**Универзитет „Св. Климент Охридски“ – Битола**

**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ**
**И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ**

**ДОКТОРСКИ ТРУД**

НА ТЕМА

# КОРИСТЕЊЕ НА АЛГОРИТМИ ЗА КЛАСИФИКАЦИЈА НА ГОЛЕМИ ПОДАТОЦИ ЗА ПОДОБРУВАЊЕ НА ПЕРФОРМАНСИТЕ ПРИ НИВНА ОБРАБОТКА

Ментор:

**проф. д-р Виолета Маневска**

Кандидат:

**м-р Владимир Настески**

Битола 2018

**ЧЛЕНОВИ НА КОМИСИЈАТА ЗА ОЦЕНКА НА ДОКТОРСКАТА ДИСЕРТАЦИЈА**

**Д-Р ВИОЛЕТА МАНЕВСКА (Ментор)**

РЕДОВЕН ПРОФЕСОР

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ

УНИВЕРЗИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ – БИТОЛА

**Д-Р ЦВЕТКО АНДРЕЕСКИ (Член)**

РЕДОВЕН ПРОФЕСОР

ФАКУЛТЕТ ЗА ТУРИЗАМ И УГОСТИТЕЛСТВО

УНИВЕРЗИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ – БИТОЛА

**Д-Р ИЛИЈА ЈОЛЕВСКИ (Член)**

РЕДОВЕН ПРОФЕСОР

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ

УНИВЕРЗИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ – БИТОЛА

**Д-Р СНЕЖАНА САВОСКА (Член)**

ВОНРЕДЕН ПРОФЕСОР

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ

УНИВЕРЗИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ – БИТОЛА

**Д-Р ТОМЕ ДИМОВСКИ (Член)**

ВОНРЕДЕН ПРОФЕСОР

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ И КОМУНИКАЦИСКИ ТЕХНОЛОГИИ

УНИВЕРЗИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“ – БИТОЛА

## Декларација

Потврдувам дека, освен за случаи каде е направено прописно цитирање, оваа докторска теза со наслов “Користење на алгоритми за класификација на големи податоци за подобрување на перформансите при нивна обработка” е оригинално дело на самиот автор; тезата не е поднесена претходно, во целост или делумно, да се квалификува за каква било друга академска награда; содржината на тезата е резултат на работата која е извршена од официјалниот датум на одобрување на пријавата; и, секоја уредувачка работа, платена или неплатена, извршена од страна на трети лица е прописно референцирана.

Докторската теза е напишана во (Word 2016) на (Windows 10) платформа.

_____

Владимир Настески

Факултет за информатички и комуникациски технологии

Отсек за менаџмент на информациски системи

Битола, Септември 2018

## Благодарност

Најголема благодарност до моето семејство за нивното разбирање, трпение и поддршка во текот на целокупното студирање, изработката на дисертацијата и мојот напредок во кариерата; непоколебливо им благодарам за несебичната сила и доверба што ми ја даваа во животот.

Исклучителна благодарност до мојот ментор проф. д-р Виолета Маневска која несебично ме поддржуваше од самиот почеток на студиите, ми пружи професионална соработка и комплетно го споделуваше своето знаење во текот на изработката на дисертацијата.

И на крај, особена благодарност до моето друштво, до сегашните и поранешните колеги од Vox Teneo кои несебично и до крај ме поддржуваа и сослушуваа во периодот на изработката на оваа дисертација.

## Апстракт

Големите податоци не претставуваат само податоци коишто зафаќаат само голем простор. Хетерогеноста, скалирањето, комплексноста и приватноста се само дел од предизвиците со кои се соочуваат големите податоци. За разбирање на вредноста на големите податоци потребни се нови или пак надоградени алатки и методи за анализа кои ќе се разликуваат од алатките и методите кои се применуваат кај традиционалните системи. На овој начин се надминуваат и недостатоците кои се јавуваат кај традиционалните системи за анализа во однос на нивната неможност во целост да се справат со предизвиците на големите податоци. Науката за податоци е новата дисциплина на која се адресираат предизвиците кај големите податоци. Алгоритмите за машинско учење се покажаа како доста ефективни во обработката на големите податоци преку создавање на моќен систем за учење и анализа на податоци.

Во оваа насока е и оваа докторска дисертација, каде се разгледани алгоритми за машинско учење преку создавање на модели за одредени студии на случај. Моделите се тестирани, анализирани, оптимизирани и подобрени во тест околина развиена со помош на визуелна рамка за создавање Apache Spark апликации. Развиените модели се базирани на алгоритми за класификација, каде зависно од природата на податоците и студијата на случај истите се подобрени и оптимизирани со користење на одреден метод за оптимизација. Во одредени модели резултатите водат и кон подобрени претпоставки или приближување кон емпириските вредности. На крај моделите се евалуирани, а резултатите се споредуваат преку користење на различни етикети за проверка, со цел утврдување на точноста на моделот. За секоја од моделите се дадени препораки за подобрување на моделите и препораките за натамошно работење.

**Клучни зборови**: големи податоци, машинско учење, mlLib, Apache Spark

## Abstract

The term Bid Data does not only represent a large scale data. The heterogeneity, scaling, complexity and the privacy are just a couple of the challenges that are presented in the Big Data. To understand the value of the Big Data, new analysis tools and methods are needed, which differc from the analysis tools and methods used in the traditional systems. In this way, the defects that arise in traditional systems of analysis in terms of their inability to fully cope with the challenges of large data will be overcome. Data science is the new discipline that addresses the challenges of large data. Machine learning algorithms have proved to be quite effective in the processing of large data by creating a powerful learning and data analysis system.

In this doctoral dissertation several algorithms for machine learning are considered by creating models for certain case studies. The models are tested, analyzed, optimized and improved using a test environment developed in a visual framework for creating Apache Spark applications. The developed models are based on several classification algorithms, where, depending on the nature of the data and the case study, the models are improved and optimized using particular optimization method. In some models the results lead to better predicted values. In the end the results of the models are evaluated and compared using different evaluation metrics in order to determine the accuracy of the model. For each model, recommendations have been made to improve models and recommendations for future work are given appropriately.

**Key words**: Big Data, machine learning, mlLib, Apache Spark

## Содржина

# 1 ВОВЕД

## 1.1 ОБРАЗЛОЖЕНИЕ НА ТЕМАТА

Денес развојот на информатичките технологии се соочува со големи предизвици во процесот на обработка на големи податоци (Big Data[1]) поврзани со нивната ефективна и ефикасна обработка. Големите податоци можат да се опишат како природен извор кој сé повеќе се зголемува. Како и секој ресурс, некои од податоците тешко можат да се извлечат заради нивната разновидност, односно заради постоењето на различни типови на податоци. Во поголемиот број на случаи големите податоци можат тешко да се прочистат или пак анализираат. Голем дел од компаниите не навлегуваат во нивните детали: ги игнорираат податоците, односно ги обработуваат само тие што им се потребни, или пак ги користат само за одредена цел.

Во целина, големите податоци наметнуваат развој на една комплетна стратегија за управување со информации што вклучува и интегрира многу нови типови на податоци и нивно управување. Во последно време се развиени многу техники за процесирање и анализирање на податоците, што се популаризира кај многу корисници за нивно адаптирање, односно користење на различни модели за анализа и визуелизација на големите податоци.

Големите податоци се опишуваат со четири атрибути[2]: **волумен (големина)**, **брзина**, **разновидност** и **веродостојност**.

Големината на податоците е најважниот атрибут. Големите податоци процесираат огромни податоци кои воглавно се од непознат карактер, како на пример новости од социјални мрежи, кликови на веб страни, мрежен сообраќај итн. Основната задача на големите

[1] https://www.ibm.com/big-data/us/en/
[2] http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data-veracity/

податоци е извршување на успешно конвертирање од податоци со мала густина во информации со голема густина, односно во информации кои имаат одредена вредност.

Брзината на податоците со која тие се анализираат се дефинира како однос помеѓу влезот на податоците, пример во одредена компанија и времето потребно компанијата да ги анализира и разбери податоците.

Постојат и неструктурирани типови на податоци. За неструктурираните и полуструктурираните типови на податоци, како што се текст, звук или пак видео, потребно е додатно процесирање за да се подигне нивната вредност во големите податоци, односно потребна е понатамошна нивна дообработка во форма на мета податоци. Неструктурираните податоци подлежат на истите потреби како и структурираните податоци, односно: сумирање (агрегација), потекло на податоците, приватност итн.

Веродостојноста на податоците се однесува на квалитетот, односно вистинитоста на податоците. Постојат најразлични алатки што ги пречистуваат податоците и ги трансформираат во веродостојни податоци.

Поради разновидните стратегии на одредена компанија која располага со големи податоци, резултатите од нивната обработка и анализа влијаат врз нејзиното пласирање на пазарот на трудот. Платформата за големи податоци ја нуди можноста за анализа на податоци, што би значело дополнително разбирање на работата на бизнисот, нивните корисници, нивниот пласман на пазарот итн.

Пресметувањето на протокот на податоци (stream computing), како дел од платформата на големи податоци, врши анализа на живи податоци, односно податоци се добиваат веднаш, се обработуваат и се добива анализа во реално време. При ова пресметување се користи таканаречената аналитичко пресметување во реално време (Real-time analytical processing - RTAP).

Најважниот слој на платформата за големи податоци претставуваат алатките за откривање, разбирање, пребарување и навигација на различните извори на големите податоци.

Најважните аспекти на платформата за големи податоци се:

- **Интеграција**. Идејата на интеграција е да се користи една платформа за менаџирање на сите податоци. Нема потреба од користење на различни платформи при анализа на податоците.

- **Анализа**. Големите податоци претставуваат место за анализа и меморирање на податоците, бидејќи новите технологии веќе не претставуваат само предпроцесирање на податоците што се претходно структурирани во складиште на податоци за понатамошна анализа.

- **Визуелизација**. Доставување на податоците до корисниците во прифатлив облик.

- **Оптимизација на обработката**. Подобрување на процесирањето и складирањето на податоци.

- **Заштита на податоците**. Постојат ранливи податоци кои треба да се заштитат, односно да се детерминираат одредени правила за заштита на податоците.

Комплексноста на податоците се зголемува од ден на ден заедно со зголемувањето на изворите на податоци, што претставува нов извор на предизвик за обработка и анализа на податоците. Податоците стануваат сé понеизвесни, а тоа директно влијае врз различни иницијативи чија основна цел е обработка на податоците. Вистинитоста на податоците се намалува како што нивната разноличност се зголемува.

Поради ваквата природа, компаниите треба да го подобрат нивото на доверба на корисниците во однос на разновидноста на податоците, а со тоа да се осигура и конзистентноста на податоците, што би претставувало важен дел од стратегијата на големите податоци.

За повеќето компании анализата на податоци претставува предност за подобрување на бизнисот, бидејќи на индиректен начин иднината на бизнисот зависи од анализата на

податоците. Најчесто компаниите не можат сами по себе да ја вршат анализата. За оваа цел, информациите коишто ги добиваат за понатамошна анализа мора да бидат доверливи.

Компаниите сè повеќе користат продукти и сервиси кои се фокусирани на корисникот, што значи зголемување на потребата од машинско учење за развој на персонализација, препораки и предвидувања, односно претпоставки. Овие проблеми се решаваат со користење на соодветни алатки и одреден програмски јазик за нивна комуникација, како на пример R или Python. Но, како што податоците стануваат сé поголеми и се користат различни типови на податоци, сé повеќе се губи време за прилагодување на постоечката инфраструктура, наместо градење на подобри модели за решавање на нивните проблеми.

Тргнувајќи од оваа точка и сметајќи дека оваа област е сé уште во својот подем, ќе биде реализирано истражување **колку користењето на модуларна платформа за обработка на големи податоци и искористување на поединечни алгоритми за машинско учење може да биде доволно едноставно и скалабилно за обработка во одредени студии на случај врз големи податоци.**

Во оваа насока е и оваа дисертација, со која ќе се даде директен научен придонес во областа на обработката на големи количества на податоци. Денес големите компании секојдневно вложуваат во споделување на разни истражувања во областа на големите податоци и во размената на искуства на разни истражувачи коишто даваат свој придонес во оваа област.

### 1.2 Предмет и област на истражување

Како што беше напоменато, големите податоци претставуваат големо количество на податоци што неминовно ја наметнува потребата од развивање нови технологии и архитектури кои ќе послужат за извлекување одредени вредности од податоците за понатамошна анализа. Изворите на големи податоци секојдневно се зголемуваат, тргнувајќи од податоци од сообраќаен менаџмент, па сè до податоци кои се пренесуваат преку персоналните уреди. Големите податоци се издигнаа во последните неколку години

поради тоа што живееме во општество во кое секојдневно бројот на податоци забрзано го зголемува.

Во општество со толку многу податоци, многу тешко може да се применат стандардните техники за извршување на ефективна анализа. Бидејќи големите податоци се релативно нова технологија на пазарот кои можат да им понудат огромен бенефит на компаниите, сè повеќе се наметнува фактот дека треба да се прифатат различните предизвици и проблеми поврзани со адаптирање и разбирање на оваа технологија. Концептот на големи податоци покажува дека множеството на податоци кое рапидно се зголемува многу тешко може да се менаџира користејќи ги стандардните концепти за анализа на податоците. Потешкотиите се најчесто поврзани со земање на самите податоци, просторот, пребарувањата, споделувањата, анализата и визуелизацијата итн.

Се поставува прашањето, што сé може да се прави со големите податоци? Со големите податоци корисниците имаат далеку подобар поглед кон податоците, поглед што значи подобрување не на самиот бизнис, туку и подобрување на услугите кон корисниците.

Со цел искористување на големите податоци од страна на компаниите или пак одредени институции, истите мора повторно да го проверат начинот на нивно менаџирање во своите податочни центри. Податоците пристигнуваат од различни извори, во или надвор од компанијата. Тие можат да бидат во форма на видеа, податоци од социјални мрежи, документи или пак машински генерирани податоци, од многу апликации или пак платформи. Од тие причини на компаниите им е потребен систем за оптимизација и организација на структурирани, неструктурирани или полуструктурирани податоци во нивните бази на податоци за понатамошна анализа. Анализата мора да биде брза со цел постигнување на бараните бизнис цели.

Скалабилноста на податоците претставува можеби еден од најголемите предизвици за компанијата. Доколку брзото зголемување на податоците влијае врз просторот на податочните центри, компаниите мора брзо да реагираат на зголемување на просторот.

Проектот Apache Hadoop[3] претставува софтверска рамка што дозволува дистрибуирано паралелно процесирање на огромни податоци низ евтини, стандардни сервери кои истовремено зачувуваат и процесираат податоци, со истовремено скалирање без граници. Предноста на Hadoop е во тоа што може да анализира и обработува податоци од различен тип: структурирани, неструктурирани, лог податоци, слики, аудио, комуникациски записи, електронски пораки итн.

Меѓу најчестите предизвици[4] кај големите податоци се следните:

- **Хетерогеност**. Луѓето ги примаат информациите со голема толерантност во однос на хомогеноста на истите. Од друга страна, машините кои анализираат разни алгоритми, очекуваат хомогени информации и немаат голема толерантност. Поради тоа, податоците мора да бидат внимателно структурирани за нивната понатамошна анализа. Компјутерските системи најефективно работат доколку зачувуваат повеќе податоци кои се со идентична големина и структура. Дури и по чистење на податоците и поправка на нивните грешки, некои од нив остануваат и понатаму. Некомплетноста и грешките може да се поправат за време на анализата, што претставува понатамошен предизвик при нивната обработка.

- **Скалирање.** Основниот параметар што се забележува кај големите податоци е нивната големина. Менаџирањето на големи податоци чија големина рапидно расте, останува основен предизвик кај големите податоци. Во минатото овој предизвик имаше голем удел во забрзување на самите процесори. Но, од денешна перспектива, големите податоци се скалираат побрзо од пресметувачките ресурси, додека пак брзините на процесорот се статични. Денес, иако хардверската технологија секојдневно се подобрува, сè уште не може да ја достигне брзината на анализа и обработка на големите податоци.

[3] http://hadoop.apache.org/
[4] http://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf

- **Приватност.** Отсекогаш приватноста на податоците претставувала проблем, особено кога станува збор за големи податоци. На пример, за електронските податоци во сферата на здравството постојат одредени закони според кои мора да се запази приватноста на податоците. Од тие причини менаџирањето со приватноста истовремено претставува и технички и социолошки проблем.

- **Навременост.** Друг параметар на големите податоци е брзината. Колку што се поголеми податоците за процесирање, толку е подолго времето за анализирање. Дизајнот на системот што ефективно се справува со големината е резултат на системот што побрзо процесира одредена големина на податоци. Во поголемиот број на случаи веднаш е потребен резултатот од анализата на податоците. Во големо множество од податоци, често пати потребно е да се најдат елементите што задоволуваат одредени критериуми. Апсолутно непрактично е целосно скенирање на податоците. За побрзо скенирање на податоците се користат одредени структури на индексирање за брзо пребарување на елементот. Проблемот што се јавува тука е дека секоја структура на индексирање е дизајнирана за да задоволи одредени критериуми. Во секој случај, за секој проблем за брзо пребарување на податоци потребни се нови структури на индексирање на податоците, што исто така претставува еден основен предизвик.

**Фокусот** на истражувањето ќе биде поставен врз истражување и примена на практични студии на случај во околина каде преку обработка на големи податоци со помош на различните алгоритми за машинско учење се добиваат напредни анализи за остварување на различни цели. За оваа потреба ќе се користи одредена платформа за обработка и анализа на големи податоци која е лесно скалабилна за тестирање и креирање на различни модели, каде што лесно може да се имплементираат различните алгоритми за машинско учење.

Целокупниот процес на истражувањето ќе се базира на следните активности:

1. Преглед и користење на Hadoop податочниот систем за пренос на податоци во облак.

2. Користење на слободни бази на податоци или бази на податоци од одредена институција.

3. Анализа на постоечки модели на студии на случај.

4. Преглед на постоечки платформи за големи податоци.

5. Планирање и креирање на практични модели за обработка на големи податоци и напредни анализи на моделите за различни студии на случај.

6. Примена и истражување на различни алгоритми за машинско учење, нивна анализа, развивање, оптимизирање, анализирање на модели и креирање препораки за практична примена на истите.

Имајќи го во предвид погоре наведеното, **ПРЕДМЕТОТ НА ИСТРАЖУВАЊЕ на оваа дисертација се дефинира како: истражување, примена и подобрување на моделите за студии на случај со користење на алгоритми за машинско учење на платформа за развој на модели за обработка и анализа на големи податоци.**

### 1.3 Цели на истражувањето

Големите компании преку вложување во истражувањата поврзани со големите податоци, најмногу се фокусираат на помагање на останатите компании да ги разберат проблемите што се генерираат од обработката на големите податоци.

Секоја компанија, на самиот почеток треба да го разбери концептот на големите податоци и која бизнис вредност може да ја извлече од нив за постигнување на одредена бизнис цел. Кога станува збор за големите податоци, меѓу поважните use-case[5]-и се вбројуваат и:

[5] Листа од дејства или случаи кои најчесто ги дефинираат интеракциите помеѓу дадена улога и одреден систем за постигнување на целта
https://en.wikipedia.org/wiki/Use_case

- **Истражување на големите податоци** - пребарување, визуелизација и подобрување на процесот на донесувањето на одлуки односно истражување на големите податоци од кое произлегува потребата секоја голема компанија да се соочи со проблемот на информации кои се зачувани на различните системи и потребата од пристап до нив за донесување на одлуки.

- **360-степени поглед на корисникот** - проширување на погледите на постоечките корисници со вклучување на додатни извори за внатрешни и надворешни информации. Со помош на ова целосно разбирање на корисниците, односно што тие кликаат, зошто купуваат одредени производи, може да се креира одлична анализа за развивање на бизнис интелигенција на одредена компанија.

- **Оперативни анализи** - анализа на различни машини и оперативни податоци за подобрување на одредени резултати. Во ова се вклучуваат и различни сензори и GPS уреди што бараат комплексна анализа и корелација низ различни типови на податоци.

- **Безбедност при размена на податоци** - намален ризик, мониторинг на безбедност на податоците во реално време. Користењето на разни платформи на големи податоци за процесирање и анализа на нови типови на податоци би помогнало во подобрување на бизнис интелигенцијата и безбедноста на податоците.

- **Модернизација на складиштата на податоци** - интеграцијата на големи податоци и можностите кои ги нудат складиштата на податоци овозможуваат да се зголеми оперативната ефективност. Оптимизацијата на складиштето на податоци наметнува нови типови на анализа.

Секој систем одлично работи за одредени use-case-и, па секоја компанија користи комбинација на алатки за да постигне одредена цел. Имајќи ги во предвид најопштите применливи случаи, најпознати use-case-и што ги користи на пример Apache Spark[6] се:

- **Стриминг (проток) на податоци (streaming data)** - Основниот use-case на Spark е да процесира проток на податоци, што широко се користи од многу компании за анализа на ваков тип на податоци.

- **Машинско учење** - Spark доаѓа со интегрирана рамка за извршување на напредни анализи што им помага на корисниците да користат повторувачки прашања на различно множество на податоци, при што од голема помош се алгоритмите за машинско учење. Во Spark е воведена рамка за скалабиност, Machine Learning Library (MLlib). Оваа рамка може да работи во области како кластерирање, класификација и димензионална редукција итн.

- **Интерактивна анализа** - Spark е доволно брз за обработка на истражувачки прашања без семплирање. Комбинацијата на Spark со разни визуелни алатки во одредени ситуации може да процесира и визуелизира комплексни множества на податоци.

- **Магла пресметувањето (fog computing)** – Магла пресметувањето го децентрализира процесирањето на податоци и нивниот простор, односно функциите се извршуваат во мрежата. Магла пресметувањето е комплексно во процесирањето на децентрализирани податоци, бидејќи е потребна ниска латентност, паралелно процесирање на машинско учење и комплексни алгоритми за графичка анализа.

Конкретната цел во оваа дисертација ќе биде анализа на алгоритмите претставени во библиотеката за машинско учење MLlib, што е дизајнирана за едноставност, скалабилност и лесна интеракција со другите алатки.

[6] http://spark.apache.org/

Како што е веќе напоменато, помеѓу најчесто применуваните алатки во областа на големите податоци се Python, R и Scala што се користат за развој на нови модули или пакети за решавање на проблемите со податоците. Искористувањето на овие модули за комерцијална употреба е ограничено, бидејќи податоците се процесираат на една машина, за анализата е потребно семплирање на податоците, додека пак миграцијата во околина на продукција има потреба од повторна дообработка на самите модули и модели.

Најчесто Apache Spark го решава овој проблем нудејќи моќни, унифицирани библиотеки и модули што се од една страна брзи (100 пати побрзи од Hadoop за процесирање на големи податоци), а од друга страна лесни за употреба. Со ова се овозможува брзо решавање на проблемите со машинско учење (графичко пресметување, пресметување и обработка на живи податоци и процесирање на интерактивни прашања во реално време).

