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410011942

[142] S. Nemati, M. E. Basiri, N. Ghasem-Aghaee, and M. H. Aghdam, "A novel aco–ga hybrid algorithm for feature selection in protein function prediction," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12 086 – 12 094, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409003637

[143] D. Ruta, "Robust method of sparse feature selection for multi-label classification with naive bayes," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, Sept 2014, pp. 375–380.

[144] Z. Zhu, Y. S. Ong, and M. Dash, "Wrapper ndash;filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 70–76, Feb 2007.

[145] M. Boullé, "Tagging fireworkers activities from body sensors under distribution drift," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept 2015, pp. 389–396.

[146] M. Boullé, *Prediction of Methane Outbreak in Coal Mines from Historical Sensor Data under Distribution Drift*. Cham: Springer International Publishing, 2015, pp. 439–451. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-25783-9_39

[147] M. Boullé, "Predicting dangerous seismic events in coal mines under distribution drift," in *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., vol. 8. IEEE, Sept 2016, pp. 221–224.

[148] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251254.1251264

[149] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, ser. OSDI '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 15–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267308.1267323

[150] "Hdfs architecture guide, howpublished = http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, note = Accessed: 2015-01-29."

[151] "Apache hadoop nextgen mapreduce (yarn), howpublished = http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/yarn.html, note = Accessed: 2015-01-29."

[152] "Hadoop wiki: List of institutions that are using hadoop for educational or production uses, howpublished = https://wiki.apache.org/hadoop/poweredby, note = Accessed: 2015-01-29."

[153] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *Journal of Parallel and Distributed Computing*, no. 0, pp. –, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731514001452

[154] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561 – 2573, 2014, special Issue on Perspectives on Parallel and Distributed Processing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731514000057

[155] R. Schreiber, "A few bad ideas on the way to the triumph of parallel computing," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2544 – 2547, 2014, special Issue on Perspectives on Parallel and Distributed Processing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731513002177

[156] A. Holmes, *Hadoop in practice.* Shelter Island, NY: Manning, 2012.

[157] T. White, *Hadoop: the definitive guide*, 3rd ed. Beijing: O'Reilly, 2012.

[158] L. George, *HBase the definitive guide.* Sebastopol, CA: O'Reilly, 2011. [Online]. Available: http://public.eblib.com/choice/publicfullrecord.aspx?p=769368

[159] Y. Jiang, *HBase administration cookbook master HBase configuration and administration for optimum database performance.* Birmingham: Packt Publishing, 2012. [Online]. Available: http://site.ebrary.com/id/10598980

[160] N. Dimiduk and A. Khurana, *HBase in action.* Shelter Island, NY: Manning, 2013.

[161] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, Oct. 2007. [Online]. Available: http://doi.acm.org/10.1145/1323293.1294281

[162] E. Hewitt, *Cassandra: the definitive guide*, 1st ed. Beijing: O'Reilly, 2011.

[163] D. Miner, *MapReduce design patterns.* Sebastopol, CA: O'Reilly, 2013.

[164] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a high-level dataflow system on top of map-reduce: The pig experience," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1414–1425, Aug. 2009. [Online]. Available: http://dx.doi.org/10.14778/1687553.1687568

[165] A. Gates, *Programming Pig.* Sebastopol: O'Reilly Media, 2011. [Online]. Available: http://public.eblib.com/choice/publicfullrecord.aspx?p=801461

[166] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets." *HotCloud*, vol. 10, pp. 10–10, 2010.

[167] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[168] S. Singh, J. Kubica, S. Larsen, and D. Sorokina, "Parallel large scale feature selection for logistic regression." in *SDM*. SIAM, 2009, pp. 1172–1183.

[169] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[170] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle, "Massively parallel feature selection: an approach based on variance preservation," *Machine Learning*, vol. 92, no. 1, pp. 195–220, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10994-013-5373-4

[171] "Mpi: A message-passing interface standard. version 3.0, howpublished = http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf, note = Accessed: 2015-01-29."

[172] Z. Sun and Z. Li, "Data intensive parallel feature selection method study," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, July 2014, pp. 2256–2262.

[173] G. Caruana, M. Li, and M. Qi, "A mapreduce based parallel svm for large scale spam filtering," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, vol. 4, July 2011, pp. 2659–2662.

[174] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera, "Mrpr: A mapreduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, pp. 331–345, 2015.

[175] A. K. Farahat, A. Elgohary, A. Ghodsi, and M. S. Kamel, "Distributed column subset selection on mapreduce," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 171–180.

[176] A. Guillén, A. Sorjamaa, Y. Miche, A. Lendasse, and I. Rojas, "Efficient parallel feature selection for steganography problems," in *Bio-Inspired Systems: Computational and Ambient Intelligence*, ser. Lecture Notes in Computer Science, J. Cabestany, F. Sandoval, A. Prieto, and J. Corchado, Eds. Springer Berlin Heidelberg, 2009, vol. 5517, pp. 1224–1231. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02478-8_153

[177] L. Zhou, H. Wang, and W. Wang, "Parallel implementation of classification algorithms based on cloud computing environment," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 10, no. 5, pp. 1087–1092, 2012.

[178] R. Rivest, "The MD5 message-digest algorithm," 1992. [Online]. Available: http://tools.ietf.org/html/rfc1321?ref=driverlayer.com

[179] H. Ochiai, H. Ikegami, Y. Teranishi, and H. Esaki, "Facility information management on hbase: Large-scale storage for time-series data," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, July 2014, pp. 306–311.

[180] R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 78–83, may 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6848734/https://xa.yimg.com/kq/groups/72986399/1986154446/name/06848734.pdf

[181] S. Mathew, "Overview of Amazon Web Services," april 2017, accessed: 2017-04-04. [Online]. Available: https://d0.awsstatic.com/whitepapers/aws-overview.pdf

[182] D. Poirier, C. Bothorel, E. G. De Neef, and M. Boullé, "Automating opinion analysis in film reviews: the case of statistic versus linguistic approach," in *Affective Computing and Sentiment Analysis*. Springer, 2011, pp. 125–140.

[183] S. Ferrandiz and M. Boullé, "Supervised selection of dynamic features, with an application to telecommunication data preparation," in *Industrial Conference on Data Mining*. Springer, 2006, pp. 239–249.

[184] F. Fessant, A. Le Cam, M. Boullé, and R. Féraud, "Modelling complex data by learning which variable to construct," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2010, pp. 324–335.

[185] M. Boullé, "Towards automatic feature construction for supervised classification," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 181–196.

[186] D. Lahbib, M. Boullé, and D. Laurent, "Itemset-based variable construction in multi-relational supervised learning," in *International Conference on Inductive Logic Programming*. Springer, 2012, pp. 130–150.

[187] A. Bondu, V. Lemaire, and M. Boullé, "Exploration vs. exploitation in active learning: a bayesian approach," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–7.

[188] M. Boulle, "Khiops: A statistical discretization method of continuous attributes," *Machine learning*, vol. 55, no. 1, pp. 53–69, 2004.

[189] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning spark: lightning-fast big data analysis*. "O'Reilly Media, Inc. 2015.

[190] A. Krasuski, "A framework for dynamic analytical risk management at the emergency scene. from tribal to top down in the risk management maturity model," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. IEEE, 2014, pp. 323–330.

[191] A. Krasuski, A. Jankowski, A. Skowron, and D. Slezak, "From sensory data to decision making: A perspective on supporting a fire commander," in *2013*

IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). IEEE, 2013, pp. 229–236.

[192] M. Meina, B. Celmer, and K. Rykaczewski, "Towards robust framework for online human activity reporting using accelerometer readings," in *Active Media Technology*. Springer, 2014, pp. 347–358.

[193] M. Hendryx, M. M. Ahern, and T. R. Nurkiewicz, "Hospitalization patterns associated with appalachian coal mining," *Journal of Toxicology and Environmental Health, Part A*, vol. 70, no. 24, pp. 2064–2070, 2007.

[194] R. B. Finkelman, "Health impacts of coal: facts and fallacies," *AMBIO: A Journal of the Human Environment*, vol. 36, no. 1, pp. 103–106, 2007.

[195] M. Kozielski, A. Skowron, L. Wróbel, and M. Sikora, "Regression rule learning for methane forecasting in coal mines," in *Beyond Databases, Architectures and Structures*, ser. Communications in Computer and Information Science, S. Kozielski, D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek, and D. Kostrzewa, Eds. Springer International Publishing, 2015, vol. 521, pp. 495–504. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-18422-7_44

[196] M. A. De Georgia, F. Kaffashi, F. J. Jacono, and K. A. Loparo, "Information technology in critical care: Review of monitoring and data acquisition systems for patient care and research," *The Scientific World Journal*, vol. 2015, 2015.