Имајќи ги во предвид повеќето познати use-case-и што ги користат алгоритмите за машинско учење и имајќи ја во предвид генерализацијата на одредени use-case-и за различни одлуки, се наметнува и **ТЕОРЕТСКАТА ЦЕЛ НА ДОКТОРСКАТА ДИСЕРТАЦИЈА: како примената и креирањето препораки врз одредени алгоритми за машинско учење со новокреирани модели на студии на случај ќе влијае врз подобра анализа и ќе креира подобри претпоставки за задоволување на одредени цели преку користење на платформа за обработка на големи податоци.**

### 1.4 Актуелност на истражувањето

Денес, користењето на големи податоци сè повеќе се популаризира бидејќи новите предности во технологијата и инфраструктурата дозволува подобро процесирање и подобра анализа на податоците. Моќта на процесирање секојдневно се зголемува, а истовремено се намалува цената на процесорите, со што оваа технологија им стана попристапна на малите и средни компании.

Големите компании ја користат можноста за креирање на продукти и сервиси со цел да им помогнат на компаниите сè повеќе да ги користат нивните продукти, а со цел добивање на

подобри резултати од анализа на податоците и подобра скалабилност. Алатките за анализа на големи податоци денес се сè помоќни, послободни, некои од нив се многу поевтини и нивната употреба е далеку полесна за корисниците.

Голем дел од компаниите се инспирирани од моќта на технологијата и се ориентираат кон искористување на големите податоци со цел решавање на нивните проблеми. Мноштвото на информации константно се менува, па затоа многу компании се насочуваат кон прибирање на информации во реално време, што станува сé попредизвикувачко за развивачите на оваа технологија.

Машинското учење[7] е наредниот предизвик кај големите податоци. Програмскиот интерфејс на машинското учење не е комплексен и лесно е разбирлив за секој и најмногу се користи кај развивачите на апликации кои сакаат да се занимаваат со податочно рударење, кориснички дизајн, експериментирање и анализа на податоци. Наспроти ова тврдење, користењето на интерфејсот кај R или Matlab библиотеките и не е така едноставно, односно потребно е посветено работење со нив за да се стигне до ниво на рутинско користење.

Во претходните децении, алгоритмите за машинско учење и нивните технологии, најчесто биле користени при напредна анализа на податоци. Денес, неколку компании го користат програмскиот интерфејс на машинското учење со цел негово популаризирање и доближување кон обичните компании. Истиот е креиран со цел лесна имплементација на машинското учење врз податоците при поставувањето на одредени претпоставки во апликациите кои обработуваат податоци. Голема предност е што овој интерфејс нуди апстрактно ниво со цел развивачите на апликации да можат лесно да го интегрираат без да се грижат дали алгоритмите се скалабилни за нивната инфраструктура.

[7] http://www.ngdata.com/machine-learning-and-big-data-analytics-the-perfect-marriage/

MLlib[8] претставува огромен успех на Spark. Основната придобивка на MLlib е што им овозможува на истражувачите да се фокусираат на проблемите со податоците и моделите, наместо решавање на комплексни проблеми поврзани со дистрибуираните податоци, како што се инфраструктура, конфигурации и сл. MLlib е библиотека за генерална употреба што нуди алгоритми кои решаваат проблеми за најчестите use-case-и, а исто така им овозможува на научниците повторно искористување и подобрување на алгоритмите за специфични use-case-и.

Помеѓу предностите на MLlib се вбројуваат:

- **Едноставност** - едноставноста на оваа библиотека доаѓа од користењето на алатки како што се R или Python. Истражувачите можат да ги користат алгоритмите што ги нуди библиотеката или пак да ги подобруваат истите со менување на параметрите во алгоритмите.

- **Компатибилност** - научниците најчесто креираат модули што се користат за одредени алатки за податоци, како R или Python. Со помош на DataFrames и MLlib многу лесно може да се интегрираат постоечките модули во Spark.

- **Скалабилност** - кодот што се изработува на една машина може повторно да се искористи на голем кластер без никаков проблем. Со ова им се овозможува на компаниите искористување на истите или пак подобрени бази на податоци.

Во исто време пак, Spark нуди решавање на повеќе проблеми со податоците поврзани со машинското учење. Екосистемот на Spark може да реши графички проблеми, обработка на живи податоци (streaming) и интеракција со процесирање на прашања во реално време со помош на SparkSQL и DataFrames. Овие можности што ги нуди оваа библиотека овозможуваат фокусирање на проблемите со податоци, наместо одржување на различни алатки за секое сценарио.

[8] https://spark.apache.org/docs/latest/mllib-guide.html

Оваа дисертација ќе се движи во правец на актуализирање, оптимизирање и подобрување на моделите од алгоритмите за машинско учење чија цел е обработка, анализа на големи податоци и добивање на сè поточни претпоставки за задоволување на одредена цел.

### 1.5 Методологија на истражувањето

Методологијата која ќе биде применета за одговарање на клучните прашања при реализација на планираните цели може да се структурира на следниот начин:

- Користење на аналитички техники за истражување на досегашните резултати во областа на големите податоци, истражување на публикувани научни трудови поврзани со големите податоци, алгоритмите за машинско учење и методологијата за справување со нив, компаративна анализа на предложените решенија, синтеза на добиените знаења и синтетизирање на најдобрите практики со цел да се актуелизира практичната примена на алгоритмите за машинско учење при решавање на проблемите поврзани со големите податоци.

### 1.6 Структура на дисертацијата

Оваа докторската дисертација се состои од повеќе глави, во кои се опфатени големите податоци, нивната инфраструктура и анализа, платформите за анализа на големите податоци, процесот на машинското учење и алгоритмите за машинско учење, метриките за евалуација, како и имплементацијата на алгоритмите за машинско учење и нивната анализа во насока на подобрување на перформансите при обработката.

### 1.7 Очекувани резултати

Резултатите кои би произлегле од истражувањата спроведени за оваа докторска дисертација директно би влијаеле врз секојдневната примена на платформата за обработка на големи податоци со користење на алгоритми за машинско учење и

препораките за нивно подобрување во решавање на практични проблеми на голем број на студии на случај.

Моделирањето во околина каде што се добиваат резултати од напредни анализи на големи податоци во голема мера ќе влијае врз решавањето на проблемот со секојдневното зголемување на податоците за одредени случаи. Исто така, ќе се покаже дека со помош на подобрување на моделите можно е добивање на сè поточни претпоставки за постигнување на одредени цели за различни студии на случај.

# 2 ГОЛЕМИ ПОДАТОЦИ

## 2.1 ДИНАМИКА НА РАЗВОЈ КАЈ ГОЛЕМИТЕ ПОДАТОЦИ

Во теоријата на научната дисциплина информатика постои модел, т.н. пирамида на знаење што претставува врска помеѓу податоците, информациите и знаењето (Слика 1). Бидејќи секојдневно се креираат нови податоци, податоците не се обработуваат ефективно во форма на информации; процесот на извлекување и креирање на нови знаења се запира при што се креираат многу поголеми проблеми поврзани со трансформирањето на податоците во информации или пак во знаење. Затоа е потребно систематско извлекување на вредностите од суровите податоци.

**Слика 1:** Пирамида на знаење [9]

Големите податоци претставуваат една парадигма чиешто појавување е со цел решавање на проблеми кои во минатото биле нерешливи, што резултира со креирање на нов мултидисциплинарен правец, наречен наука за податоците.

[9] https://en.wikipedia.org/wiki/DIKW_pyramid

Науката за податоците претставува множество од дисциплини за решавање на предизвиците со големите податоци. Податоците, технологиите и луѓето се трите основни столбови на оваа научна дисциплина, па повеќе од јасно е дека овие три компоненти не можат во никој случај да се постават на исто рамниште. Податоците се насекаде, а различните технологии се тие што секојдневно се развиваат со цел решавање на проблемот со големите податоци.

Поимот големи податоци веќе не претставува енигма – тој ги опишува предизвиците со податоците со кои научниците или пак корисниците кои работат со големи податоци се соочуваат и ќе се соочуваат во иднина. Пред да се навлезе во детали за тоа што навистина претставуваат големите податоци, се поставува прашањето, како и зошто започна ерата на големи податоци? Постојат неколку фактори што се клучни: најважниот фактор е брзиот развој на компјутерската технологија резултирајќи со огромна експанзија на податоци, започнувајќи од зголемувањето на брзината на процесорите и брзината на мрежите, па се до софтверските технологии што поддржуваат дистрибуирани и паралелно-процесирачки рамки. Од друга страна пак, доста важна улога играат и веб базираните софтвери, како што се алатките за пребарување, социјалните мрежи, системите за онлајн плаќања итн. Сите овие фактори делуваат врз зголемувањето на податоците, што се смета за започнување на нова ера, односно ера на големи податоци.

Секојдневно се креираат околу $2.5$ трилиони бајти на податоци, од кои $90\%$ од податоците во светот се креирани само во последната деценија (Cárdenas, 2013). Забрзаниот раст на информации е причина за создавање на нови технологии за анализа на огромните количества на податоци.

Исто така, денес, развојот на информатичките технологии се соочува со големи предизвици во процесот на обработка на големи податоци поврзани со нивната ефективна и ефикасна обработка. Кај големите податоци како и кај секој ресурс, некои од податоците тешко можат да се извлечат заради нивната разновидност, односно заради постоењето на различни типови на податоци. Во поголемиот број на случаи големите податоци тешко

можат да се прочистат или пак анализираат. Голем дел од компаниите не навлегуваат во нивните детали: ги игнорираат податоците, односно ги обработуваат само тие што им се потребни, или пак ги користат само за одредена цел.

Напредокот во технологијата, вклучувајќи ја и дигиталната економија, како и дигитализацијата на индустриите, сé повеќе ја наметнуваат потребата од креирање или подобрување на некои постоечки методи, социјална интеграција, бизниси интелигенција или пак анализа на податоци. Овие трендови предизвикуваат еден експоненционален раст на генерираните податоци, креирајќи нови постапки за работење со податоците, нови форми на пресметување со зголемени пресметувачки брзини, што резултира со појавување на нови проблеми, како на пример проблемот со скалабилност на податоците. Сите овие предизвици и проблеми кои се актуелни и кои се наметнуваат во последните години се вгнездени во поимот големи податоци. Во таа смисла, поимот големи податоци се поистоветува со поимот Web 2.0 набљудуван како синтеза на повеќе трендови (Lugmayr, 2017).

Денес, најчестиот проблем на кој се наидува при работењето со големите податоци е скалирањето, односно зголемувањето на самите податоци и нивната понатамошна обработка за постигнување на одредена цел. Иако податоците се зголемуваат секојдневно, реалниот проблем што датира во хардверската индустрија од самиот почеток е самото процесирање и капацитетот каде податоците треба да се сместат. Како решение за овој проблем се наметнуваат пресметувањето во облак и виртуелизацијата како нови технологии за складирање на податоците и нивна побрза обработка (Das, 2013).

Феноменот Интернет на нештата (Internet of Things - IoT), што најчесто е поврзан со создавањето на огромното количество на податоци, преставува еволуција каде што одредени објекти ја имаат способноста да дејствуваат врз други објекти. Феноменот IoT отвора сè повеќе нови предизвици. Така на пример, енергетските компании денес можат да имаат преглед на податоците во реално време; податоците собрани од мерните станици поставени во домовите за потрошувачката на енергија можат да делуваат директно врз

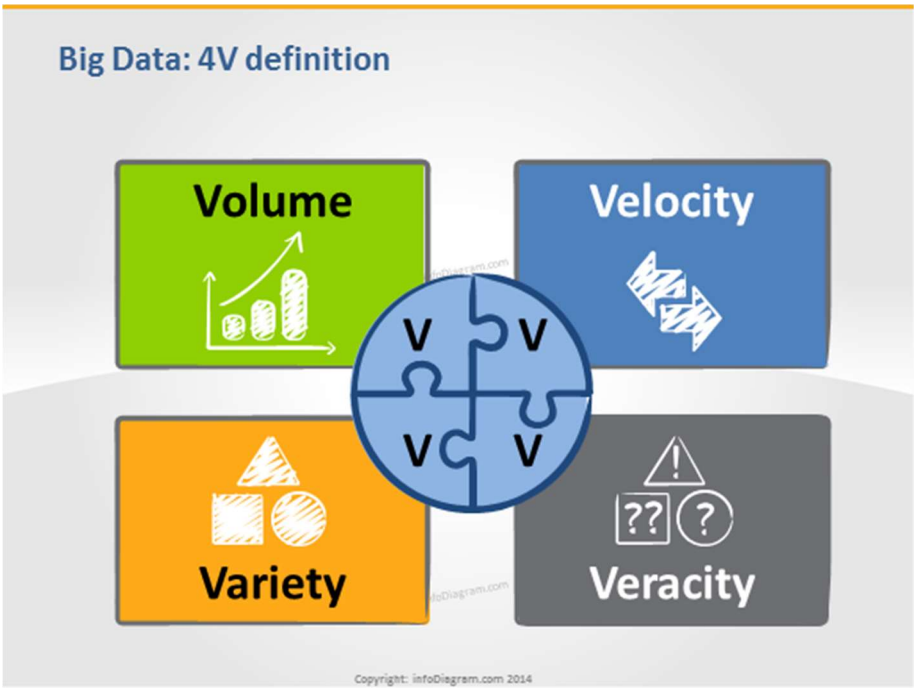
самата компанија, односно врз нејзиното подобрување во иднина. Друг пример се податоците од социјалните мрежи. Податоците собрани од социјалниот свет се комплексни, најмногу заради нивната природа, односно нивниот неструктуриран тип: слики, Twitter новости, Facebook постови итн. На социјалната мрежа Facebook, вклучувајќи го и креирање на нови корисници во текот еден ден, се генерираат и по неколку десетици терабајти дневно. Форматот на податоци што Facebook го произведува се структурирани податоци, односно генерира JSON (JavaScript Object Notation - JSON) формат на податок, што од една страна лесно е да се зачува и анализира за разлика од неструктурираните податоци кои тешко можат да се анализираат. Twitter од друга страна претставува сосем поинаков феномен. На оваа социјална мрежа, преку креирање на најразлични секвенци од карактери од страна на корисниците се генерира огромен број на структурирани податоци.

Социјалниот свет самиот по себе претставува точка на модулација. Истиот се движи од текстуален кон визуелен свет за што потврда претставуваат бројните социјални страници (Vine, Snapchat, Pinterest итн.) кои се базирани на сликовна или видео комуникација, што од своја страна претставува нов предизвик при собирањето и анализата на неструктурираните податоци.

Во целина, големите податоци наметнуваат развој на една комплетна стратегија за управување со информации што вклучува и интегрира многу нови типови на податоци и нивно управување. Во последно време се развиени многу техники за процесирање и анализирање на податоците, што се популаризира кај многу корисници за нивно адаптирање, односно користење на различни модели за анализа и визуелизација на големите податоци.

## 2.2 Карактеристики на големите податоци

Големите податоци најчесто се опишуваат со четири атрибути познати како 4В (4V). Тоа се: големина (Volume), разновидност (Variety), брзина (Velocity) и веродостојност (Veracity) (Слика 2).




**Слика 2:** Четирите карактеристики на големите податоци [10]

Најважниот атрибут кај големите податоци претставува големината. Имајќи го во предвид фактот дека податоците пристигнуваат од најразлични извори од различен тип (податоци од еколошки карактер, финансиски, медицински податоци итн), секојдневно се генерираат огромен број на разновидни податоци кои подлежат на дополнителни анализи за остварување на одредена цел. Социјалните мрежи се најпопуларните мрежи кои генерираат големо количество на податоци дневно. Само за споредба, во 2000 година, околу 800.000 PB беа достапни во светот (Zikopoulos P. a., 2011).

Компаниите што користат огромен број на податоци се соочуваат со проблемот за нивно менаџирање. Имајќи ја соодветната платформа за анализирање на сите податоци се формира подобра слика на самата компанија, корисниците на бизнисот и конкуренцијата. Како што податоците се зголемуваат, процентот на податоци што компанијата може да ги процесира, разбира и анализира се зголемува, што води до т.н. слепа зона (Слика 3).

[10] https://blog.infodiagram.com/2014/04/visualizing-big-data-concepts-strong.html


**Слика 3:** Слепа зона за процесирање и анализа податоци (Zikopoulos P. a., 2011)

Слепа зона претставува зона која не е позната, а која може да биде или нешто навистина позитивно и добро за бизнисот или пак бизнисот да нема никаква корист.

Големината како атрибут кај големите податоци креира нов предизвик кај податочните центри (Data marts), а тоа е разновидноста на податоците. Со развојот на технологијата, употребата на сензори, паметни уреди, како и социјалните технологии, самите податоци што ги поседуваат компаниите стануваат покомплексни, бидејќи не само што имаат обични релациони бази на податоци, туку сурови, полуструктурирани или неструктурирани податоци од веб страниците, лог податоци, индекси на пребарување, форуми од социјални медиуми, е-пораки, документи, податоци од сензори на активни и пасивни системи итн. Што е поинтересно, традиционалните системи може да ги зачуваат и да извршат анализа со цел да го разберат концептот на овие лог податоци, бидејќи повеќето од информациите што се генерираат не се совпаѓаат со технологијата на традиционалните системи. Иако некои компании се во тек со новите технологии, сè уште не го разбираат концептот и можностите на големите податоци.

Разновидноста ги опфаќа сите типови на податоци, односно претставува чекор напред во потребата за анализа на податоци од традиционално структурирани во сурови, полуструктурирани и неструктурирани податоци како дел од процесот на донесување на одлуки. Искуствата покажуваат дека традиционалните платформи за анализа не се способни во целост да се справат со разновидноста на податоците. Успехот на една компанија се согледува во можноста за справување со анализата на различни типови на податоци, традиционални и нетрадиционални.

Пример, 80% од вкупните податоци во светот се неструктурирани податоци или пак во најдобар случај полуструктурирани. Најголем дел од времето за обработка се фокусира на останатите 20% од податоците. Доколку, на пример, се разгледа Twitter, структурата е во JSON формат, вистинскиот текст не е структуриран. Податоците од видео формат или слика не се зачувуваат лесно во релациони бази; одредени информации може да се сменат, односно да бидат динамични (Zikopoulos P. a., 2011).

Основната цел на компаниите кои ги користат големите податоци како извор за понатамошна цел треба да им биде можноста за анализа на сите типови на податоци, релациони и нерелациони, текст, податоци од сензор, аудио, видео и други типови на податоци.

Брзината на големите подароци се однесува на брзината на податоците кои се генерираат, складираат и анализираат во одредено време, имајќи ја во предвид брзината на зголемување на податоците секојдневно и протокот на движење на податоците од различни извори.

Конзистентноста кај базите на податоци се темели на два модели: ACID (Atomic - атомичност, Consistent - конзистентност, Isolated - изолација, Durable - трајност) и BASE (Basic Availability - основна достапност, Soft-state мека состојба, Eventual consistency – евентуална конзистентност). Моделот ACID најчесто се користи кај релационите бази на податоци, додека пак BASE моделот најчесто се користи кај нерелационите бази на податоци од причина што некои од нерелационите бази на податоци имаат потреба од

елиминирање на својствата на моментална конзистентност, освежување на податоците или точност со цел да се добијат други бенефиции, како што се издржливоста и скалабилноста (GC, 2016).

Со цел постигнување на брзина, мора да започне од самиот проток на податоците. Брзината на податоците во овој случај може да се дефинира како брзина со која податоците течат. Бидејќи денес компаниите обработуваат податоци од ред на петабајти, традиционалните системи за зачувување и обработка на податоци скоро невозможно е да се справат со брзината на размена и анализа на податоците.

Во справувањето во т.н. борба со конкуренцијата понекогаш е потребно идентификувањето на проблемот да биде за неколку секунди или милисекунди пред некој друг да го идентификува истиот. Бидејќи податоците секојдневно се зголемуваат, компаниите мора да ги анализираат податоците во скоро реално време. Така, како нов концепт што како тренд е воведен од страна на IBM[11] во форма на нова парадигма на големите податоци е стриминг пресметувањето (streaming computing). Во традиционалните системи прашањата се пишуваат во форма на статични податоци (пример, да се прикажат сите граѓани кои се во животна опасност). Со стриминг пресметувањето може да се обработи слично прашање, но податоците цело време би се менувале, односно корисникот би имал скоро реален, жив преглед на податоците соодветно како тие би биле преземени (пример од GPS уредите).

Ефективноста со големите податоци во суштина е извршување на анализа наспроти големината и разновидноста на податоците во скоро реално време. Денес малкумина размислуваат за брзината на големите податоци, бидејќи брзината веќе е вградена во новите технологии, односно претставува дел од инвестицијата на компанијата. Анализата на големите податоци, освен анализата на живи податоци во скоро реално време, претставува дополнителна можност за креирање на модел што ќе предвидува одредени стратегии кои би биле од корист на компанијата. Целокупната идеја за големите податоци

[11] https://www.ibm.com/ky-en/marketplace/stream-computing

е да се осознае што сé може да се анализира и што може да се разбере од анализата на моделот.

Веродостојноста е поим што сé повеќе ги опишува големите податоци, поточно се однесува на квалитетот или вистинитоста на податоците. Алатките што управуваат со веродостојноста на големите податоци во најголем дел го отфрлаат шумот и ги трансформираат податоците во веродостојни.

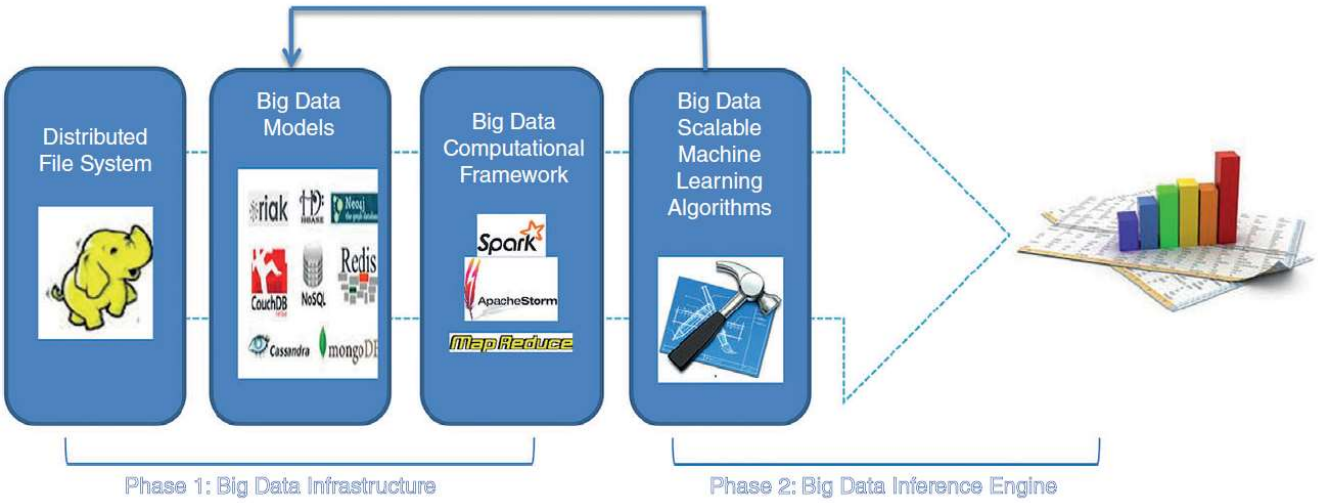
Платформата на големи податоци им ја нуди можноста на компаниите да ги анализираат податоците и да им се отворат поширoки гледишта на нивниот бизнис, корисниците итн. Компанијата во овој случај треба да направи ограничување на потребните и непотребните податоци за остварување на својата цел.

Имајќи ги во предвид карактеристиките на големите податоци, проблемите поврзани со квалитетот на големи податоци може да влијаат врз веродостојноста на податоците, со што се поставува прашањето дали податоците што ги поседува компанијата се реални и дали резултатите од нивната анализа се вистинити? Фактот што еден од тројца бизнис лидери не им верува на информациите што ги користат за идни одлуки претставува најбитен фактор да се обрни силно внимание на овој атрибут (Zikopoulos P. , 2015).

### 2.3 Инфраструктура на големите податоци

Големите количества на податоци, просторот и нивната анализа има потреба од децентрализација и спроведување на нова стратегија во пристапот кон традиционалните складишта на податоци што се совпаѓа со моделот познат како MAD (Magnetic - магнетно, Agile - агилно, Deep- длабоко). Големите податоци може да зачувуваат било каков тип на податоци без никаква трансформација, делувајќи како магнет што привлекува секакви типови на податоци. Инфраструктурата на големите податоци треба да биде брза и целосно скалабилна за справување со еволуцијата на податоците од различна природа (Слика 4). Истата треба да нуди детални репозиториуми за податоците со цел изработка на длабоки анализи. Со други зборови, инфраструктурата треба повеќе да претставува основна

стратегија којашто ќе понуди шема за обработување на прашања, отколку шема којашто би била базирана на дизајн (Gupta, 2016).



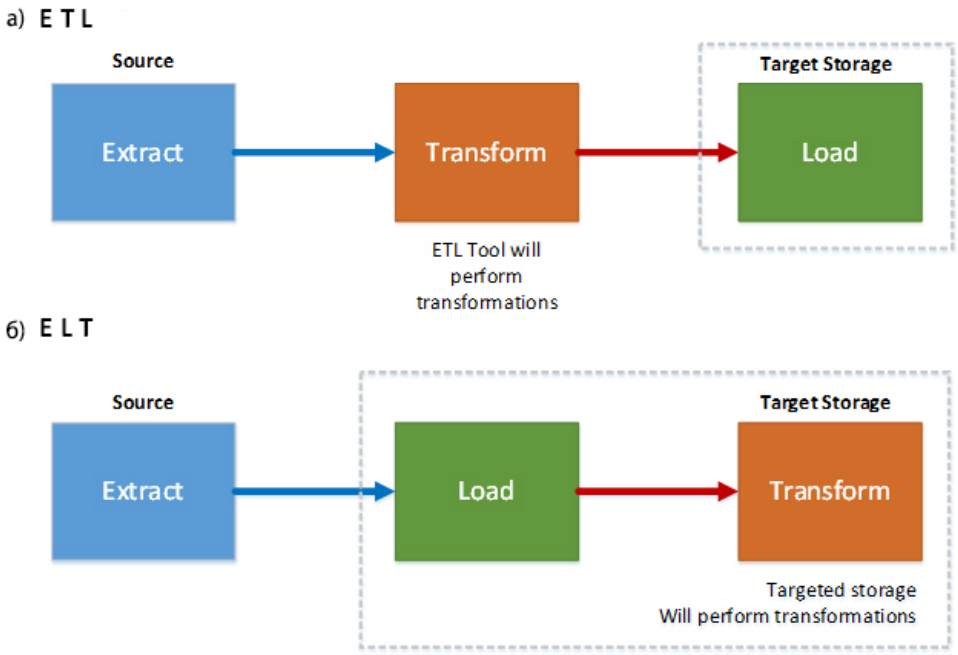

**Слика 4:** Инфраструктура на големи податоци (Gupta, 2016)

Компаниите се соочуваат со проблемот на собирање податоци од различни извори во најразличен формат и зачувување на еден или повеќе податочни системи. Најчесто податочниот систем не се состои од ист тип на податоци како изворниот, односно форматот на податоците е различен.

Екстракција, трансформација и полнење (Extract, Transform, Load – ETL), односно екстракција, прочистување/трансформација и полнење (Extract, Cleaning & Transform, Loading – ETCL) претставува процес за собирање податоци од различни извори, нивно трансформирање и прочистување според одредени бизнис правила и полнење во одреден податочен систем, односно репозиториум. Во процесот на трансформација најчесто се вклучуваат различни операции како што се: сортирање, филтрирање, агрегација, спојување на податоци, валидација итн, прикажано на Слика 5 а).

Процесот на екстракција, полнење и трансформација (Extract, Load, Transform - ELT) се разликува од ETL процесот во тоа што трансформацијата на податоците се извршува

директно врз крајниот податочен систем (Слика 5 б)). Во овој процес процесирачките капацитети на крајниот репозиториум се користат за трансформација на податоците.




**Слика 5:** а) ETL процес б) ELT процес (IntellifySolutions, 2017)

Во инфраструктурата на големите податоци, традиционалниот ETCL процес треба да се замени со ELT пристапот (Zikopoulos P. , 2015). Според тоа, за извршување на напредни анализи на инфраструктурата ѝ е потребен дистрибуиран податочен систем, NoSQL бази на податоци и рамки коишто се доволно скалабилни и ја поседуваат можноста за зачувување на големи податоци.

### 2.3.1 Дистрибуиран податочен систем

Дистрибуираниот податочен систем (Distributed File System - DFS) претставува скалабилен простор за пристап на големи податоци. DFS овозможува паралелно процесирање на податоци, а во случајот со големите податоци, Hadoop дистрибуираниот податочен систем (Hadoop Distributed File System – HDFS) што претставува најдобриот избор којшто е толерантен на грешки и им дозволува на корисниците да зачувуваат различни типови на

податоци што истовремено нуди редундантност и сигурност. HDFS е овозможен од Hadoop рамката што претставува програмска рамка во Java и поддржува обработка и складирање на големи количества на податоци во дистрибуирана околина, вклучувајќи различни алатки за големи податоци. Рамката се состои од две главни компоненти: HDFS системот за складирање на големи податоци и MapReduce за анализа на големите податоци. Hadoop исто така претставува рамка за извршување на анализи на големи податоци што овозможува сигурност и скалабилност со имплементација на парадигмата MapReduce.

MapReduce, претставува модел со паралелно програмирање, инспириран од Map и Reduce од функционалните јазици, што како модел е развиен за процесирање на големи податоци. Исто така претставува основа на Hadoop и извршува различни процесирања на податоци и аналитички функции. Парадигмата MapReduce повеќе се базира на користење на многу компјутери или ресурси отколку на зголемување на моќноста или капацитетот на складирањето на податоците на еден компјутер (Services, 2012). Основната идеја на MapReduce е да се подели задачата на повеќе нивоа кои ќе се извршуваат паралелно со цел да се намали времето за завршување на задачата.

Првата фаза на MapReduce е мапирање на влезните податоци во множество од клучеви/парови со вредности како излез. Map функцијата, како што веќе е напоменато, ја разделува задачата на помали фази, при што секоја фаза си има свој клуч, односно пар со одредена вредност. На овој начин, неструктурираните податоци (на пример текст) може да се мапираат во структуриран пар (на пример клучот да е збор од текстот, а вредноста е бројот на појавувања на тој збор). Излезот од оваа функција претставува влез во Reduce функцијата. Функцијата Reduce извршува складирање на податоците на излезот со помош на комбинација на сите вредности што делат исти клучеви за добивање на конечен резултат од задачата.

Функцијата MapReduce во Hadoop зависи од два јазли: Job Tracker и Task Tracker јазлите. Job Tracker јазлите се оние чии задачи се да ги дистрибуираат функциите Map и Reduce до сите расположливи Task Tracker-и. Функцијата започнува со доделување на дел од влезниот

податок на HDFS системот на јазолот Job Tracker. Од друга страна пак, јазлите на Task Tracker-от ги извршуваат задачите и комуницираат со Job Tracker-От. Комуникацијата помеѓу јазлите најчесто е помеѓу податоците и фолдерите на HDFS.

Hadoop претставува MAD модел, стекнувајќи ја популарноста за анализа на големи податоци преку читање на податоците во дистрибуиран податочен систем и извршување на паралелни пресметки на податоците со помош на MapReduce. Hadoop го добива својот магнетизам и брзина преку фактот дека податоците се прочитани во Hadoop само преку копирање на податоците во дистрибуираниот систем. На тој начин може да се читаат податоци од било какви извори на податоци, како и адаптирање на системите за нивна еволуција.

Во HDFS податоците се делат на блокови, а блоковите понатаму се дистрибуираат и се реплицираат низ различни јазли. HDFS поддржува write-once-read-many семантика на податоците. Блоковите најчесто се со големина од 64 MB, а истите се реплицирани трипати. HDFS кластерот пак ги извршува задачите во т.н. master-slave архитектура. Истиот се состои од еден именски јазол и повеќе податочни јазли. Именскиот јазол управува со податочниот именски простор и го контролира пристапот на податоците. Именскиот јазол исто така ги мапира податочните блокови до податочните јазли, а истовремено и креира, зачувува и ги враќа податочните блокови во правец на именските блокови.

Во втората верзија на Hadoop се вклучува и YARN[12] (Yet Another Resource Negotiator), технологија за управување со кластери, што ги одделува функциите за управување со ресурси од програмската форма, што значително ги подобрува перформансите за користење на различни модели за процесирање.

[12] http://searchdatamanagement.techtarget.com/definition/Apache-Hadoop-YARN-Yet-Another-Resource-Negotiator

### 2.3.2 Податочни модели за големи податоци

Појавата на големите податоци предизвика експлозија од различни типови на бази на податоци поради фактот што традиционалните бази на податоци не можат да се скалираат како што тоа се бара кај големите податоци. Моделите за големи податоци може да се поделат во два типа на модели: NoSQL и NewSQL податочни модели при што истите можат да се користат и во облик на традиционални бази на податоци. Теоремата CAP (Consistency - конзистентност, Availability - достапност, Partition tolerance – поделена толеранција)[13] покажува дека било кој мрежен податочен систем може да има најмногу две или три посакувани својства: конзистентност, висока достапност на податоците и толеранција на мрежни партиции. Теоремата исто така покажува дека било кој дистрибуиран систем не може симултано да гарантира конзистентност, висока достапност и да има толеранција во однос на поделбата за да може да се постигнат одредени размени во определено време за да се постигнат бараните перформанси и достапност на зададената задача. Дистрибуираниот систем не може да попушти на толерантноста на мрежните партиции, а сепак да остави простор за конзистентност и достапност.

Релационите бази на податоци повеќе се фокусираат на конзистентност отколку на достапност и строго ги почитуваат ACID својствата. Од друга страна пак, NoSQL базите се фокусираат повеќе на достапност отколку на конзистентност, почитувајќи ги BASE својствата. Овој тип на бази на податоци се фокусира на висок степен на скалабилно складирање на податоците при што менаџирањето на задачи се одвива на апликациски слој.

NoSQL се бази на податоци кои се хоризонтално скалабилни, може да се справат со голем број на типови на големи податоци и во основа се категоризираат во четири модели: модел со клучни зборови, модел со широки колони, документ-ориентирани и графички бази на податоци.

[13] https://dzone.com/articles/understanding-the-cap-theorem

Од друга страна пак, NewSQL како друг тип на модел на големи податоци е познат како нова генерација на системи за управување со скалабилни релациони бази на податоци. Овој модел го имплементира најдоброто од традиционалните бази на податоци и NoSQL. NewSQL се базира на релационен модел и нуди скалабилност на NoSQL почитувајќи ги ACID својствата, а истовремено поддржува и SQL прашања. Архитектурата на NewSQL е различна од релационите бази на податоци, односно поддржува техники за скалирање, како што се вертикално и хоризонтално партиционирање, односно партиционирање и кластерирање. Попознати имплементации на NewSQL се NuoDB и VoltDB.

Овие податочни модели меѓусебно се разликуваат, а нивниот избор зависи од задачата којашто треба да се изврши.

### 2.4 Анализа на големи податоци

Големите податоци претставуваат множество од податоци чија големина е над можноста постоечките софтверски алатки и системи за зачувување на податоци да ги складираат, менаџираат и процесираат податоците за одредено време.

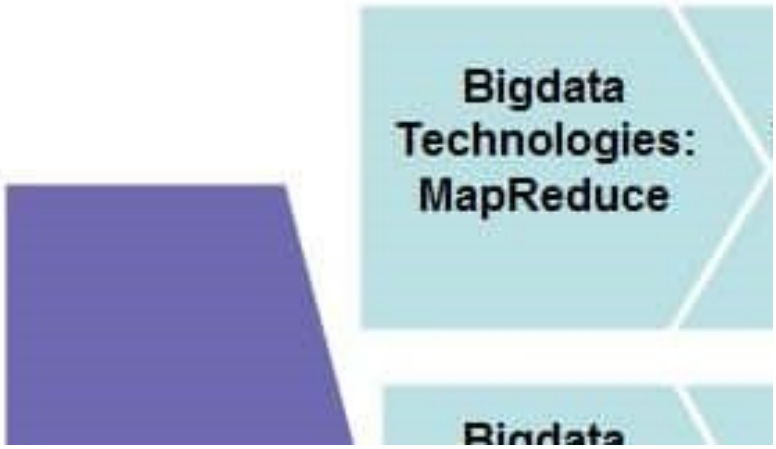
Едни од најголемите предизвици на кои се наидува кај големите податоци се зачувувањето, споделувањето, анализата и визуелизацијата на податоците. За таа цел потребни се напредни техники за анализа на ваков тип на податоци.

Компаниите денес зависат од анализата на податоци што непосредно зависи не само од големината на податоците, туку и од типот на податоци кои се проследуваат за понатамошна анализа.

Бизнис интелигенцијата може да понуди стандардни бизнис OLAP извештаи, односно ad-hoc извештаи кои се добиени од постоечки податоци. Ad-hoc анализите претставуваат анализа на статични податоци, кои имаат свое место во компанијата, но не претставуваат основа за подобрување на бизнисот.

Истовремено, се јавува потребата од примена на разни статистички анализи, предвидувања, оптимизација, креирање на модели за предвидување и текстуално рударење на големи множества од веќе достапни податоци. Се разбира, постојат и одредени проблеми со перформансите кои се појавуваат кога се користат големи податоци во релационен модел за анализи што би се користеле за анализа на големи податоци, односно за анализи кои би покажале резултати за иднината на компанијата.

Анализата на големи податоци треба да им помогне на компаниите да имаат подобар преглед кон водечките фактори на нивниот бизнис преку вклучување на технологија за големи податоци во компанијата (Слика 6). Преку оваа технологија би се користеле напредни техники за анализа на многу големи множества на податоци. Тука спаѓаат масивни паралелно-процесирачки бази на податоци, бази на податоци во меморија, апликации за пребарување, гридови за податочно рударење, дистрибуирани податочни системи, дистрибуирани бази на податоци итн.




**Слика 6:** Архитектура за анализа на големи податоци[14]

[14] https://www.predictiveanalyticstoday.com/

Меѓу неколкуте предности на компаниите од користењето на анализата на големите податоци се вбројуваат:

- Забрзување на реализацијата на одредени цели на компанијата преку создавање на нови погледи со користење на анализата на големи податоци;
- Креирање на комплексни дизајни на модели за претскажување (хипотези);
- Можност за интеграција на големи податоци со користење на традиционални бази или други системи;
- Дистрибуиран Hadoop податочен систем за побрзо читање на податоци и подобра скалабилност;
- Добивање податоци за едно бизнис решение од повеќе извори и негова анализа;
- Развој на многубројни апликации за работа со големи податоци и анализи;
- Лингвистички анализи и екстракција на релевантни информации од лог податоци и социјални медиуми;
- Визуелизација, откривање и споделување на скриени статистики;
- Изготвување на ad-hoc извештаи со анализи и одговори во реално време;
- Одговори во реално време од неструктурирани податоци.

### 2.4.1 Алатки и методи за анализа на големи податоци

Со напредокот на технологијата и зголемувањето на протокот на податоци во и вон самите компании се појавува потребата за барање на нови начини за ефективно анализирање на податоците. Самото зголемување на обемот на податоците ја намалува брзината на нивната обработка за донесување на ефикасни одлуки користејќи ги стандардните техники за обработка и анализа.

Како резултат на ова се јавува потребата за развој на нови алатки и методи кои се специјализирани за анализа на големи податоци, вклучувајќи ја и архитектурата за зачувување и менаџирање на таквите податоци. Податоците и колекцијата на податоци, процесирањето и одлуките кои се донесени од податоците се само дел од ефектот на појавувањето и зголемувањето на обемот на големите податоци.

Предложената т.н. рамка за анализа и донесување на одлуки при обработка на големи податоци – B-DAD (Big – Data Analysis and Decisions) (Elgendy N. , 2013), што ги вклучува алатките и методите за анализа на големи податоци во самиот процес на донесување на одлуки, ги мапира различните складишта на податоци, самиот менаџмент, алатките за процесирање, анализа, методите, визуелизацијата и евалуацијата во различни фази од процесот на донесување на одлуки. Така, промените што се поврзани со анализата на големите податоци се опишуваат во три глобални области: архитектура и складирање на големи податоци, податочно и аналитично процесирање и анализа на големи податоци. Големите податоци многу брзо еволуираат, така што брзо се развиваат и нови алатки за нивна анализа.

### 2.4.2 Складирање и управување со големи податоци

Основната задача на компаниите кои имаат потреба од менаџирање на големи податоци е каде и како да се зачуваат податоците. Традиционалните методи за структурирани податоци и враќање на податоците обработуваат релациони бази на податоци, податочни центри или складишта на податоци (Data Warehouse). Податоците се прикачени и зачувани со помош на ETL-процесот, односно ELT-процесот, што претставуваат алатки кои ги екстрахираат податоците од надворешни извори, ги трансформираат податоците во облик прифатлив за потребните операции на компанијата и на крај го полнат складиштето на податоци. На тој начин податоците се чисти, трансформирани и категоризирани пред да бидат спремни за рударење или пак за искористување на некои од онлајн алатките за нивна анализа.

Околините за големи податоци се повикуваат на MAD моделот што се разликува од традиционалните аспекти за анализа на податоци, како што е на пример Enterprise Data Warehouse (EDW) околината. Пристапите на околината EDW примаат нови извор на податоци само кога тие ќе се прочистат и интегрираат во новата околина. Поради сеприсутноста на податоците, околините за големи податоци треба да бидат доволно достапни со цел да можат да ги привлекуваат сите извори на податоци, без разлика на нивниот квалитет. Исто така, поради зголемувањето на изворите на податоците, како и подобрувањето на нивните анализи, складирањето на големи податоци треба да овозможи анализа за брзо и лесно креирање и адаптирање на податоци. Истото може да се овозможи со брза база на податоци, чии логички и физички концепти може да се адаптираат во синхронизација со брзата еволуција на податоците. Додека за досегашните анализи на податоци се користат комплексни статистички методи, новите анализи би требало брзо да ги проучуваат големите множества на податоци, репозиториумите со податоци треба да бидат детални и истовремено да може да се користат софистицирани алгоритми за обработка на податоците.

Нерелационите бази на податоци, како на пример NoSQL, се развиени со цел да зачувуваат и управуваат со неструктурирани, односно нерелациони податоци. Тоа значи дека NoSQL базите на податоци имаат за цел масивно скалирање, флексибилност на моделот на податоци и поедноставно развивање на апликации. За разлика од релационите бази на податоци, NoSQL базите го одвојуваат процесот на управување од процесот на складирање на податоците. Овие бази се фокусираат на складирање на скалабилни податоци со високи перформанси, додека пак прашањата поврзани со самите податоци ги извршуваат на апликациско ниво, наместо преку прашања во некој специфичен прашален јазик карактеристичен за релационите бази на податоци.

Од друга страна пак, системот за управување кај нерелационите бази на податоци е во меморијата на серверот, на начин што се елиминираат влезно-излезните дискови и овозможуваат одговори во реално време од базата на податоци. Наместо користење на

физички дискови, примарната база на податоци може да се зачува во главната меморија. Ова резултира во подобрување на перформансите и дозволува развој на нови апликации.

### 2.4.3 Аналитичко процесирање на големи податоци

По процесот на складирање на големите податоци следува аналитичкото процесирање.

Во основа, постојат четири главни барања за аналитичко процесирање на големи податоци. Првото е читање на самите податоци. Кај традиционалните системи можно е попречување на мрежниот сообраќај во извршувањето на прашања во процесот на читањето на податоците, што значи дека што кај големите податоци ова време треба значително да се намали. Второто барање е брзо процесирање на прашањата за извршување на големи задачи во реално време, односно податоците прво треба добро да се структурираат за брзо процесирање на прашањата. Третото барање е поврзано со искористувањето на просторот за складирање на податоците. Ова се должи на зголемениот пораст на активностите на корисниците што ја зголемува потребата од скалабилност и компјутерска моќ, односно кај големите податоци мора да се внимава на ограниченият простор за зачувување на податоците. Четвртото барање е адаптивноста за извршување на динамични задачи. Множествата од големи податоци се анализираат од различни корисници, односно од различни активности, за различни цели и на различни начини, така што системот мора да се адаптира на непредвидливи динамични промени во процесирањето на податоци.

Откако големите податоци ќе се зачуваат, истите се управуваат и процесираат, а менаџерите при донесувањето на одлуки треба да ги извлечат само потребните информации за понатамошна анализа на податоците.

### 2.4.4 Методи за анализа на големи податоци

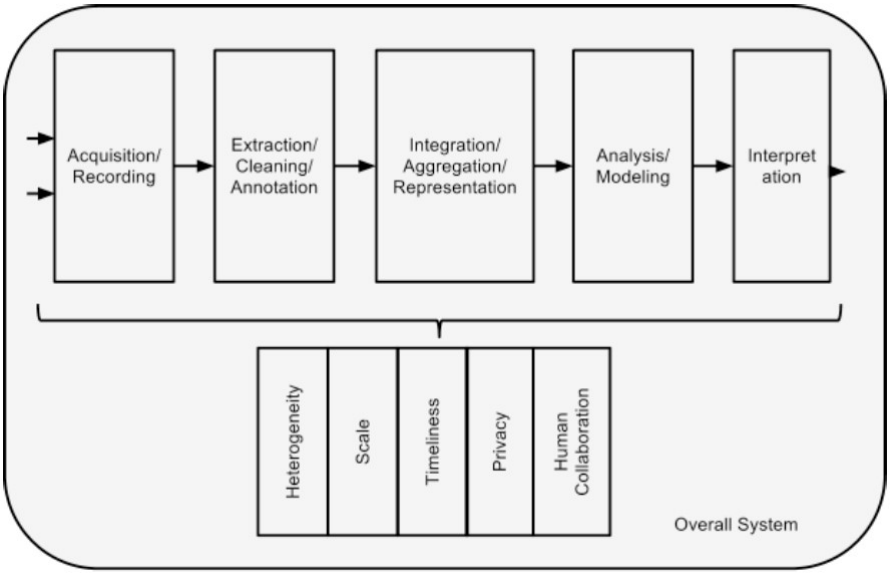
Денес целта не е само собирање на податоци, туку разбирање на нивното значење и важност за донесување на одлуки. Од друга страна, анализата на податоци претставува процес каде се користат разни алгоритми со цел анализа на множествата податоци и извлекување корисни релации, врски или информации. Анализата на податоците се

користи да се извлечат претходно непознати, корисни, важни и скриени релации и информации од големи множества на податоци. Затоа, анализата на податоци игра важна улога во истражувањето на податоците наспроти менаџерите кои донесуваат одлуки со кои сè повеќе обработуваат постоечки податоци, односно истите учат од веќе постоечките податоци за донесување на идни одлуки.