[197] S. Sendelbach and M. Funk, "Alarm fatigue: a patient safety concern," *AACN advanced critical care*, vol. 24, no. 4, pp. 378–386, 2013.

[198] G. B. Moody and R. G. Mark, "A database to support development and evaluation of intelligent intensive care monitoring," in *Computers in Cardiology, 1996*. IEEE, 1996, pp. 657–660.

[199] A. Aboukhalil, L. Nielsen, M. Saeed, R. G. Mark, and G. D. Clifford, "Reducing false alarm rates for critical arrhythmias using the arterial blood pressure waveform," *Journal of biomedical informatics*, vol. 41, no. 3, pp. 442–451, 2008.

[200] G. Clifford, W. Long, G. Moody, and P. Szolovits, "Robust parameter extraction for decision support using multimodal intensive care data," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1887, pp. 411–429, 2009.

[201] B. Baumgartner, K. Rodel, and A. Knoll, "A data mining approach to reduce the false alarm rate of patient monitors," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. IEEE, 2012, pp. 5935–5938.

[202] F. Afghah, A. Razi, S. R. Soroushmehr, S. Molaei, H. Ghanbari, and K. Najarian, "A game theoretic predictive modeling approach to reduction of false alarm," in *Smart Health*. Springer, 2015, pp. 118–130.

[203] J. Behar, J. Oster, Q. Li, and G. D. Clifford, "Ecg signal quality during arrhythmia and its application to false alarm reduction," *Biomedical Engineering, IEEE Transactions on*, vol. 60, no. 6, pp. 1660–1666, 2013.

[204] "Cardiac monitors, heart rate meters, and alarms [american national standard]," *ANSI/AAMI*, no. EC13:2002, 2002.

[205] W. Zong, G. Moody, and R. Mark, "Reduction of false arterial blood pressure alarms using signal quality assessement and relationships between the electrocardiogram and arterial blood pressure," *Medical and Biological Engineering and Computing*, vol. 42, no. 5, pp. 698–706, 2004.

[206] B. Przywara *et al.*, "Projecting future health care expenditure at european level: drivers, methodology and main results," Directorate General Economic and Monetary Affairs (DG ECFIN), European Commission, Tech. Rep. Economic Papers 417, July 2010.

[207] H. Sun, V. D. Florio, N. Gui, and C. Blondia, "Promises and challenges of ambient assisted living systems," in *2009 Sixth International Conference on Information Technology: New Generations*, April 2009, pp. 1201–1207.

[208] P. C. Hallal, L. B. Andersen, F. C. Bull, R. Guthold, W. Haskell, and U. Ekelund, "Global physical activity levels: surveillance progress, pitfalls, and prospects," *The Lancet*, vol. 380, no. 9838, pp. 247–257, 2017/01/24 2012. [Online]. Available: http://dx.doi.org/10.1016/S0140-6736(12)60646-1

[209] S. Kahlmeier, T. M. A. Wijnhoven, P. Alpiger, C. Schweizer, J. Breda, and B. W. Martin, "National physical activity recommendations: systematic overview and analysis of the situation in european countries," *BMC Public Health*, vol. 15, p. 133, 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4404650/

[210] W. H. Organization *et al.*, *Global recommendations on Physical Activity for health.* Geneva, Switzerland: World Health Organization, 2010.

[211] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *International Conference on Pervasive Computing.* Springer, 2004, pp. 158–175.

[212] S. Liu, R. X. Gao, D. John, J. W. Staudenmayer, and P. S. Freedson, "Multisensor data fusion for physical activity assessment," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 3, pp. 687–696, March 2012.

[213] S. Arlot, A. Celisse *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.

[214] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "Shimmer x2122; x2013; a wireless sensor platform for noninvasive biomedical research," *IEEE Sensors Journal*, vol. 10, no. 9, pp. 1527–1534, Sept 2010.

[215] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *European Symposium on Artificial Neural Networks (ESANN)*, ser. Computational Intelligence and Machine Learning, Apr 2013.

[216] “EC2 Instance Types – Amazon Web Services (AWS),” accessed: 2017-03-04. [Online]. Available: https://aws.amazon.com/ec2/instance-types/