Меѓу напредните методи за анализа на податоци се вклучуваат правилата за асоцијација, кластерирање, класификација и регресија.

Меѓу најдобрите рамки за анализа на големи податоци се вбројува напредната визуелизација на податоци (Advanced Data Visualization – ADV) (Elgendy N. a., 2014). Меѓу најклучните предизвици за корисниците е како и во која форма податоците ќе бидат претставени со цел понатаму да можат донесувачите на одлуки правилно да ги анализираат податоците со цел преземање на конкретни акции. ADV комбинира различни методи за анализа на податоци преку интерактивна визуелизација со цел овозможување на детален преглед на податоците. Со зголемувањето на обемот на податоците и нивната комплексност, визуелизацијата како пристап е најкорисна кога многу малку знае за податоците што треба да се анализираат. ADV има предност во тоа што го користи човечкиот потенцијал и разумните способности со цел овозможување на анализа на податоците како на ниско така и на подетално ниво.

ADV овозможува брзи анализи, подобри донесувања на одлуки и поефективна презентација на резултатите преку статистички графикони и добар интерфејс. ADV може да ги скалира своите визуелизации од илјадници до милиони податоци; справувањето со различни типови на податоци е исто така голема доблест на визуелизацијата почнувајќи од обичните податоци, па сé до невронските мрежи. Со оваа рамка се овозможува анализата на бизнисот да ги истражува податоците во многу поширок спектар и низ различни извори во потрагата по правилно множество на податоци во реално време.




**Слика 7:** Анализа на големи податоци (Labrinidis, 2012)

Анализата на големи податоци се состои од неколку фази, прикажани на Слика 7 (Labrinidis, 2012). Многу корисници најчесто се фокусираат повеќе на фазата за анализа, односно моделирањето: иако оваа фаза е клучна, истата е некорисна без останатите фази. Најчестите предизвици за големите податоци се содржани во останатите фази. На пример, големите податоци мора да се водат според нивната содржина што најчесто е хетерогена. На овој начин мора да се следи кој е основниот извор на податоците со цел да се согледаат грешките. Исто така, потребно е да се креираат добри прашања во врска со податоците, при што понекогаш би се јавила потребата од одредени паметни системи.

Податоците се насекаде, но истите се зачуваат некаде на одреден извор, односно на одреден податочен систем. Повеќето од податоците не се од корист, а за правилна анализа потребно е истите да се филтрираат и компресираат. Еден од предизвиците е дефинирање на филтрите на начин што нема да отфрлат корисни информации. Друг проблем на кои се наидува е автоматското генерирање на мета-податоци што ќе го објаснат кои податоци и како истите се зачувани. За успешна анализа исто така потребно е да се обрни внимание на потеклото на податоците, што би била корисна информација зачувана во мета-податоците.

Зачуваните информации најчесто не се во формат којшто е спремен за анализа. По зачувувањето информациите потребно е да бидат опишани во структурирана форма која е соодветна за анализа. Истото може да преставува и проблем, бидејќи податоците може да бидат во форма на слики, видеа, односно да бидат во неструктуриран формат.

Поради хетерогеноста на податоците, не е доволно да се снимат истите, да се прочистат и да се постават на складиште на податоци. Анализата на податоци е еден комплексен процес, кој не бара само локализација, идентификација и разбирање на податоците. Кај анализата на големите податоци процесот треба да биде автоматизиран, што подразбира креирање на семантика и структура на податоците во форма којашто е разбирлива за компјутерот. Дури и помалата анализа што зависи само од едно множество на податоци, анализата би зависела од дизајнот на базата. Добра анализа на податоци има потреба од интелигентен дизајн, односно од добро развиени техники за одбегнување на одредени недостатоци при процесот на анализа.

Методите за креирање на прашања и рударењето на големи податоци се разликуваат од традиционалните статистички методи за анализа на помали податоци. Големите податоци најчесто се со шум, истите се динамични, хетегорени и недоверливи. Рударењето на големи податоци бара интегрирани, чисти, доверливи и пристапни податоци, декларативни прашања и интерфејси за податочно рударење, скалабилни алгоритми и околини за пресметување на големи податоци.

Кај социјалните медиуми се популаризира социјалното вмрежување и размената на содржини. Анализата на социјалните медиуми може да се користи за извлекување на корисни информации и за изработка на претпоставки. Анализата се базира на развивање и евалуација на информатички рамки и алатки со цел колекција, мониторирање, сумирање, анализа и виртуелизација на податоците од социјалните медиуми. Најбитно е дека анализата на социјалните медиуми ги разбира реакциите и разговорите помеѓу корисниците во онлајн заедниците, со цел извлекување на корисни податоци и врски од нивните реакции врз база на страните на социјалните медиуми.

Анализата на социјалните медиуми (Social Network Analysis – SNA) се фокусира на врската помеѓу социјалните ентитети. SNA ги мапира и ги мери формалните и неформалните врски со цел разбирање на објектите помеѓу засегнатите страни, како на пример кој - што споделува, кој - што знае или пак каква информација се користи за споделување итн.

SNA се разликува од традиционалната анализа на социјалните медиуми во тоа што се обидува да ги запомни социјалните врски помеѓу мрежата на корисниците, за разлика од традиционалната анализа што се обидува да анализира што корисниците на социјалните медиуми разговараат со цел да се откријат корисни врски, односно информации за корисниците. Оваа анализа најчесто се добива со текстуално рударење.

Текстуалното рударење се користи за анализа на документ или пак множество на документи со цел разбирање на содржината и значењето на информацијата што ја содржи. Денес, текстуалното рударење стана еден од најбитните фактори каде информацијата е во текстуална форма што најчесто доаѓа во нерелациона форма.

Рударењето на мислење (opinion mining) е сè повеќе важно кај податоци каде корисниците го изразуваат своето мислење, како што се содржини на блогови, мислење за продукти, форуми и социјални податоци од социјални медиуми како што се Twitter или пак Facebook. Оваа анализа најчесто се фокусира на разбирање на чувствата на текст со субјективна содржина преку дефинирање на опции или гледишта на одредени теми, односно класификација на гледиштата на позитивни или пак негативни. Анализата на рударење на мислења користи процесирање на природен јазик и анализа на текст со цел да се идентификува и извлечи информацијата преку изнаоѓање на зборови кои означуваат некое гледиште, како на пример врски помеѓу одредени зборови.

Можноста за анализа на големи податоци е бескорисна доколку корисникот не ја разбира анализата. Донесувачот на одлуки е оној кој треба да ги интерпретира резултатите од анализата. Овој процес најчесто вклучува испитување на сите претпоставки направени од анализата.

Не е потребно само добивање на одредени резултати. Потребно е да се добијат суплементарни информации кои ги објаснуваат како резултатите се добиени, врз база на влезните податоци. Со разбирањето за тоа како најдобро да се зачуваат информациите, да се обработат со разни техники за креирање на мета-податоци, може да се создаде одредена инфраструктура којашто ќе им ја додели на корисниците можноста за интерпретација на резултатите од анализата.

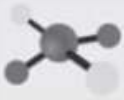
## 2.5 ПЛАТФОРМИ ЗА АНАЛИЗА НА ГОЛЕМИ ПОДАТОЦИ

### 2.5.1 Вовед

Со цел подобрување на бизнисот, компаниите треба да ги анализира веќе постоечките платформи за анализа на големи податоци чија анализа на структурирани податоци е ограничена. Често пати, соочувајќи се со проблемот на простор и анализа на многу податоци, компаниите треба да донесат одлуки за зачувување на важните податоци за понатамошна обработка.

Корисниците на платформа за големи податоци не би можеле да вршат анализата на големи податоци без одреден систем за менаџмент на релациони бази на податоци или пак без Hadoop; со други зборови, потребна е рамка за процесирање и разбирање на природен јазик.

Платформата за анализа на големи податоци може да обработува и живи податоци, така што од самиот почеток треба да знае како да обработува податоци во секое време. Платформата за анализа на големи податоци треба да ги поддржува сите типови на податоци. Исто така, потребно е да ги извршува сите зададени пресметки што се потребни за анализата на податоците. На Слика 8 се претставени неопходните технологии на платформата за анализа на големи податоци.

| | | |
|---|---|---|
| Discover, Explore, and Navigate Big Data Sources |  | Federated Discovery, Search, and Navigation 1 |
| Land, Manage, and Store Huge Volumes of Any Data |  | Hadoop File System<br>Hadoop Processing<br>24/7 Data Collection 2 |
| Structure and Controlled Data |  | In-Memory Analytics, MPP Analytic Appliances for Extreme Performance 3 |
| Analyze Unstructured Data |  | Text Analytics Engine<br>Content Analytics<br>Video/Audio Analytics 4 |
| Analyze Data in Real Time |  | Stream Computing<br>24/7 Decisioning 5 |
| Rich Library of Analytical Functions and Tools |  | In-Database Analytics Libraries<br>Big Data Visualization 6 |
| Integrate and Govern All Data Sources |  | Integration, Data Quality, Security, Lifecycle Management, MDM 7 |

**Слика 8:** Пристапи и технологии што треба да ги поседува одредена платформа за големи податоци (Zikopoulos P. , 2015)

Кога станува збор за подобрување на анализата, основниот недостаток е  незнаењето за нештата кои би требало претходно да се знаат. Секој проект што содржи големи податоци би требало да започне со прашањето „што би можело да се знае за податоците“, односно би требало да се знае што сè е потребно пред донесување големи одлуки за податоците.

Процесот на анализа започнува со разбирање на изворите на податоци, откривање за тоа кои податоци се достапни до одреден извор и разбирање на квалитетот и релациите до други елементи. Овој процес, познат како „Откривање на податоци“ овозможува креирање на точните модели за анализа вклучувајќи ги и стратегиите за пресметување на податоци. За олеснување при процесот на откривањето на податоците, платформата за големи податоци треба да ја има можноста за откривање на податоците во самото место. Исто така, платформата треба да ги има овозможено опциите на индексирање, пребарување и навигација низ различни извори на големи податоци.

Кога се инвестира во процесот на откривање на податоци, компанијата веднаш има бенефит. Бенефитот вклучува зголемена продуктивност со подобар пристап на потребните информации и подобрување на донесувачите на одлуки.

Кога станува збор за големина и разновидноста на податоците, во областа на големите податоци сè повеќе се зборува за Hadoop. Кај податоците што се зачувани во Hadoop системот се јавува потребата од високо специјализирани вештини за програмирање, додека пак за повеќето алгоритми потребен е паралелизам со цел нивно извршување во Hadoop, иако понекогаш се смета дека Hadoop би требало да биде не повеќе од обичен систем за подготовка на податоци за нивните проекти со складишта на податоци.

Платформата за анализа на големи податоци, од друга страна пак, мора да биде полиглот, односно да може да користи NoSQL бази на податоци, Hadoop и RDBMS (Relational Database Management System - RDBMS) технологии. Овие технологии поддржуваат различни т.н. зони на информационен менаџмент, од кои одредена зона бара податоците да бидат високо структурирани и контролирани.

Доколку одлуката е работа со Hadoop и големи податоци, базите на податоци мора да бидат колона-ориентирани во внатрешната меморија припаѓаат на т.н. in-memory колона ориентирани бази на податоци. Предноста во пресметувањето во внатрешната меморија е во брзината на процесирање на податоците за анализа и извештаи.

Дел од податоците треба да бидат сместени на Hadoop репозиторими, но некои мора да се сместат во структурирана и контролирана околина, односно во одредена алатка за анализа. Традиционалните архитектури ги одделуваат аналитичките од податочните околини. Софтверот за анализа има своја инфраструктура и ги враќа податоците назад од складишта на податоци или од други системи со цел извршување на комплексна анализа. Платформата за анализа на големи податоци мора да ги извршува и процесирањето на податоците и комплексните анализи. Понекогаш потребно е да обработат бази на податоци големи од неколку петабајти на одреден модел, а притоа добивање на резултати од комплексни анализи и тоа без реплицирање или семплирање на податоците.

Платформата за анализа на големи податоци мора да го има својството на менаџирање, зачувување и враќање на неструктурирани и структурирани податоци, односно мора да постојат алатки за анализа и истражување на неструктурирани податоци. Од друга страна пак, компаниите кои сакаат да ги подобрат анализите на големите податоци мора да ја имаат можноста да анализираат така како што се генерираат податоците, а потоа да преземат некоја акција. Идејата е да се добие резултат пред да се зачуваат податоците. Оваа опција е т.н. анализа на податоци во „движење“ односно анализа на стриминг податоци. Во зависност од периодот на денот, големината на податоците може сериозно да варира. За таа цел потребно е платформата за големи податоци не само да поддржува анализа на „мирни“ податоци, туку и ефективно менаџирање на живи податоци кои брзо ја зголемуваат капацитетот и овозможување на анализа на тие податоци.

Друга основна цел на платформата за анализа на големи податоци е можноста за намалување на времето на аналитичкиот циклус, односно времето што е потребно за откривање и трансформирање на податоци, креирање и развој на модели, и анализа и добивање на резултати. Платформата потребно е да поддржува повеќе аналитички итерации со цел да се забрза развојот на моделите. Иако ова е основната цел на платформата за анализа на големи податоци, истата потребно е сè повеќе да се фокусира на подобрување на продуктивноста на развивачот на апликации преку лесно откривање на податоците, креирање и развој на модели, визуелизација на резултатите и интеграција со одредени апликации со кориснички интерфејс. Анализата е дисциплина којашто сè уште се развива, па затоа сè уште не постои посакуван пристап за креирање и визуелизација на модели. Најбитно е да се посвети време за визуелизација на податоците, односно на типот за визуелизација на големи податоци како што се Steamgraphs, Treemaps, Gantt charts и многу други. Како останати пакети за анализа, платформата за големи податоци би требало да ги подржува и алатките како SAS, SPSS и R, како и множество од алгоритми како што се алгоритмите за машинско учење (Shalev-Shwartz, 2014).

### 2.5.2 Преглед на платформи за големи податоци

Софтверите за анализа на големи податоци се фокусираат на ефективна анализа за големи множества на податоци. Анализите истовремено им помагаат на компаниите да имаат подобар преглед на своите податоци преку претворање на податоците во информации со висок квалитет отворајќи им ја можноста за подобро решавање на одредена ситуација.

Платформата за големи податоци е платформа на информатичката технологија која има својства, функционалности на една апликација за големи податоци, претставена во решение за креирање, развој, калкулации и менаџирање на големи податоци. Софтверот за анализа на големи податоци помага при откривање на скриени релации, непознати корелации, трендови на пазарот, предности на корисниците и многу други корисни информации од различни множества на податоци.

Во прилог следува преглед на пет познати платформи за големи податоци.

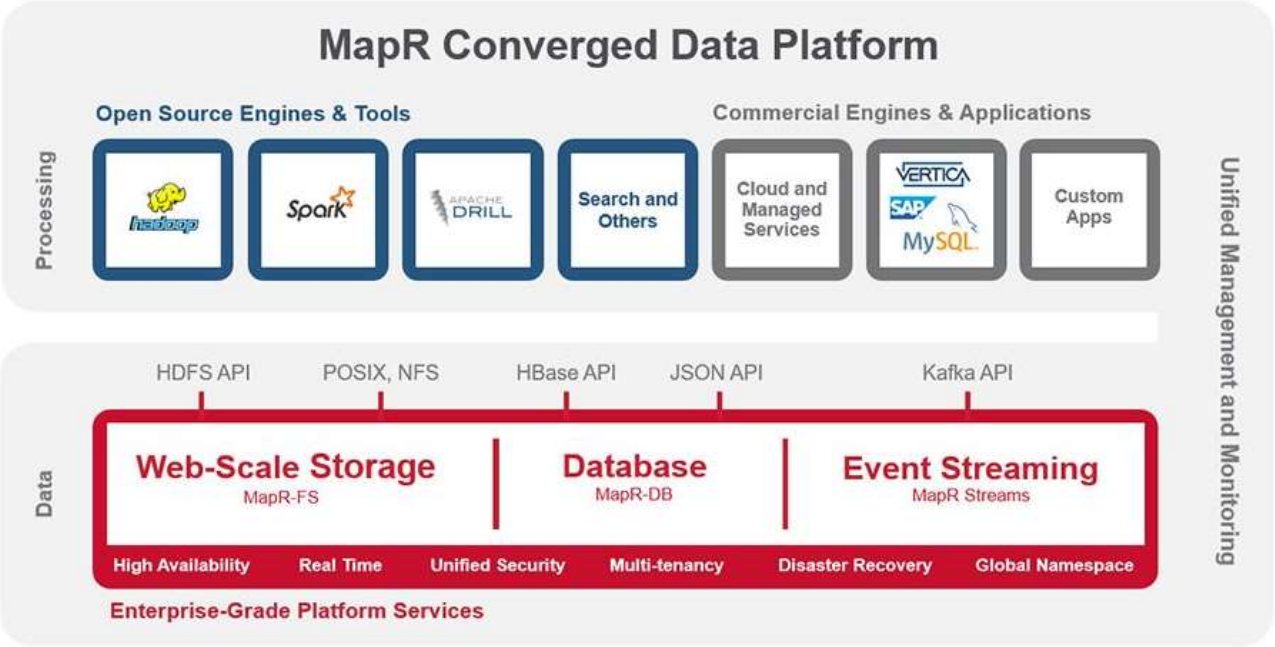
#### 2.5.2.1 MapR converged Data Platform

MapR претставува дистрибуирана платформа за сигурно зачувување и процесирање на податоците (MapR Technologies, Inc. All Rights Reserved , 2017). Пакетите на MapR содржат open-source проекти што нудат интерактивни и апликации во реално време. MapR платформата за податоци (Слика 9) има интегрирано Hadoop, Spark и Apache Drill со капацитет на бази на податоци за реално време, стриминг податоци и скалабилни простори слободни за наредната генерација на апликации за големи податоци. Оваа платформа на компаниите им нуди сигурност, безбедност и извршување на операции низ податоците во реално време со намалено искористување на ресурсите на хардверот и останати оперативни трошоци со кои корисникот би се соочил при анализа на податоци од страна на традиционалните бази на податоци. MapR исто така користи стандардни API-а, со цел и новите апликации да ги запазат основните стандарди за анализа на податоци.

Платформата нуди добар кориснички дизајн со лесен пристап кон сервиси за менаџирање на податоци, вклучувајќи висока достапност, заштита на податоци, кластери кои сами се

обновуваат, контрола за пристап, обработка на живи податоци, менаџмент и мониторирање на податоци.

Основната цел на оваа платформа е да им ја овозможи на корисниците опцијата за избор на соодветните алатки за анализа на нивните податоци. Меѓу алатките се вклучуваат Hive, Pig, Apache Hbase, Mahout и многу други.




**Слика 9:** MapR converged Data Platform

### 2.5.2.2 IBM Big Data

IBM Big Data (Слика 10) платформата прикажува нови прегледи во податоците на компанијата со драстично намалување на трошоците за простор и одржување (IBM Official website, 2017). Платформата врши дистинкција на големите податоци на познатите четири димензии: големина, разновидност, брзина и веродостојност. Преку ова, компанијата би ја имала можноста за следење на сите промени, вклучувајќи го Hadoop, напредни стриминг анализи итн. Можностите и за менаџмент со складишта на податоци кои ги нуди оваа платформа им овозможуваат на корисникот добивање на анализи со голема брзина.

Компаниите можат да ја зголемат моќта на Apache Hadoop преку алатки за забрзување, анализа, визуелизација, перфромансност и безбедност.

Одличниот систем за раководење со содржина на оваа платформа исто така и овозможува на компанијата детален животен циклус на содржината и менаџирање со документите преку т.н. рентабилна контрола на постечките и новите типови на содржина со помош на скалабилноста, безбедноста и одржливоста.

Во ера кога секојдневно компаниите генерираат многу податоци, платформата е повеќе од потребна. Платформата ефикасно може да доставува анализи во реално време при константни промени на податоците што се во постојано движење, овозможувајќи дескриптивни и предиктивни анализи за одлуки во реално време. Корисниците може да ги снимаат и анализираат сите податоци, преку цело време и во било кој дел од денот. Со помош на стриминг пресметувањето, корисниците зачувуваат помалку, анализираат повеќе и платформата нуди подобри и побрзи анализи.

**Слика 10:** IBM Big Data

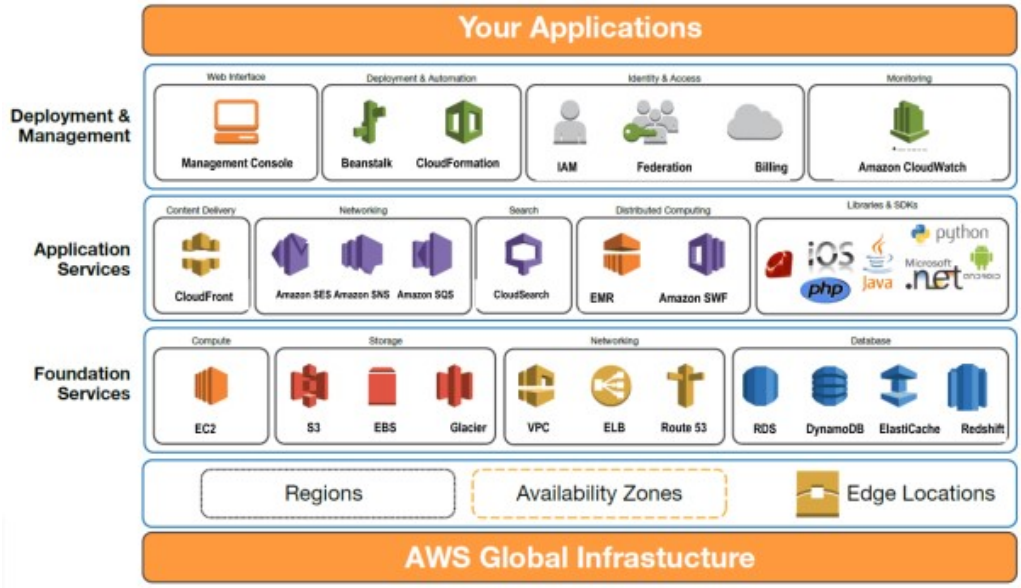
### 2.5.2.3 Amazon Web Service

Amazon Web service (Слика 11) како платформа нуди голем број на глобални cloud продукти, меѓу кои се алатки за пресметување, зачувување, анализа, мрежа, менаџмент, IoT, безбедност и други (Amazon Web Services, Inc. or its affiliates, 2018). Овие сервиси им помагаат на компаниите брзо да обработуваат податоци, да им се намалат трошоците и да

им се зголеми скалабилноста. Платформата може да ја користат големи компании преку нудење на различни продукти. Корисниците на оваа платформа имаат можност да изберат меѓу различните продукти со цел остварување на одредена цел.

Продуктот на Amazon Web Elastic Compute cloud, претставува веб сервис кој нуди променлив капацитет во облакот. Продуктот е дизајниран со цел да го зголеми пресметувачкиот простор на облакот за корисниците преку лесниот веб интерфејс. Овој сервис им нуди на компаниите комплетна контрола врз нивните ресурси преку користењето на околината на Amazon.

Amazon Simple storage service од друга страна пак претставува сервис кој преку својот обичен интерфејс им нуди на корисниците зачувување и враќање на било која количина на податоци од било кој извор. Aurora пак, е MySQL алатка за релациони бази на податоци што комбинира брзина и достапност на комерцијални бази на податоци.




**Слика 11:** Amazon Web Service

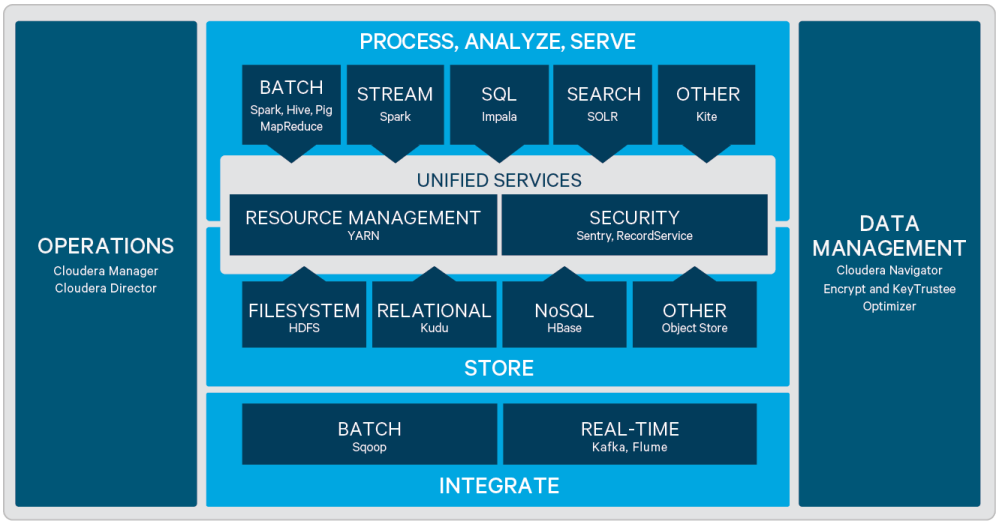
### 2.5.2.4 Cloudera Enterprise BigData

Cloudera Enterprise (Слика 12) претставува најбрза, најлесна и најбезбедна платформа за големи податоци (Cloudera, 2018). Преку оваа платформа корисниците може да добијат

резултати од било кои податоци преку било кои околини, благодарение на скалабилноста на платформата. Cloudera исто така нуди безброј можности со податоците од компаниите, и тоа од оперативни или аналитички бази поставувајќи ги во една околина. Платформата е идеална за сите кориснички потреби; со други зборови, корисниците може да им ги достават податоците до научниците на податоци со цел градење и развој на податочни модели.

Cloudera ја модернизира IT архитектурата на компанијата преку овозможување на ELT и напредни SQL анализи за добивање на резултати, истражување на податоците и бизнис интелигенција. Преку добивањето на резултати во реално време, корисникот може да ги следи живите податоци, што е овозможено преку стриминг апликациите понудени од самата платформа.

Како лидер во можностите што ги нуди, Cloudera им помага на корисниците да го подобрат својот бизнис. Cloudera Manager е една од административните Hadoop алатки што претставува дел од платформата. Истата содржи интелигентни и единствени опции за мониторирање, што драстично ги забавува операциите на кластерот. Алатката е дизајнирана за голема флексибилност, а истата многу брзо се интегрира со многу third-party алатки и со новите компоненти на Hadoop.



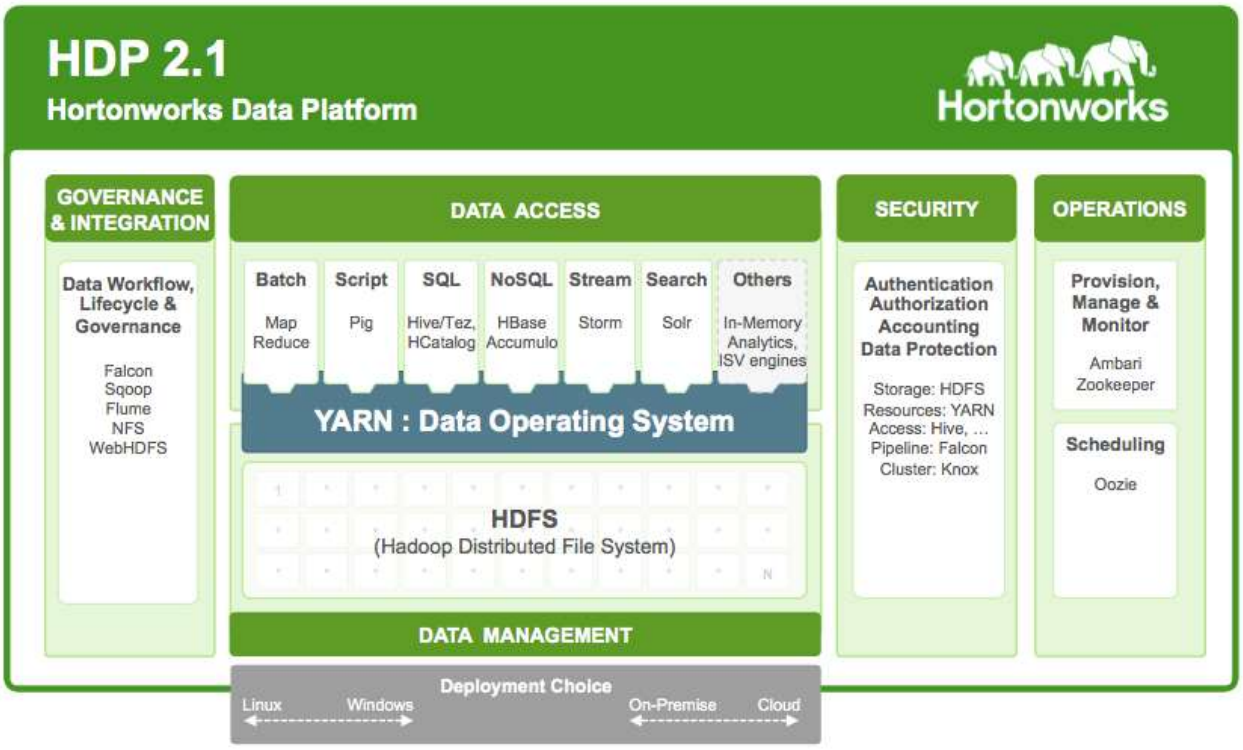

**Слика 12:** Cloudera Enterprise BigData

#### 2.5.2.5 Hortonworks

Hortonworks (Слика 13) претставува безбедна, open-source платформа за големи податоци, вклучувајќи Hadoop дистрибуција што е базирана на единствена и централизирана архитектура (Hortonworks , 2017). Платформата ги адресира комплетните потреби на корисникот преку апликации во реално време, а истовремено доставува робустни анализи кои го забрзуваат процесот на донесување на одлуки и иновации. Hortonworks вклучува голема процесирачка сила што им овозможува на корисниците интеракција со истите податоци на повеќе начини истовремено, што би значело дека апликациите понудени од страна на платформата може да ги обработуваат податоците избирајќи го најдобриот начин: од обични податоци до интерактивни SQL бази на податоци, или пак пристап со мала латентност преку NoSQL бази на податоци.

Како додатоци кои се поддржани од страна на Apache Spark, Storm и Kafka, платформата поддржува различни студии на случаи за науката за податоци, пребарување и стриминг пресметување. Повеќето од овие додатоци се безбедни преку користење на рамката за менаџирање на протокот на податоците од и во Hadoop. Структурата, заедно со многу други алатки за лесна и автоматска обработка на податоци е клучна за одличната интеграција на Hadoop во модерните архитектури на податоци, како и солидна за соработка со многу други менаџмент провајдери што се занимаваат со обработка и анализа на податоци.

Hortonworks исто така им обезбедува на корисниците интеграција и проширување на нивните безбедносни решенија преку нудење на конзистентно решение на модерната архитектура на податоци на компаниите.




**Слика 13:** Hortonworks Data Platform

# 3 МАШИНСКО УЧЕЊЕ

*We are drowning in information and starving for knowledge. — John Naisbitt "Megatrends" - 1982*

Едно од основните прашања што се поставува со развојот на компјутерските науки е прашањето како компјутерите да се научат сами да учат. Доколку може да се сфати начинот како компјутерите да се програмираат сами да учат, односно автоматски да се подобруваат со стекнување на одредено искуство, драстично би се зголемил напредокот во информатичката технологија. На пример, доколку компјутерите учат од медицински записи за тоа кои третмани се најефективни за нови болести; доколку паметните системи на домаќинствата учат од искуството за оптимизирање на трошоците за енергија врз база на одредени релации за потрошувачка на енергија во самиот дом, успешното разбирање за тоа како компјутерите учат, би отворило нови погледи на користењето на компјутерите и нови нивоа на компетентност и нивно прилагодување. Детално разбирање на алгоритмите за обработка на информации и за машинско учење може да доведе до подобро разбирање на способностите за учење на човекот како индивидуа.

Сè уште се поставува прашањето како тоа компјутерите можат да научат речиси исто како што учат и луѓето. Постојат различни алгоритми кои се доволно ефикасни за одредени типови на задачи за учење, со што започнува и теоретското разбирање за учењето. Исто така постојат и многу практични апликации што покажуваат успешно учење на компјутерот, со што започнува и ерата на комерцијализирање на ваквите типови на апликации. За проблеми како што се препознавањето на говор, алгоритмите кои се базирани на машинско учење денес ги надминуваат сите останати пристапи што се развиени до сега. Во областа податочно рударење многу често се користат алгоритмите за машинско учење за откривање на скриени релации; знаења од големи комерцијални бази на податоци што се користат во апликации, како на пример за кредитирање, финансиски трансакции, медицинска евиденција итн. Како што човековото разбирање за компјутерите се развива, така и машинското учење игра голема улога во компјутерската наука и технологија.

За решавање на даден проблем потребен е алгоритам. Поимот алгоритам се дефинира како секвенца на инструкции кои треба да се извршат со цел претворање на влезните податоци во излезни. Од друга страна пак, за решавање на други проблеми нема потреба од користење на алгоритам, како на пример, разграничување на емаил пораките со спам содржина од обичните пораки. Влезниот податок претставува обичен емаил документ, односно одредено множество на текстуални податоци. Излезните податоци би биле со одговор да или не, во зависност од тоа дали пораката е спам или не е. Во ваков случај не би се знаело како да ги претвориме влезните податоци во излезни, единствени промени што можат да се приметат се промените во време од индивидуален испраќач до испраќач.

Со напредокот на компјутерската технологија, отворена е можноста за зачувување и процесирање на голем број на податоци, како и пристап до нив кои се наоѓаат на пример на големи растојанија преку компјутерска мрежа. Денес, на многу лесен начин може да се прочистат илјадници пораки што содржат со спам содржина, а од нив исто така може да се научи што поточно ги прави различни од обичните пораки. Со други зборови, потребно е компјутерот, односно машината, автоматски да ги оддели пораките со помош на одреден алгоритам. Во овој случај нема потреба да се учи, како на пример за сортирање на броеви веќе постојат различни алгоритми, но сепак постојат апликации за кои нема соодветен алгоритам.

Постои процес што ги објаснува податоците кои се посматраат. Иако не се познати деталите за процесот, како на пример однесувањето на корисниците, сепак тој процес не е комплетно настанат случајно. Луѓето во супермаркетите не одат да купуваат случајни работи. Постојат одредени шеми, односно релации на податоците.

Процесот не може целосно да се идентификува, но може да се креира добро и корисно приближување до целосниот процес. Тоа приближување можеби нема да опиши сè, но сè уште може да опиши одреден дел од податоците, со други зборови, целосниот процес не може да се идентификува, но може да се одредат одредени релации и врски помеѓу податоците. Овој процес претставува основата на машинското учење. Одредените

релации, односно врски помеѓу податоците може да помогнат во креирањето на претпоставки, по претпоставка дека иднината, барем блиската иднина можеби нема да биде толку поразлична од минатото кога одредени податоци се обработуваа, така што идните претпоставки може да се земат за точни.

Машинското учење не е само проблем со бази на податоци, машинското учење исто така е дел од вештачката интелигенција. Интелигентен систем е систем којшто во променлива околина ја има способноста за учење. Доколку системот учи и се адаптира на примени, дизајнерот на системот нема потреба за давање на решенија за сите можни ситуации.

Машинското учење исто така помага при барање на решенија кај проблеми како препознавање на говор, препознавање на облици и роботика. Машинското учење ги „програмира“ компјутерите на оптимизација врз база на одреден дел од податоци или одредено искуство од минатото. Постои модел во кој се дефинирани некои параметри, а учењето се извршува за оптимизација на параметрите на моделот преку користење на тренирачко множество или пак минато искуство. Моделот само тогаш може да биде предвидлив односно да дава претпоставки за иднина, или да стане дескрпитивен, односно да стекнува знаење од податоците.

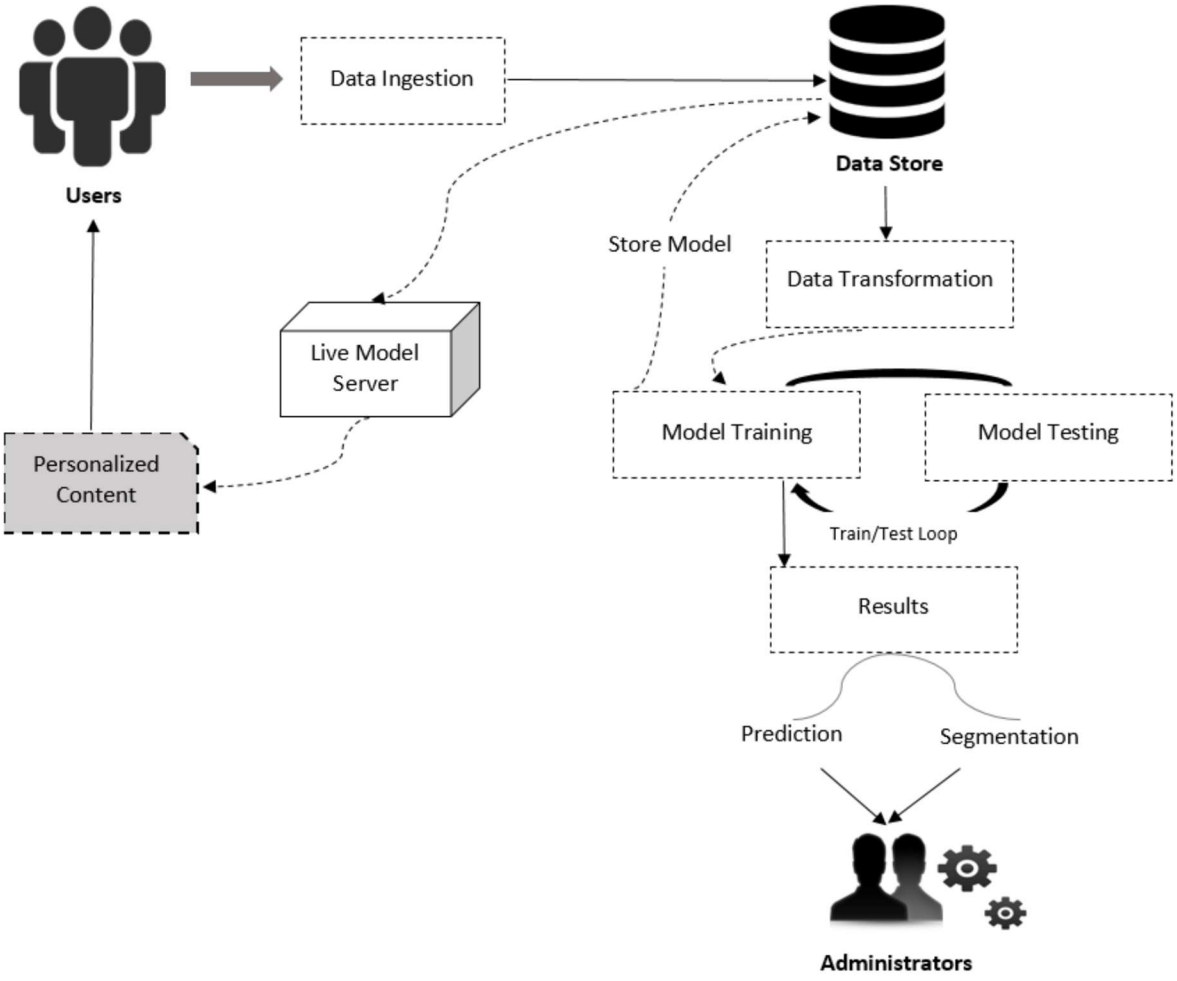
Машинското учење при градењето на математички модели во основа ја има теоријата на статистиката, бидејќи основната задача е да се извлече заклучок од одреден примерок. Улогата на компјутерската наука е двострана: прво, во процес на тренирање потребни се ефикасни алгоритми за решавање и оптимизирање на проблемот, како и зачувување и процесирање на големо количество на податоци кои се достапни. Како второ, откако моделот би го поминал процесот на учење, потребно е репрезентативниот примерок и решението од алгоритмот да бидат ефикасни. Во одредени апликации ефикасноста на алгоритмот или учењето, поточно просторната и временската комплексност на апликацијата е исто толку битна колку и точноста на претпоставките.

## 3.1 ПРОЦЕС НА МАШИНСКО УЧЕЊЕ

Во рамките на процесот со машинско учење креирањето на модели и искористувањето на машинското учење со користење на големи податоци претставува можеби еден од најголемите предизвици. Постои предодредена патека, односно процес за креирање на добри резултати од алгоритмите за машинско учење. Но, при креирањето на модели потребно е добро разбирање на проблемот којшто треба да се реши. Постојат најразлични поедноставени или посложени чекори за дефинирање на процесот на креирање на модел со машинско учење. Основните пет чекори се следните:

- Дефинирање на проблемот
- Подготовка на податоците
- Избор на соодветен алгоритам
- Подобрување на резултатите
- Презентација на резултатите

При решавањето на одреден проблем истиот мора да се дефинира преку одредена листа на потреби преку истражување на слични проблеми, претставено на Слика 14.




**Слика 14:** Дијаграм на процеси за машинско учење[15]

Подготовката на податоци е процес на сумирање и визуелизација на атрибутите преку одредени хистограми. Податоците може да се истражат преку најразлични начини, како на пример преку пребарувач или некои лог податоци. Откако податоците се анализирани истите треба да се зачуваат, најчесто во форма на сурови податоци. Процесот на зачувување на податоците може да биде комплексен, бидејќи пред сè зависи од тоа што треба да се постигне со креирање на моделот. Тука, постои избор помеѓу повеќе податочни системи како што се HDFS, Amazon S3 итн. Исто така, тука прашањето на избор е дали тоа ќе бидат релациони бази на податоци или пак дистрибуирани бази на податоци.

[15] https://www.cetic.be/

Повеќето од моделите на машинско учење работат со параметри од базата на податоци, кои најчесто се претставени во нумеричка форма за да може на крај да се добијат прецизни резултати. Како што веќе е напоменато, податоците најчесто се добиено во сурова форма. На пример, може да се зачувуваат одредени настани на корисник, колку пати корисникот прегледувал одредена веб страна итн. Во овој чекор потребно е да се зачуваат податоци од надворешни извори како што е на пример локацијата на корисникот.

За користење на суровите податоци во моделот, потребно е пред-процесирање на податоците што вклучува:

- Филтрирање на податоци – доколку е потребно да се креира модел од подмножество на сурови податоци, потребно е да се креира одреден критериум за филтрирање.

- Состојба со податоци што недостасуваат, нецелосни или испрекинати податоци – најчесто базите на податоци од самиот почеток на одреден начин не се чисти. Тука спаѓаат податоци што недостасуваат или пак нецелосни податоци кои се јавуваат како резултат на одредени хардверски или софтверски грешки. За таа цел потребно е да се филтрираат податоците што не се соодветни.

- Справување со аномалии и грешки – грешните податоци може лошо да влијаат врз резултатите од анализата на моделот, па во овој случај потребно е да се филтрираат истите да се избегнат грешни резултати на крај.

- Спојување на различни извори на податоци – понекогаш е потребно спојување на податоците, на пример, податоци од настаните за секој корисник со различни извори на податоци како што се профил на корисникот како гео-локација, време итн.

- Агрегација на податоци – кај одредени модели потребна е агрегација на влезните податоци како на пример сумирање на бројот на различни типови на настани по корисник

По процесот на пред-процесирање на податоците, потребно е да се трансформираат податоците во одредена форма што е соодветна за моделите за машинско учење. За многу типови на модели, претставувањето на податоците може да биде во векторска форма или во форма на матрица со нумерички податоци. Во овој случај се појавуваат следните предизвици:

- Претставување на податоците од текстуална форма во нумеричка;
- Извлекување на корисни параметри од текстуалните податоци;
- Обработка на слики или податоци во аудио форма;
- Често пати е потребно да се конвертираат нумеричките податоци во текстуални со цел да се намали бројот на вредности за одредена променлива;
- Трансформација на нумерички параметри;
- Нормализацијата и стандардизацијата на нумерички параметри овозможува сите влезни податоци да бидат конзистентни во скалирањето.

Моментот кога податоците се подготвени за тренирање, следува фазата на тренирање и тестирање на моделот. Оваа фаза вклучува избор на модел. Тука, потребно е да се избери најдобриот пристап за моделирање или најдобриот параметар за даден модел, што во суштина претставува тестирање со различни модели и тестирање со најдобрите параметри. Процесот во најопшт случај вклучува вкрстена валидација.

Доколку се работи со нерелациони бази или пак големи податоци, потребно е да се креира иницијален циклус за работа со помал репрезентативен примерок од податоците или да се креира процес на селекција на модели со користење на паралелни методи.

По процесот на оптимизација на податоците и моделот потребно е соочување со задачите кои се веќе во одреден систем со цел добивање на прецизни претпоставки. Во овој процес се вклучува истражување и поставување на тренирачкото множество во одреден

централизиран систем каде што може реално да се согледа процесот. На овој начин се освежува моделот на тренирачкото множество.

Во процесот на креирање модел многу важно е да се следат перформансите на системот на машинско учење, со цел лесно откривање на недостатоците на моделот. Точноста на моделот и претпоставките најдобро може да се увиди доколку истиот е поставен на реален систем. Најчестото прашање кое се поставува е изборот на метрика за евалуација што би била најсоодветна за оценување на точноста на моделот што соодветствува со бизнис интелигенцијата на проблемот. Тука, точноста на моделот е најважна, но постојат и други фактори кои се битни за компанијата како што се корисничкото искуство, активноста итн.

Во вистинските системи треба да се следи точноста на метриките на моделот и бизнис метриките. Во овој случај потребни се многу експерименти со различни модели за да може да се оптимизираат бизнис метриките преку одредени промени во моделот. Доколку се работи со живи податоци, тестирањето би било ризично и доста скапо, бидејќи доведува до произлегување на многу грешки, слаба обработка на податоци итн.

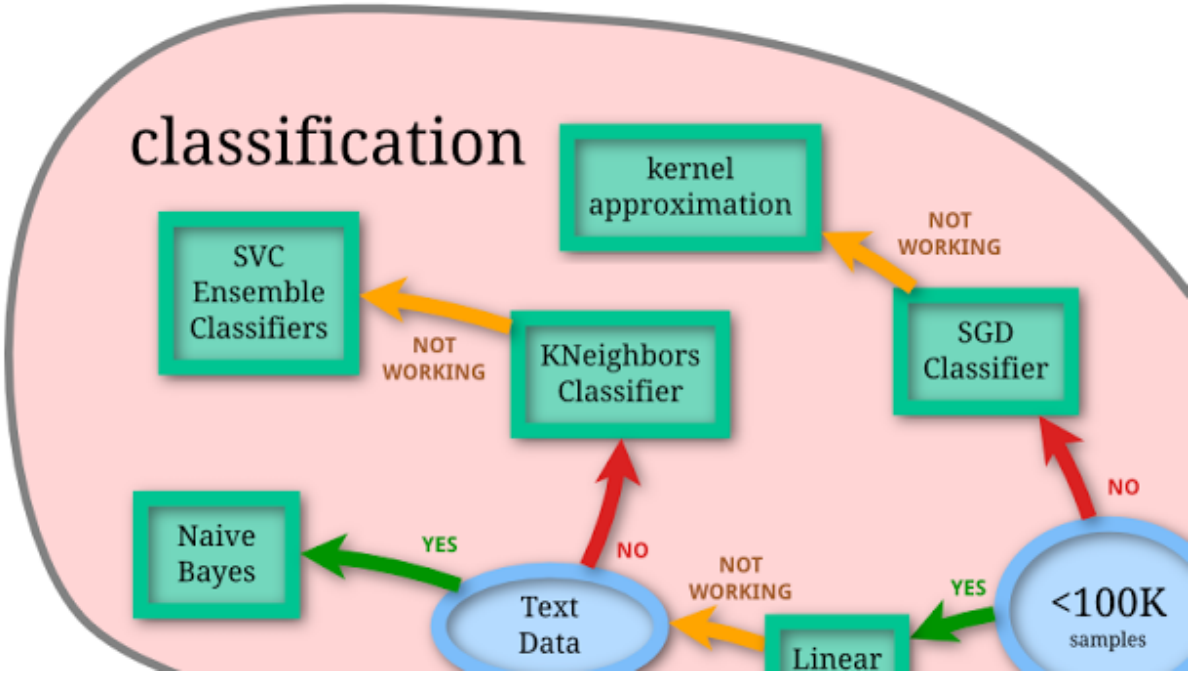
Тука, доста важен фактор е реакцијата на моделот, односно повратниот одговор од самиот модел. Тоа претставува процес каде предвидувањата за моделот се „хранат“ со корисничкото однесување. Во реален систем, тренирачките податоци влијаат врз моделот преку создавање на потенцијалното однесување на корисникот.

Постојат механизми со кои може да се ограничи негативното влијание од одговорот на моделот што вклучуваат непристрасни податоци од тренирачкото множество со мал дел на податоци кои доаѓаат од корисници што не се изложени на моделот, при што се подобруваат перформансите на системот.

Фазата избор на соодветен алгоритам за машинско учење е тесно поврзана со проблемот што треба да се реши при креирање на моделот. Прв чекор е категоризација на проблемот што се одвива во два под-чекори: категоризација на влезните податоци и категоризација на излезните податоци. Доколку во моделот за влез се користат податоци со лабели

(дефинирани, именувани, лабелирани податоци), тоа би претставувало проблем на надгледувано учење, во друг случај станува збор за проблем со ненадгледувано учење. Од друга страна пак, доколку излезните податоци се нумерички, станува збор за регресионен проблем. Доколку се класни податоци проблемот е класификациски.

Откако ќе се категоризира проблемот, потребно е да се идентификуваат алгоритмите за решение проблемот (Слика 15). Вклучувањето на алгоритмите во моделот би значело идентификување на параметрите и поставување на одреден критериум за евалуација.

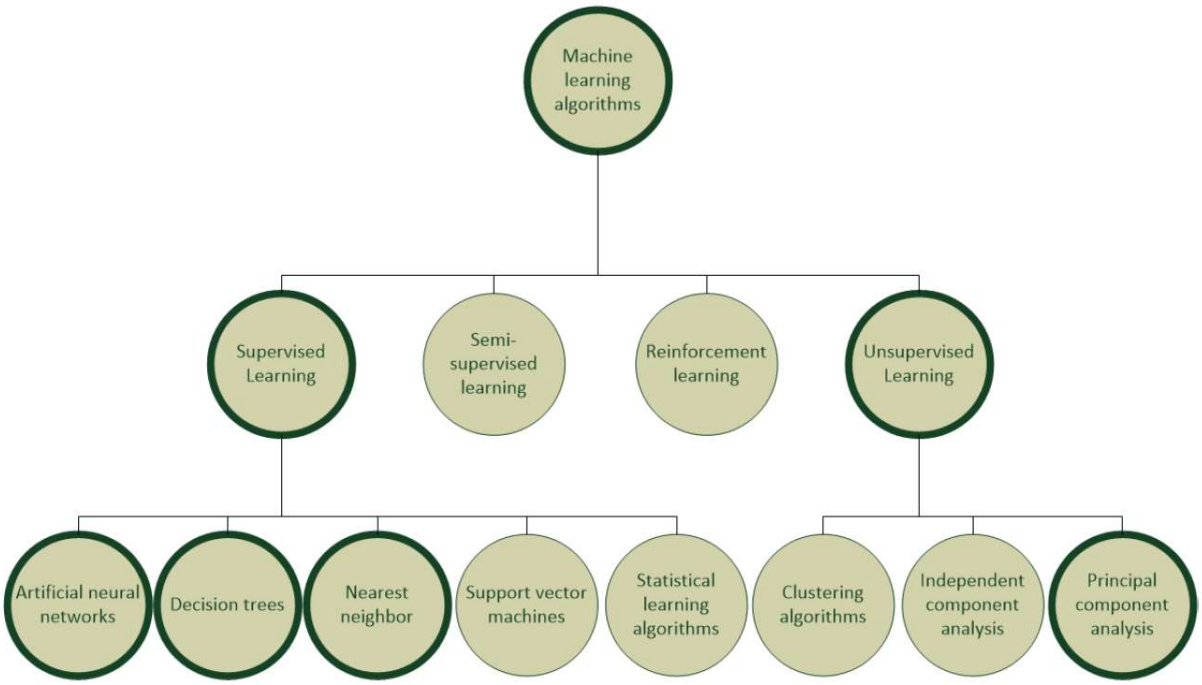



**Слика 15:** Дијаграм за избор на соодветен алгоритам развиен од страна на авторите на scikit-learn.org [16]

[16] http://scikit-learn.org/

### 3.2 ТИПОВИ НА МАШИНСКО УЧЕЊЕ

Во науката за податоци машинското учење најчесто е поделено на два типови на учење (Слика 16). Предвидливото, односно надгледуваното учење има за цел да учи преку мапирање на влезни податоци $x$ во излезни податоци $y$, преку дадено множество од влезно-излезни парови $D = \{(x_i, y_i\}_{i=1}^{N}$, каде што $D$ преставува тренирачко множество, додека пак $N$ претставува бројот на тренирачки примероци.




**Слика 16:** Типови на алгоритми за машинско учење (Folkers, 2016)

Секој тренирачки влезен податок $x_i$ е претставен во форма на $D$-димензионален вектор од броеви. Овој вектор, односно овие броеви се нарекуваат својства, атрибути или пак коваријации кои може да бидат и комплексен објект, како што се слика, израз, емаил порака, график итн.

Типот на излезниот податок $y_i$ т.н. променлива на одговор може да биде од било кој тип, но најчесто се претпоставува дека оваа променлива е категорична односно номинална променлива од некое конечно множество, каде што $y_i \in \{1, \ldots, C\}$, или пак, би можела да

се претстави во скалар со реални вредности. Во случај кога вредноста на $y_i$ е претставена во категорична форма, проблемот е т.н. класификација или пак како препознавање на облици, а доколку пак $y_i$ е претставен со реална вредност, проблемот е т.н. регресија. Постои и трет случај, односно ординална регресија што се случува кога множеството од решенија $Y$ има некое природно подредување.

Вториот тип на машинско учење е т.н. описно, или ненадгледувано учење. Во овој тип на учење, дадени се само влезни податоци, односно $D = \{x_i\}_{i=1}^{N}$. Целта на овој пристап е да се најдат „интересни“ шеми, односно релации во податоците. Пристапот понекогаш е познат и како откривање на знаења. Овој проблем е доста слабо дефиниран, бидејќи не е познато какви типови на шеми помеѓу податоците треба да се бараат, а тука постои и метрика за грешка којашто би се користела, за разлика од надгледуваното учење каде што се споредуваат претпоставките $y$ за дадени надгледувани вредности $x$.

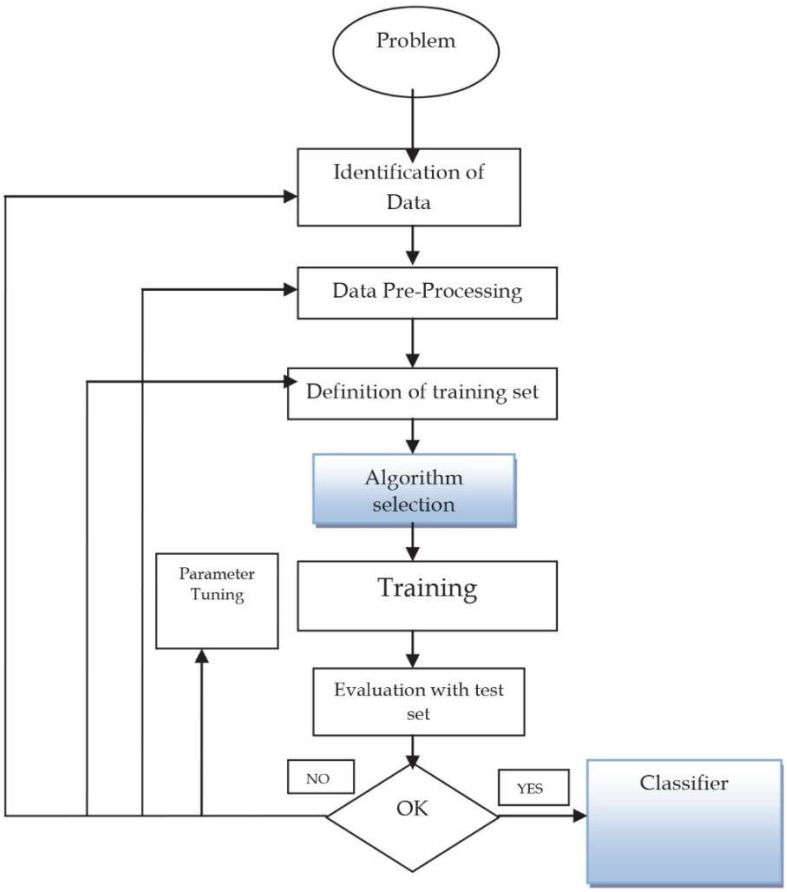
Постои и трет вид на машинско учење, познат како засилено учење. Овој тип на учење најчесто се користи за да се тренира учењето како да се однесува кога се сигнализира одредена награда или казна.

### 3.2.1 Надгледувано учење

Како што веќе е напоменато, во првиот тип на алгоритми за машинско учење односно во алгоритмите со надгледувано учење, алгоритмите може да научат одредена задача преку одредени тренирачки податоци. Надгледуваното учење е најчестата техника за тренирање на невронски мрежи и дрва на одлуки. И двете техники зависат од информациите дадени од преддефинираните класификатори. Кај невронските мрежи, класификацијата се користи со цел одредување на грешката на мрежата, а понатаму да ја научи мрежата да ја минимизира истата, додека пак кај дрвата на одлуки, класификаторите се користат за одредување на кои атрибути ги нудат најдобрите информации за решавање на одредена задача.

#### 3.2.1.1 Класификација

Кај алгоритмите за класификација, целта е да се научи како да се мапираат влезни податоци $x$ во излезни податоци $y$, каде што $y \in \{1, \ldots, C\}$, каде што $C$ е бројот на класи (Слика 17). Доколку $C = 2$, овој тип на класификација се нарекува бинарна класификација каде $y \in \{0,1\}$, а доколку пак $C > 2$, класификацијата е т.н. повеќекласна класификација. Доколку пак лабелите меѓусебно не се исклучуваат се нарекува повеќелабелна класификација. Најчесто кога се користи изразот класификација се однесува на повеќекласната класификација со единствен излез.




**Слика 17:** Класен дијаграм на алгоритмите за класификација

Друг начин за решавање на одреден проблем е преку т.н. приближна функција. По претпоставка дека $y = f(x)$ за одредена функција $f$, целта е да се научи како да се оцени функцијата $f$ со дадено тренирачко множество, а потоа да се креираат претпоставки

користејќи $\hat{y} = \hat{f}(x)$. Целта е да се креираат претпоставки на нови влезни податоци, односно податоци кои не се претходно видени.

За решавање на двосмислени случаи, како резултат пожелно е да се добие одредена веројатност. За почеток, веројатноста е распределена на сите лабели, преку даден влезен вектор $x$ и тренирачко множество $D$ со веројатност $p(y|x, D)$. Во основа, ова претставува вектор со должина $C$. Доколку е потребно да се избере помеѓу повеќе модели, претпоставката би била $p(y|x, D, M)$, каде што $M$ го означува моделот.

Одреден веројатен излез може да се пресмета преку најдобрите погодоци и вистинските лабели преку:

$$\hat{y} = \hat{f}(x) = \underset{c=1}{\overset{C}{argmax}}\, p(y = c|x, D)$$

Ова равенство ја претставува најверојатната лабела и се нарекува мода или модус на дистрибуцијата $p(y|x, D)$ што е позната како MAP предвидување (maximum a posteriori).

Класификацијата е најчесто користената форма на машинско учење и се користи за решавање на доста интересни и тешки реални проблеми.

На пример, во 2011 IBM го претстави својот компјутерски систем Watson којшто го победи првакот во квизот на знаење Jeopardy[17]. Во својата студија на случај (IoT one, 2017), IBM за Watson искористи најразлични техники, но една од најдобрите е дека користи модул што го одредува коефициентот на точност на одговорот на прашањето. Системот само го одбира одговорот доколку е доволно уверен дека истиот е точен. На сличен начин, Google неодамна има воведено систем познат како SmartASS (Smart Ad Selection System - SmartASS) кој ја предвидува веројатноста на кликот на корисникот што се базира на историјата на пребарување и други кориснички додатоци кои се специфични за огласи (Cade, 2010). Оваа

[17]https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/

веројатност е позната како CTR (Click Through Rate - CTR) и може да се користи да го максимизира очекуваниот профит.

Друг пример за класификација е развиен од статистичарот Ronald Fisher (Murphy, 2012). Целта е на неговата студија е да се тренира моделот за да научи да прави разлика помеѓу различни типови на цвеќиња ирис. Ботаничарот за овој модел ги разграничи цвеќињата според четири клучни карактеристики: должина и ширина на листовите и должина и ширина на листовите од цветот. Преку одредена визуелизација на податоците, може лесно да се разграничат класите на секое од цвеќињата, а понатаму со користење на одреден метод на машинско учење, моделот ќе направи разлика помеѓу различните типови на цвеќиња.

Класификацијата се користи во решавање на проблемот со класифицирање и филтрирање на документи. Во овој тип на класификација, целта е да се класифицира документот, како што е на пример веб страна или емаил порака во една од $C$ класите, односно да се пресмета $p(y = c|x, D)$, каде што $x$ претставува текст. Специфичен случај на овој тип на класификација е филтрирање на емаил пораки со спам содржина, каде што вредностите за класите ако се спам се $y = 1$ или ако се хам се $y = 0$.

Повеќето од класификаторите претпоставуваат дека влезниот вектор $x$ има фиксна големина. Најчест начин на претставување на документи во векторски формат на параметри е преку користење на т.н. „торба од зборови[18]“. Основната идеја на овој формат е дека $x_{ij} = 1$ ако и само ако зборот $j$ се појавува во документот $i$. Доколку се користи оваа трансформација на секој документ во множеството на документи, се добива бинарна матрица од документи × каде што се појавува саканиот зборот. Во суштина, проблемот со класификација на документи се сведува на барање на мали промени во одредена низа од битови, како на пример, најчестите пораки со спам содржина имаат голема веројатност да ги содржат зборовите „купи“, „евтино“, итн.

[18] https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Класификацијата на слики е меѓу најпознатите методи. Препознавањето на ракопис претставува класификација каде сликата се состои од изолирани букви и броеви. Множеството на податоци во овој метод се нарекува MNIST [19](Modified National Institute of Standards - MNIST), што претставува множество од 60.000 слики за тренирање и 10.000 слики за тестирање на цифрите од 0 до 9, испишани од различни луѓе. Големината на сликите е 28x28 пиксели и имаат сиви вредности во опсег од 0 до 255.

Еден од потешките проблеми е препознавањето на објекти во сликата и се нарекува детекција на објекти или локализација на објекти. Најпознат е пристапот на препознавање на лица. Еден од пристапите на овој проблем е поделба на сликата на многу помали делови кои се преклопуваат на различни локации и ориентации, а потоа да се класифицира одреден дел во кој е одредено дали има текстура на лице или не. Потоа системот ги враќа локациите каде што веројатноста на најдено лице е голема. Ваквите системи се вградени во скоро сите дигитални камери, така што камерата автоматски го одредува центарот за автоматски фокус врз лицата.

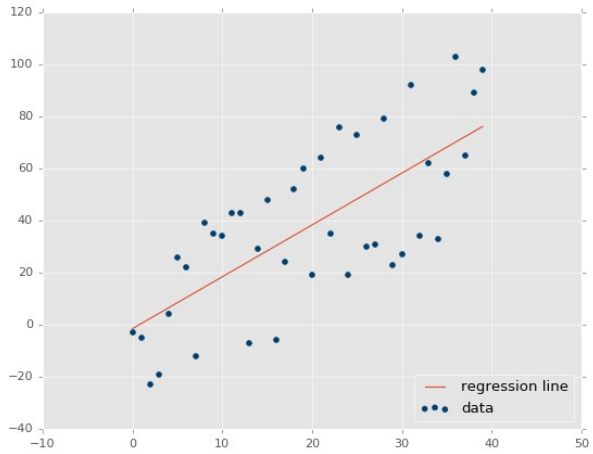
Откако се пронајдени лицата, процесот продолжува со методот препознавање на лица, што означува оценување на идентитетот на личноста. Во овој случај, бројот на класите односно лабелите мора да биде голем. Исто така, освен лицето потребни се и други методи за одредување на идентитетот на личноста, односно потребни сè повеќе детали како на пример одредената форма на коса на личноста може да биде важен фактор во донесувањето на одлука за идентитетот итн.

#### 3.2.1.2 Регресија

Во поглед на класификацијата, регресијата е иста, но се разликува во променливата на одговор, што во случајот на регресија е непрекината. На Слика 18 е прикажан пример за

[19] https://en.wikipedia.org/wiki/MNIST_database

регресија: како влез се претставени влезни податоци кои припаѓаат на множеството реални броеви, $x_i \in \mathbb{R}$, и излезни податоци кои исто така се реални вредности, $y_i \in \mathbb{R}$.




**Слика 18:** Пример за регресија[20]

Регресијата во основа претставува моделирање на врските помеѓу променливите што се итеративно преработени преку користење на метод на грешка во претпоставките кои се креирани во моделот. Методите за регресија се во основа статистички методи.

Постојат најразлични примери за регресија, меѓу кои:

- Предвидување на цените на утрешниот пазар на труд преку однапред поставени услови на пазарот и други споредни информации;
- Предвидување на годините на корисникот преку гледање одредено видео на интернет;
- Предвидување на локацијата на 3Д простор на раката на робот, контролирана преку сигнали до различни мотори;

[20] https://pythonprogramming.net/regression-introduction-machine-learning-tutorial/

- Предвидување на температурата на било која локација во одредена зграда преку користење на податоци за времето, сензори на вратите итн.

### 3.2.2 Ненадгледувано учење

*„Кога човекот учи да гледа, никој не му кажува кои се точните одговори – тој само гледа. Дури и кога мајката одговара со – тоа е куче – претставува многу мала информација за тоа што се гледа. Многу би биле среќни доколку добиваме неколку бита на информации, дури и еден бит во секунда. Визуелниот систем на човечкиот мозок има $10^{14}$ невронски врски, додека пак човекот живее за $10^{9}$ секунди. Така што, нема потреба да учиме еден бит во секунда. Потребни ни се околу $10^{5}$ битови во секунда. А, постои само едно место од каде што може да се научат толку информации: од самиот извор*.“ – проф. Geoffrey Hilton, 1996, University of Toronto.

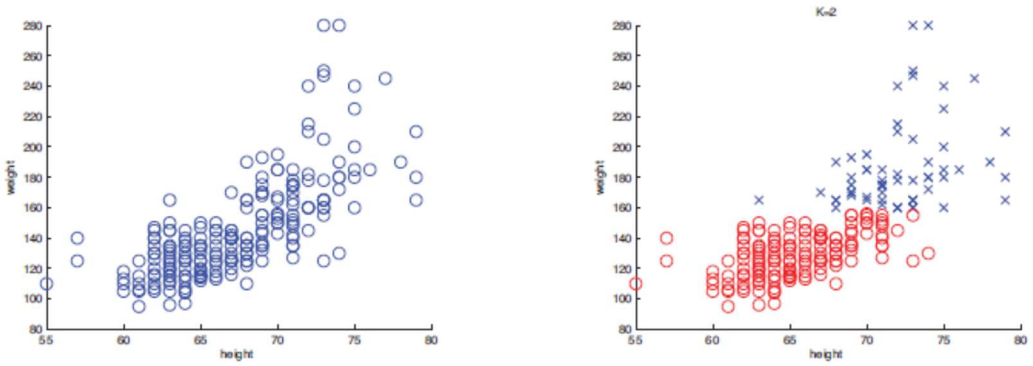
Основната задача на ненадгледуваното учење е да бара скриени структури во нелабелираните податоци, познато како откривање на знаење. Една од најпознатите форми на ненадгледувано учење е кластерирањето.

За разлика од надгледуваното учење, не се знае кој е посакуваниот излез за секој влезен податок. Наместо тоа, се креира проценка на густина, односно креираните модели се претставуваат во следната форма: $p(x_i|\theta)$. Кај ненадгледуваното учење постојат две основни разлики од надгледуваното учење. Прво, испишаната форма е $p(x_i|\theta)$, наместо $p(y_i|x_i, \theta)$ што покажува дека надгледуваното учење користи кондиционална проценка на густина, додека пак кај ненадгледуваното учење се користи некондиционална проценка на густината. Второ, $x_i$ претставува вектор на параметри, па затоа потребно е да се креираат модели на претпоставка со повеќе варијации. Во надгледуваното учење, $y_i$ најчесто претставува обична променлива којашто треба да се предвиди. Тоа значи дека за најчестите проблеми со надгледуваното учење, потребно е да се користат едноваријантни модели на веројатност, при што проблемот е драстично поедноставен.

Ненадгледуваното учење е повеќе својствено за човечкото и животинското учење, што е исто така повеќе употебувано од надгледуваното учење, бидејќи нема потреба од дополнително човечко влијание за рачно маркирање на податоците. Маркираните податоци, односно лабелите не само што тешко се добиваат, туку и содржат релативно малку информации, не доволно за да ги оценат параметрите на комплексни модели.

#### 3.2.2.1 Откривање на кластери

Основниот пример за решавање на проблем со ненадгледуваното учење е проблемот со кластерирање на податоците во групи. На пример, на Слика 19 податоците од графиците се прикажани во две димензии, претставувајќи висина и тежина на група од 210 луѓе. Не се исклучува и можноста да постојат и повеќе кластери, подгрупи, иако не се знае точно колку од нив. Нека $K$ претставува бројот на кластери. Првата задача е да се оцени распределувањето преку бројот на кластери $p(K|D)$ што покажува дали постојат подпопулации во самите податоци. За поедноставување на моделот во распределбата $p(K|D)$, $K^* = \arg max_K\, p(K|D)$. Кај надгледуваното учење се знае дека има две класи (машко или женски), но во случајот на ненадгледувано учење може да се одберат колку и да било кластери. Изборот на моделот со точната комплексност се нарекува селекција на модел.




**Слика 19:** а) Множество на податочни точки претставени во дводимензионален подпростор, вграден во 3Д простор б) 2Д репрезентација на податоците, генерирани преку pcaDemo3d (Murphy, 2012)

Наредната задача е да се оцени кој кластер припаѓа на која точка. Нека $z_i \in \{1, \dots, K\}$ претставува кластерот каде што е доделена точката $i$. Од друга страна, $z_i$ претставува скриена или пак латентна променлива, бидејќи тренирачкото множество не е посматрано. Може точно да се заклучи каде припаѓа секоја податочна точка илустрирана на Слика 18 (б) преку равенството:

$$z_i^* = \arg max_k p(z_i = k | x_i, D)$$

Постојат повеќе примери за кластерирање, меѓу кои:

- Во астрономијата, системот за автоматски класи откри нов тип на ѕвезди врз основа на одредени мерки за астрофизичко кластерирање (Schubert, 1988);
- Во e-commerce најчесто се користи кластерирање на корисници во групи врз база на нивните купени производи, пребарувањето на интернет итн (Berkhin, 2006);
- Во биологијата, каде што се користи кластерирање на податоците од цитометријата во групи со цел откривање на различни подпопулации на клетки.

#### 3.2.2.2 Откривање на латентни фактори

Кога корисниците работат со податоци од високи димензии, често пати е од корист да се намали димензионалноста преку проектирање на податоците во простор со пониски димензии. Ова се нарекува димензионална редукција.

Идејата на ова техника е дека иако податоците може да се појавуваат во високи димензии, можно е да има мал број на степени на варијација, соодветно на латентните фактори. На пример, кога се моделира појавување на слики со лица, можно е да има неколку основни фактори кои го опишуваат поголемиот дел од варијабилноста, како што се осветлувањето, позата, идентитетот итн.

Во случајот кога се користи како влез во други статистички модели, како на пример репрезентација на мали димензии, често пати се добиваат поточни претпоставки бидејќи

најчесто фокусот е на суштината на објектот, филтрирајќи ги несуштинските параметри. Исто така, корисни се репрезентациите во мали димензии за овозможување на брзо пребарување на најблиските соседи, додека пак дводимензионалните проекции се корисни за визуелизација на податоци со големи димензии.

Најчестиот пристап на редукцијата на димензиите се нарекува анализа на главни компоненти (Principal Components Analysis – PCA). Пристапот претставува ненадгледувана верзија на линеарна регресија со повеќе излези, каде што се набљудува високо-димензионалниот излез $y$, но не и излезот $z$ од мала димензија. Па на тој начин моделот ја има формата $z \longrightarrow y$, односно пресликувањето е инвертно и се извлекува заклучок за латентиот ниско-димензионален излез $z$ од набљудуваниот високо-димензионален излез $y$.

Редукцијата на димензионалноста и PCA се користат во многу области, како на пример:

- Во биологијата се користи PCA за интерпретација на ген во низа со податоци, имајќи го во предвид фактот дека секое мерење во суштина е резултат на многу гени кои се меѓусебно поврзани по однос на нивното однесување;
- Во процесирањето на природен јазик, што се нарекува лантентна семантичка анализа за враќање на документи;
- Во процесирањето на сигнали со цел одделување на сигналите во нивните различни извори;
- Во компјутерската графика, во проекција на податоци кои претставуваат движечка слика во помала димензија за креирање на анимации;

### 3.3 Метрики за евалуација

По самата имплементација на алгоритмите во моделите потребно е да се изврши оценување на ефективноста и ефикасноста на самите алгоритми. Подобрувањето на

резултатите од метриките за евалуација значи и подобрување на самите модели. Помеѓу метриките за евалуација спаѓаат: коефициент на корелација (Correlation coefficient), средна апсолутна грешка (Mean absolute error – MAE), квадратен корен од средната грешка (Root mean square error – RMSE) итн.

### 3.3.1 Коефициент на корелација

Коефициентот на корелација ја мери јачината на врската помеѓу две променливи. Коефициентот на корелација има опсег од $-1$ до $1$, односно колку е поголема апсолутната вредност на коефициентот, толку е појака врската помеѓу променливите. Најјаките врски имаат коефициент $-1$ или пак $1$, додека пак слабите врски имаат вредност околу $0$. Доколку вредноста на корелацијата има позитивни вредности тоа значи дека доколку вредноста на едната променлива се зголеми, тогаш и вредноста на другата променлива има тенденција кон зголемување. Од друга страна пак, негативната корелација значи дека доколку една од вредностите на променливите се зголеми, другата вредност се намалува.

Коефициентот на корелација $C_i$ за $i$-тото множество од податоци се пресметува на следниот начин[21]:

$$C_i = \frac{Cov(T,P)}{\sigma_t \sigma_p}$$

каде што $Cov(T,P)$ претставува коваријанса од целната колона $T$ и излезите од моделот $P$, додека пак $\sigma_t$ и $\sigma_p$ се стандардни девијации кои се пресметуваат на следниот начин:

$$Cov(T,P) = \frac{1}{n}\sum_{j=1}^{n}(T_j - \overline{T})(P_{ij} - \overline{P})$$

$$\sigma_t = \sqrt{\frac{\sum_{j=1}^{n}(T_j - \overline{T})^2}{n}} \qquad \sigma_p = \sqrt{\frac{\sum_{j=1}^{n}(P_{(ij)} - \overline{P})^2}{n}}$$

[21] https://www.gepsoft.com/gepsoft/APS3KB/Chapter09/Section1/SS02.htm

при што $P_j$ претставува $j$-тата предвидената вредност од алгоритмот за машинско учење, $T_j$ е $j$-тата целна вредност, додека пак $\overline{T}$ и $\overline{P}$ се средните вредности од целните и предвидените вредности за одредени тест случаи, односно:

$$\overline{T} = \frac{1}{n}\sum_{j=1}^{n} T_j \text{ и } \overline{P} = \frac{1}{n}\sum_{j=1}^{n} P_{(ij)}$$

### 3.3.2 Средна апсолутна грешка

Средната апсолутна грешка ја мери апсолутната разлика помеѓу надгледуваните и предвидените вредности за одредена карактеристика (Willmott, 2005). Оваа метрика се пресметува преку равенството:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|x_{pred} - x_{obs}\right|$$

каде што $x_{pred}$ се предвидените вредности, додека пак $x_{obs}$ се надгледуваните вредности.

Во овој случај се зема во предвид апсолутната вредност на разликата од предвидените и вистинските вредности се дели со бројот на случаи во множеството на податоци, односно ја мери средната апсолутна вредност од грешки во дадено множество со предвидени вредности. Вредностите од метриката се движат од $0$ до $\infty$. Исто така вредноста на оваа метрика може да биде и негативна што значи дека колку е помала вредноста, толку е поточен алгоритмот.

### 3.3.3 Квадратен корен од средна грешка

Квадратниот корен од средната грешка или средна квадратна девијација, претставува стандардна девијација од разликите помеѓу предвидените и набљудуваните вредности. Ефектот на секоја грешка на метриката е правопропорционален со големината на квадратната грешка, односно доколку се појават поголеми грешки, истите имаат непропорционално влијание врз метриката. Оваа метрика се користи за да ја измери точноста на грешките за нумерички предвидувања, за споредување на грешките од

предвидувањата на различни модели за одредени податоци, а се дефинира со следното равенство:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{pred} - x_{obs})^2}$$

## 3.4 Опис на алгоритмите за машинско учење

### 3.4.1 Дрва на одлуки

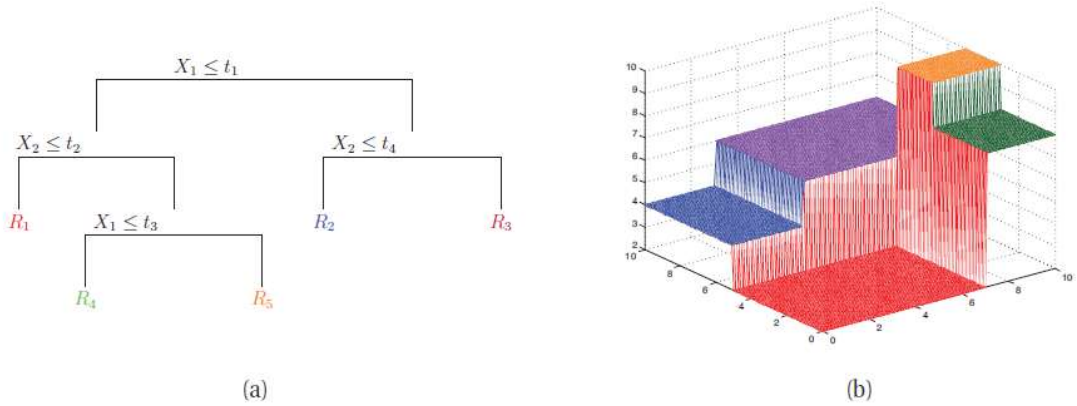
Дрвата на одлуки, или познати како класификациони и регресиони дрва (Classification And Regression Trees - CART) се дефинирани како рекурзивна поделба на влезниот простор и дефинирање на локален модел во секој регион од влезниот простор. Поделбата е во форма на дрво, со еден лист во регион.

#### 3.4.1.1 Основи на алгоритмот

Како еден од најпопуларните алгоритми за машинско учење, дрвата на одлуки најчесто го имитираат размислувањето на човекот и се претставени во структура на дијаграм. Во алгоритмот секој јазол е претставен во форма на тест за одреден атрибут, секоја гранка е претставена како резултат од тестот на атрибутот, додека пак секој лист ги претставува класите, односно лабелите (Breiman, 1984).

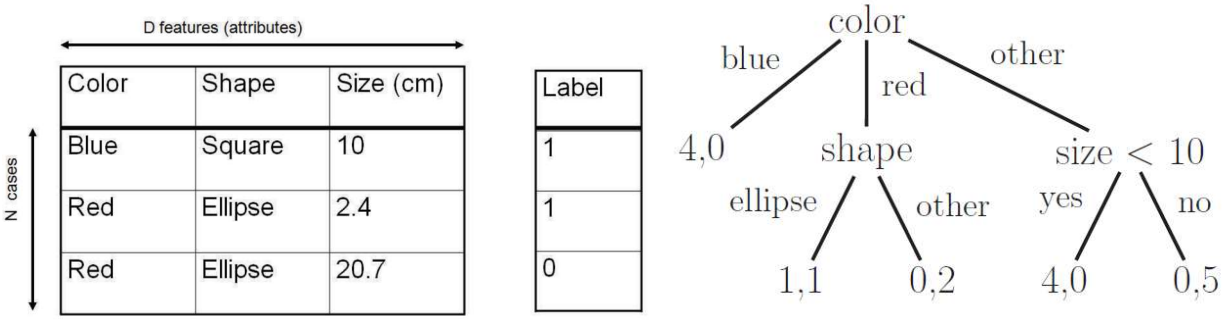
Вредностите на атрибутот за дадена точка $X$ се тестираат низ дрвото на одлуки почнувајќи од коренот на дрвото до листот кој го содржи предвидувањето. На Слика 20 е прикажан CART моделот. Условот во првиот јазол е дали вредноста на $x_1$ е помала од одредена гранична вредност (праг) $t_1$. Доколку е помала вредноста на $x_1$, следува вториот услов каде се испитува дали одредена вредност на $x_2$ е помала од некоја друга гранична вредност $t_2$. Доколку и овој услов е исполнет, тогаш се добива одреден просторен квадрант $R_1$. Ако условот не е исполнет, се проверува дали $x_1$ има помала вредност од $t_3$, итн. Резултатот од

овие паралелни поделби на оските е поделба на 2Д просторот во пет региони, прикажано на Слика 20 (б).




**Слика 20:** Пример за регресионо дрво со два влеза (Hastie et al. 2009)

Методот може да се генерализира во класификација преку зачувување на дистрибуцијата во класни маркери, односно лабели за секој лист, прикажано на Слика 21. Листот означен како $(n_1, n_0)$ покажува дека постојат $n_1$ позитивни примероци и $n_0$ негативни примероци што одговараат на патеката на дрвото. Тука, сите лисја се чисти, што значи дека истите имаат примероци од една класа или од друга, единствен исклучок е листот којшто има црвени кругови и има дистрибуција „1,1“. На пример, прво се проверува бојата на објектот. Доколку е сина, се преминува на левата гранка и завршува со лабела „4,0“, што значи дека има 4 позитивни примероци и 0 негативни примероци кои одговараат на критериумот. Па така, се предвидува дека $p(y = 1|x) = 4/4$ доколку $x$ има сина боја. Доколку има црвена боја, се проверува и обликот на објектот: ако е круг, се објектот се маркира со „1,1“, па предвидувањето би било $p(y = 1|x) = 1/2$. Доколку има црвена боја и не е круг, предвидувањето е $p(y = 1|x) = 0/2$. Ако објектот има друга боја, може да се провери големината, доколку е помала од 10, $p(y = 1|x) = 4/4$, во друг случај $p(y = 1|x) = 0/5$. Овие веројатности се само емпириски приказ на позитивни примероци што го задоволуваат секој услов за сите вредности, дефинирано од коренот до секој лист.




**Слика 21:** Едноставен CART модел (Zikopoulos P. a., 2011)

```
#Import Library
#Import other necessary libraries like pandas, numpy...
from sklearn import tree
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create tree object
model = tree.DecisionTreeClassifier(criterion='gini') # for classification,
here you can change the algorithm as gini or entropy (information gain) by
default it is gini
# model = tree.DecisionTreeRegressor() for regression
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

**Код 1: Python код за Дрва на одлуки (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

#### 3.4.1.2 Растење на дрво

Процесот на растење на дрвото започнува со изборот на поделба низ сите поделби за секој јазол, резултирајќи со најчистите деца јазли. Во процесот на растење на дрвото во предвид се земаат само униваријантните поделби, односно секоја поделба што зависи од вредноста на само една предвидена променлива.

На пример, доколку $X$ претставува номинална категорична променлива од $I$ категории, постојат $2^{I-1}-1$ можни поделби за предвидувачот. Доколку пак $X$ е ординална или непрекината променлива со $K$ различни вредности, постојат $K-1$ различни поделби за $X$. Процесот на растење на дрвото започнува од коренот со повторување на следните чекори:

1. Наоѓање на најдобрата поделба за секој предвидувач. Вредностите за секој од непрекинатиот и ординален предвидувач се подредуваат од најмал до најголем. Потоа, се поминува низ секоја вредност од подредените предвидувачи почнувајќи од врвот, со цел испитување на секоја точка на поделба за кандидатот и одредување на најдобрата поделба. Од тука, најдобрата точка на поделба претставува најдобриот критериум за поделба.
2. Наоѓање на јазолот со најдобра поделба. Преку овој чекор се одбира јазолот кој го максимизира критериумот на поделба.
3. Поделба на јазолот со користење на најдобрата поделба.

За даден јазол $t$, одбрана е најдобрата поделба $s$ за максимизирање на критериумот на поделба $\Delta i(s,t)$.

Доколку $Y$ претставува категорично зависна променлива, за даден јазол $t$, нека веројатностите $p(j,t)$, $p(t)$ и $p(j|t)$ се претставени како:

$$p(j,t)=\frac{\Pi(j)N_{w,j}(t)}{N_{w,j}},\quad p(t)=\sum_j p(j,t)\,,\qquad p(j|t)=\frac{p(j,t)}{p(t)}=\frac{p(j,t)}{\sum_j p(j,t)}$$

каде што $\Pi(j)$ претставува претходна веројатност на зависната променлива $Y$, $Y=j$, $j=1,..J$

$$N_{w,j}=\sum_{n\in \mathfrak{h}} w_n f_n I(y_n=j)$$

односно

$$N_{w,j}(t)=\sum_{n\in \mathfrak{h}} w_n f_n I(y_n=j)$$

$I(a = b)$ претставува индикатор за функцијата и има вредност $1$ доколку $a = b$ и вредност $0$ доколку $a \neq b$, додека пак $\hbar = \{x_n, y_n\}_{n=1}^{N}$ претставува целокупниот примерок за учење.

#### 3.4.1.3 Поткастрување на дрво

За заштита од преоптоварување, дрвото може да се прекине да се развива доколку намалувањето на грешката не е доволна за креирање на дополнителна комплексност при додавање на ново поддрво. Поделбата не е отсекогаш потребна, бидејќи секој параметар засебно можеби има одредена моќ на предвидување.

Познати се два типа на поткастрување:

1. Пред-поткастрување – пред-поткаструвањето означува запирање на растењето на дрвото во процесот индукција и одлука за задржување на јазолот.

2. После-поткастрување – процесот на после-поткастрување се извршува по градењето на дрвото, користејќи пристап од-долу-на-горе, врз база на вредноста од класификационата грешка. Доколку грешката е минимизирана, јазолот е поткрастен преку отстранување на неговите гранки.

Најкористен пристап е после-поткастрување, односно најпрвин е потребно градење на дрвото, а потоа поткастрување преку користење на одредена шема за поткастрување на гранки за добивање најмал случај на грешка.

За поткастрување на дрвото и добивање на секвенца од попрости дрва се користи cost-complexity методот на поткастрување (Kantardzic, 2011). Методот се основа на т.н. параметар на комплексност – $\alpha$ кој постепено се зголемува за време на процесот на поткастрување. На почетокот од последното ниво се поткаструваат јазлите – деца доколку добиената промена во предвидената комплексност на трошок е $\alpha$ пати помала од промената во комплексноста на дрвото. Со други зборови, $\alpha$ претставува мерка за дополнителна точност за гарантирање на помала комплексност на дрвото.

За дадено дрво $T$ и реален број $\alpha$, ризикот за комплексноста на трошок на $T$ се пресметува како:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$

каде што $|\tilde{T}|$ е бројот на листови од дрвото, а $R(T)$ е проценката на ризик од ресупституција на $T$.

#### 3.4.1.4 Gini индекс

Gini индексот се користи за мерење на нечистотијата во множеството од податоците за учење. За даден јазол $t$ индексот е дефиниран како:

$$i(t) = \sum_{i,j} C(i|j)p(i|t)p(j|t)$$

Критериумот за поделба за намалување на нечистотијата е дефиниран преку:

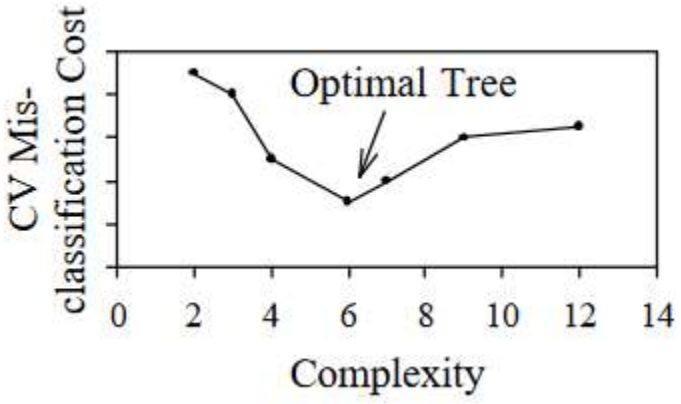
$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

каде што $p_L$ и $p_R$ се веројатностите за премин во состојба) на левиот дете-јазол $t_L$ и за десниот дете-јазол $t_R$. Истите се пресметуваат преку $p_L = p(t_L)/p(t)$ и $p_R = p(t_R)/p(t)$.

#### 3.4.1.5 Вкрстена валидација

Вкрстената валидација претставува метод за валидација на градењето на моделот, избегнувајќи ја потребата од нова и независно валидационо множество од податоци (Lewis, 2000). Во методот се врши поделба на $N$ секции на примерокот за учење, што се овозможува слична дистрибуција на излези на секоја секција од податоците. $N-1$ секции се комбинираат како тренирачко множество во процесот на градење на моделот, додека пак една од овие секции се зачувува како тестирачко множество. На овој начин се градат $N$ различни модели, при што секој од овие модели може да се тестира со тестирачкото множество. Методот на вкрстена валидација се основа на просечно извршување на сите $N$ различни модели со што се добива глобална слика за извршувањето на целосниот модел.

Во алгоритмот, вкрстената валидација се извршува $N$ пати низ процесот на градење и поткастрување на дрвото, при што се добиваат $N$ различни секвенци од дрвото. Врз база на бројот на листови, дрвата во секвенците се вклопуваат за креирање на проценка од извршувањето на дрвото во предвидените излези за тестирачкото множество во форма на функција од бројот на листови или комплексност. Со ова се овозможува минимален трошок во случај кога дрвото е доволно комплексно за вклопување на информациите во тренирачкото множество.




**Слика 22:** Генерирање на оптимално дрво со најмала комплексност на трошок

#### 3.4.1.6 Предности и недостатоци

Алгоритмот има повеќе предности: лесно може да се интерпретира, лесно се справува со дискретни или континуирани влезни податоци, не е чувствителни на монотони трансформации на влезни податоци, може да изврши автоматска селекција на променливи, релативно е робустен, може да се скалира за големи множества од податоци и може да се измени за лесно справување со исчезнати влезни податоци.

Постојат и неколку недостатоци кај дрвата на одлуки. Основниот недостаток е дека не може многу точно да предвидува во споредба со другите типови на модели, што делумно се должи на алчниот карактер на конструкцијата на дрво во алгоритмот. Друг недостаток е нестабилноста: мали промени на влезните податоци може да имаат големи ефекти на

структурата на дрвото поради хиерархиската природа на процесот за растење на дрва, што предизвикува грешка на врвот на дрвото, а има ефект врз остатокот на дрвото.

#### 3.4.1.7 Случајни шуми

Случајната шума претставува група на непоткастрени класификациони или регресиони дрва креирани од случајна селекција на примероци од тренирачкото множество. Случајните параметри се селектирани во фазата на индукција, додека пак претпоставките се креирани по пат на агрегација на самите претпоставки (Murphy, 2012).

Еден од начините да се намали отстапувањето на проценката е заедно да се проценат повеќе проценки. На пример, може да се тренираат $M$ различни дрва на различни подмножества од податоци, избрани случајно со помош на замена, а потоа да се пресмета:

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x)$$

каде што $f_m$ претставува $m$-тото дрво. Оваа техника се нарекува bagging (bootstrap aggregating).

Но за жал, повторното извршување на некој од алгоритмите за учење на различни подмножества на податоци може да доведи до високо поврзани показатели, што ја ограничува количината на намалување на остапувањето. Техниката позната како случајни шуми се основа на учење на дрва врз база на случајно избрано подмножество на влезни променливи и случајно избрано подмножетво на случаи на податоци. Ваквите модели имаат многу добра точност на предвидување и многу често се користат во многу апликации.

Кај случајните дрва постои и начин да се изврши Баесова интерференција на просторот кај дрвата т.н. Баесови дрва со адаптивна регресија (Bayesian Adaptive Regression Trees - BART), но оваа техника е доста бавна кај тренирачкото множество, а од друга страна дава доста добри класификатори. За жал, методите што користат повеќе дрва ги губат нивните основни својства (Cunningham, 2008).

Секое од дрвата во случајните шуми расти на следниот начин:

- Со случајно $N$ семплирање, доколку бројот на случаи во тренирачкото множество е $N$ со замена од оригиналните податоци. Примерокот понатаму се користи за растење на дрвото.

- За $M$ број на влезни променливи, променливата $m$ е одбрана така што за секој јазол е дефинирано $m < M$, $m$ променливите се случајно избрани од $M$ и најдобрата поделба од $m$ се користи за разделување на јазолот. За време на растење на шумата, $m$ има константна вредност.

- Секое дрво е пораснато со најголем степен на растење.

Случајни шуми како алгоритам има значајно подобрување во перформанси во споредба со класификаторите на едно дрво.

Апликации во коишто може да се користи алгоритмот:

- Апликациите на банките за предвидување дали барателот на кредит има висок ризик за добивање на истиот;

- Автомобилската индустрија за предвидување на успехот или неуспехот за траењето на одреден механички дел;

- Здравствена индустрија дали пациентот може да развие хронична болест или не;

- Регресивни задачи за предвидување на просечниот број на споделување на вест на социјалните медиуми и резултатот од перформансите;

- Во апликациите за препознавање на говор и класификација на слики и текстови.

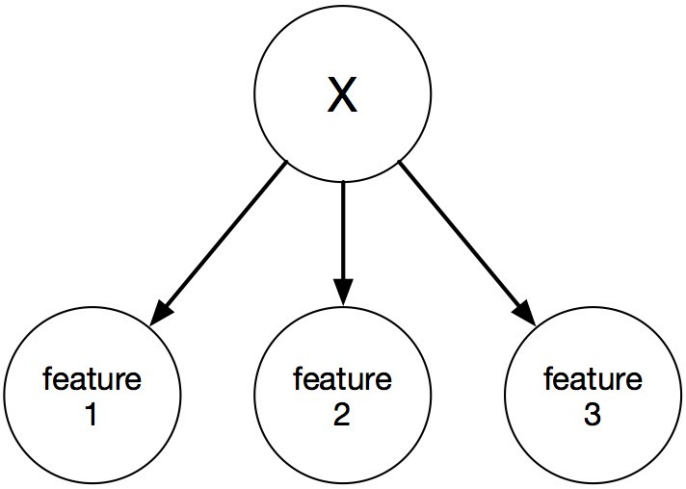
```
#Import Library
from sklearn.ensemble import RandomForestClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create Random Forest object
model= RandomForestClassifier()
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

**Код 2: Python код за Случајни шуми (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

### 3.4.2 Метод на наивен Баес

Методот на наивен Баес претставува една од класификационите техники што се базира на Баесова теорема со претпоставка дека постои независност помеѓу индикаторите (Karau, 2015). Со други зборови, во алгоритмот се претпоставува дека одреден параметар не е поврзан со било кој параметар во множеството со податоци. На пример, овошјето може да биде јаболко доколку е црвено, тркалезно и околу 10cm во дијаметар. Иако овие параметри зависат еден од друг, класификаторот од методот на наивен Баес би ги разгледал сите својства како индивидуални со цел да претпостави дека овошјето е јаболко.

Моделот што е развиен со помош на методот на наивен Баес (Слика 23) лесно се применува за многу големи множества на податоци. Поради едноставноста на методот, во перформанси ги надминува дури и софистираните класификациони методи.




**Слика 23:** Методот на наивен Баес [22]

За дадена класна променлива $y$ и зависно својство во форма на вектор $x_1$ до $x_n$, Баесовата теорема би била:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$$

Со помош на наивна зависност на претпоставка дека:

$$P(x_1|y, x_1, \dots, x_{i-1}, \dots, x_n) = P(x_i|y)$$

За сите $i$, релацијата е поедноставена во следната форма:

$$P(y|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots, x_n)}$$

Сè додека $P(x_1, \dots, x_n)$ се влезните податоци, може да се користи следното класификационо правило:

$$P(y|x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

[22] http://www.cs.cornell.edu/courses/cs4780/2015fa/iframe/iframe-5/index.html

каде што:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

Во овој израз може да се користи MAP предвидување со цел оценување на $P(y)$ и $P(x_i|y)$; првата претпоставка е релативна фреквенција на класата $y$ во тренирачкото множество.

И покрај едноставните претпоставки на овој алгоритам, класификаторите најдобро функционираат во реални ситуации, најмногу во познатите класификација на документи и филтрирање на пораки со спам содржина. Класификаторите најчесто работат со мала количина на тренирачки податоци за проценка на потребните параметри.

Класификаторите на методот на наивен Баес може да бидат доста брзи во споредба со многу од софистицираните методи. Раздвојувањето на својството на класната кондиционална дистрибуција значи дека секоја дистрибуција одделно може да биде проценета како еднодимензионална дистрибуција. На овој начин се ублажуваат проблемите што произлегуваат од димензионалноста.

Баесовата оптимизација припаѓа на алгоритмите за секвенционална оптимизација на модели (Sequential Model-Based Optimization – SMBO) и овозможува користење на резултатите од претходна итерација за подобрување на методот за семплирање во наредното тестирање[23].

Доколку во моделот се дефинирани т.н. хиперпараметри, односно вредности поставени на почетокот од процесот на учење, претставени со $\lambda$, после тренирањето на моделот се добива вредноста $\upsilon$ со користење на одредена евалуциона метрика. Со користење на претходно евалуирани вредности од хиперпараметри се пресметува следното очекување на просторот од хиперпараметарот. Можно е да се изберат оптималните вредности на

[23] https://en.wikipedia.org/wiki/Bayesian_optimization

хиперпараметарот врз база на идното очекување. Процесот итеративно се повторува до доведување на оптимални вредности од оптимизацијата.

Со други зборови, доколку податоците се поделени на тренирачки $S_{train}$ и валидацискиот примерок $S_{valid}$ се решава одреден проблем $P$ што претставува функција од параметрите на моделот ($w$), тренирачкиот примерок $S_{train}$ и одрдерени параметри, како на пример $\alpha$ и $\beta$.

Решавањето на проблемот со оптимизација за фиксно множество од вредности дава одредена вредност $w$. Оптималната вредност на $w$, т.н. $w^*$ како функција од $\alpha$ и $\beta$ се пресметува како:

$$w^*(\alpha, \beta) = arg \min_w P(w, \alpha, \beta, S_{train})$$

Вредноста на $w^*$ се користи за предвидување на валидацискиот примерок за добивање на валидациска грешка, односно функцијата $w^*$ претставува валидациска функција на грешка што како влез ги прима вредностите од хиперпараметрите $\alpha$ и $\beta$ и враќа валидациска грешка. Со други зборови, целта на оптимизацијата на хиперпараметрите е да се најде множество од вредности $\alpha$ и $\beta$ што ја минимизираат валидациската функција на грешка.

Алгоритмот се користи во следните случаи:

- Доколку е потребно уредување на множество од големи податоци;
- Доколку примероците имаат неколку атрибути;
- За даден параметар на класификација, атрибутите што ги опишуваат примероците се условно независни.

Апликации во коишто може да се користи алгоритмот:

- Сентиментална анализа (sentiment analysis) – се користи од страна на Facebook за анализа на промена на статуси преку позитивни или негативни емоции;

- Категоризација на документи – се користи од стана на Google за индексирање на документи и наоѓање на релевантни резултати, како на пример, рангирање на страни. Механизмот на рангирање на страни во базата на податоци ја маркира страната како важна преку анализа и класификација со користење на техниката на класификација на документи;

- Класификација на новости за технологија, забава, спорт, политика;

- Филтрирање на пораки со спам содржина – Gmail го користи овој алгоритам за класификација на пораките дали содржат спам содржина.

```
#Import Library
from sklearn.naive_bayes import GaussianNB
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create SVM classification object model = GaussianNB() # there is other
distribution for multinomial classes like Bernoulli Naive Bayes, Refer link
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

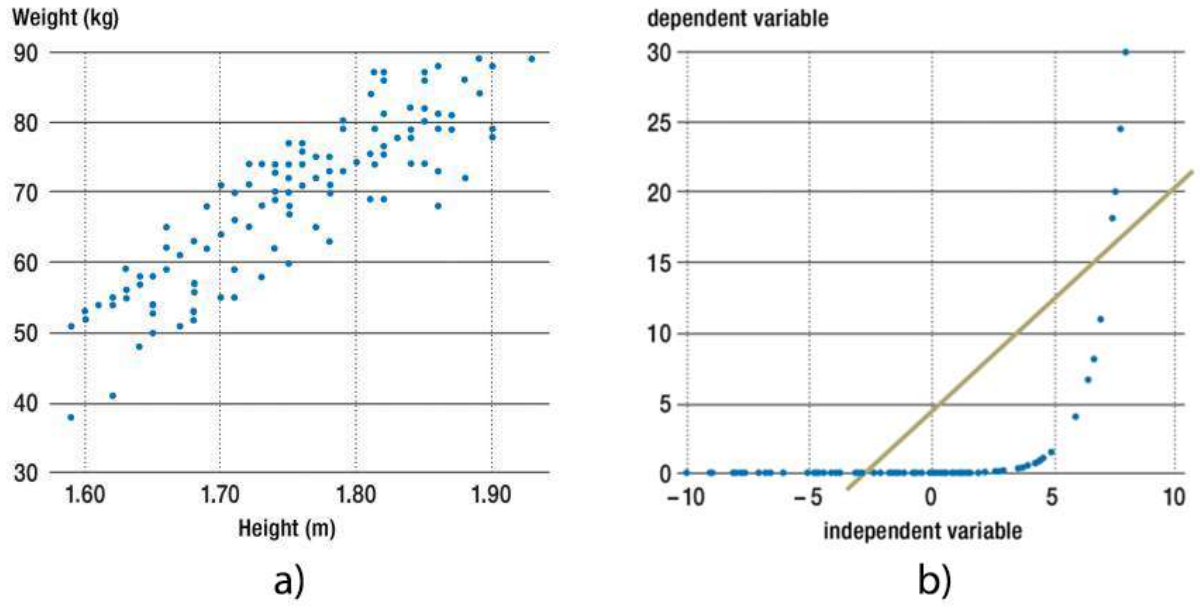
**Код 3: Python код за методот на наивен Баес (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

### 3.4.3 Линеарна регресија

Алгоритмот линеарна регресија претставува една од алатките за статистика и машинско учење и се користи за предвидување на одредена зависна променлива ($Y$) од дадено множество на независни променливи односно карактеристики ($X$).

Зависната променлива може да биде непрекината, додека пак независните променливи може да бидат или прекинати, бинарни или категорични. Релацијата помеѓу две независни

променливи се прикажува графички од каде што може да се направи разлика дали релацијата е линеарна или нелинеарна (Слика 24).




**Слика 24:** а) Линеарна релација б) Експоненцијална релација (Schneider, 2010)

Линеарната регресија се состои од една независна променлива претставена преку релацијата:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

каде што $\beta_0$ е местото каде што се пресретнува $y$, $\beta_1$ претставува наклонот, односно регресиониот коефициент, додека пак $\varepsilon$ е случаен термин на грешка (шум), позитивен или пак негативен.

За множество од $n$ набљудувани вредности на $x$ и $y$, претставени во формата $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, со користење на релација за линеарна регресија, вредностите формираат систем од линеарни равенства, претставени во форма на матрици:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Доколку

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & \vdots \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

релацијата е $Y = XB$.

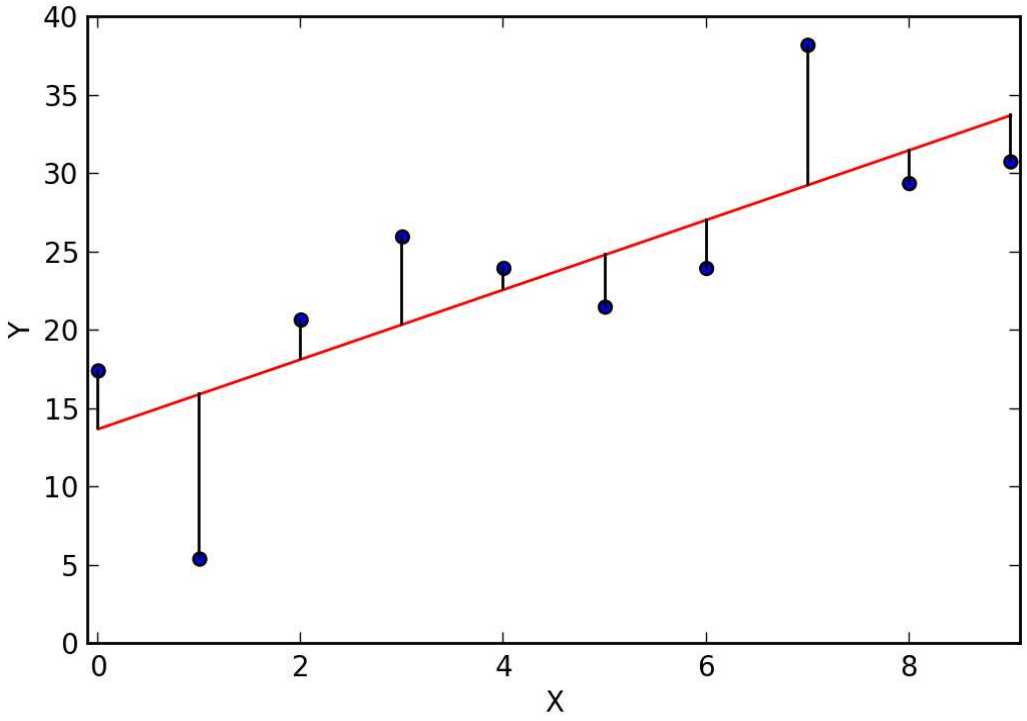
Целта на линеарната регресија на најмалите квадрати (ordinary least square – OLS) е да се тренира линеарниот модел за предвидување на $Y$, истовремено намалувајќи ја грешката. Со намалување на грешката се подобрува предвидувањето, односно самата функција (Zou, 2003).

OLS ги избира параметрите на линеарната функција со решавање на системот линеарни равенки, минимизирајќи ја сумата од квадрати од разликата помеѓу зависните вредности во дадено множество на податоци предвидени со линеарна функција:

$$\hat{y} = \beta_0 + \beta_1(x) + \varepsilon$$

односно со зголемување на вредноста на $X$, останатите вредности се константни, а $Y$ ја зголемува вредноста на $\beta_1$.

Оваа регресија графички може да се престави во правоаголен координатен системкако на Слика 25.




**Слика 25:** OLS функцијата претставена во правоаголен координатен систем (Hackernoon website, 2017)

Cost функцијата на линеарниот модел, односно грешката на линеарниот модел е претставена со следното равенство (Towards Data Science, 2017):

$$Cost = \frac{\sum_1^n((\beta_1 x_1 + \beta_0) - y_i)^2}{2n}$$

Моделите на линеарна регресија најчесто се оптимизираат со користење на пристапот на најмали квадрати, но исто така може да се оптимизираат и со други методи, како на пример со методот на минимизирање на недостатокот од поклопување. Пристапот со најмали квадрати може да се користи за собирање на моделите кои не се линеарни. Па затоа термините „најмали квадрати“ и „линеарен модел“ се тесно поврзани.

```
#Import Library
#Import other necessary libraries like pandas, numpy...
from sklearn import linear_model
#Load Train and Test datasets
```

```
#Identify feature and response variable(s) and values must be numeric and
numpy arrays
x_train=input_variables_values_training_datasets
y_train=target_variables_values_training_datasets
x_test=input_variables_values_test_datasets
# Create linear regression object
linear = linear_model.LinearRegression()
# Train the model using the training sets and check score
linear.fit(x_train, y_train)
linear.score(x_train, y_train)
#Equation coefficient and Intercept
print('Coefficient: \n', linear.coef_)
print('Intercept: \n', linear.intercept_)
#Predict Output
predicted= linear.predict(x_test)
```

**Код 4: Python код за линеарна регресија (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

### 3.4.4 Логистичка регресија

Алгоритмот логистичка регресија, иако го содржи името регресија спаѓа во класификационите алгоритми (Слика 26). Алгоритмот се користи за проценување на дискретни вредности, односно бинарни вредности што се основаат на дадено множество од индивидуални вредности, односно ја пресметува веројатноста на појавување на одреден настан со помош на погодување на податоците со помош на fitting функцијата со помош на користење на логистична функција, односно:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

каде што $e$ претставува Ојлерова константа, $x_0$ е вредноста на $x$ во средната точка на логистичката крива, $L$ претставува максималната вредност на кривата додека пак $k$ е вредноста на стрмнината на кривата.

Сигмоидата[24] како математичка функција е претставена како:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

За реалните вредностите на $x$ во опсег од $-\infty$ до $+\infty$, се добива $C$-кривата, прикажана на Слика 26, што претставува стандардна логистичка функција за $L = 1$, $k = 1$ и $x_0 = 0$.

**Слика 26:** Логистичка регресија [25]

Според оваа функција, вредностите на излезните податоци се помеѓу $0$ и $1$. Логистичката регресија е слична со алгоритмот на линеарната регресија бидејќи најдобро работи со одземање на атрибути кои не се поврзани со излезните променливи, односно со параметри кои зависат еден од друг.

Моделите што се добиваат со помош на користење на алгоритмот се линеарни. Меѓу повеќето предности на алгоритмот е дека излезните податоци имаат добра веројатност, при што алгоритмот ја има можноста за регулирање со цел избегнување на преоптоварување. Моделите развиени со алгоритмот лесно може да се подобрат со додавање на нови податоци, односно нови слични параметри. Како недостаток се смета дека алгоритмот има тенденција за потфрлање кога се користат повеќе и нелинеарни

[24] https://en.wikipedia.org/wiki/Sigmoid_function
[25] https://en.wikipedia.org/wiki/Logistic_regression

граници на одлуки, што не се доволно флексибилни за обработка на покомплексни врски помеѓу податоците.

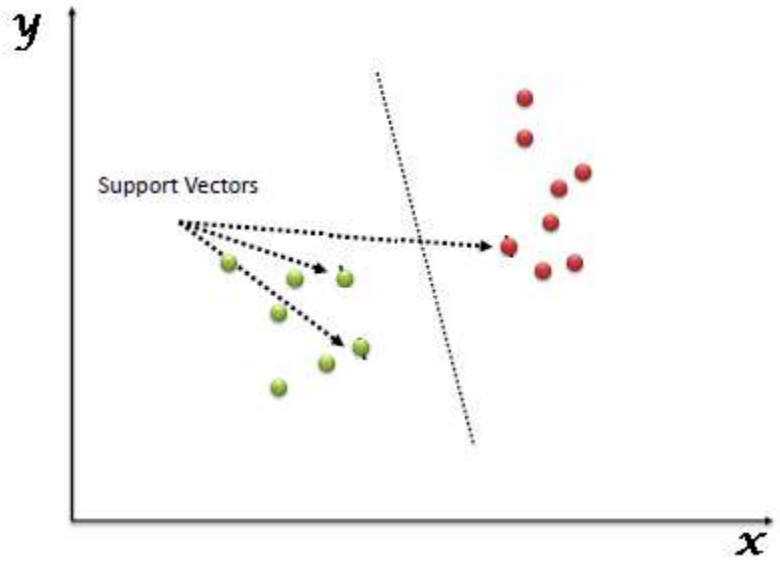
```
#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)
```

**Код 5: Python код за логистичка регресија (Pedregosa, Scikit-learn: Machine Learning in Python, 2017)**

### 3.4.5 Други алгоритми за машинско учење

#### 3.4.5.1 Support Vector Machine (SVM)

SVM е еден од алгоритмите за нагледувано учење што може да се користи и за класификација и за регресија (Durgesh, 2010). Во алгоритмот, сите податоци се претставени во форма на точки во $n$-димензионален простор, каде што $n$ претставува број на параметри во моделот, со вредност за секој параметар претставен во одредени координати. Потоа, се извршува процесот на класификација со помош на наоѓање на хиперрамнината што ги разграничува двете класи, односно двата параметри (Слика 27).




**Слика 27:** Support Vector Machine алгоритам [26]

Векторите за поддршка се точките коишто се најблиски до хиперрамнината, односно точките од множеството на податоци кои доколку се отстранат ќе ја сменат позицијата на хиперрамнината што ги дели истите.

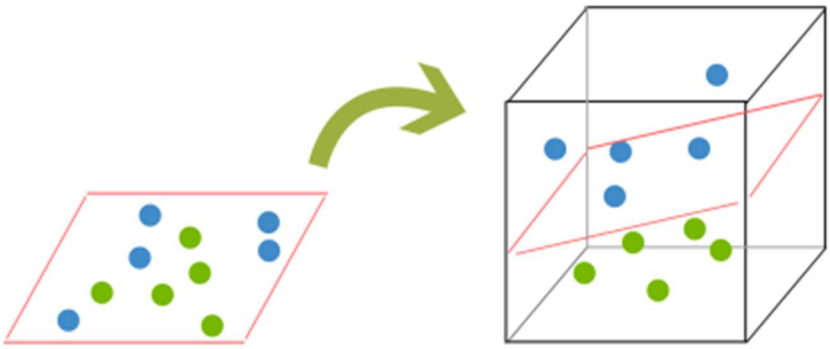
Колку што се подалеку точките од хиперрамнината, толку е посигурно дека тие се правилно класифицирани. Па затоа, потребно е точките да бидат што подалеку една од друга иако би се наоѓале на соодветната страна. Кога ќе се додадат нови податоци за тестирање, било која страна на хиперрамнината би одлучила за класата којашто би се додала на точките.

Растојанието помеѓу хиперрамнината и најблиската точка на секоја страна е наречена маргина. Целта е да се избери хиперрамнина со најдобра маргина помеѓу хиперрамнината и било која точка од тренирачкото множество што би резултирало до подобра шанса за класификација на нови податоци.

За класификација на покомплексно множество на податоци потребно е да се премине од пониска димензија во повисока. Оваа техника во овој алгоритам е позната како kernel -

[26] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

функција, претставено на Слика 28. Со помош на овој трик, неделивиот проблем на податоците станува делив, односно проблемот се расчленува во различни функции коишто се наречени kernel-функции.



**Слика 28:** SVM kernel трик [27]

Предностите кај овој алгоритам се точноста, истиот работи солидно на помали бази на податоци и може да биде поефикасен затоа што користи подмножество на точки на тренирање.

Алгоритмот се користи кај прогноза на берзата кај различни финансиски институции. На пример, може да се користи за споредба на релативните перформанси кај берзите споредено со перформансите на други берзи во истиот сектор. Релативната споредба на берзата помага при менаџирање на влогот врз основа на класификациите на алгоритмот.

Како недостаток на алгоритмот се појавува фактот дека истиот не е добар за поголеми множества на податоци поради времето на обработка на податоците и е помалку ефективен на нечисти множества на податоци со класи кои се преклопуваат.

[27] https://prateekvjoshi.com/2012/08/24/support-vector-machines/

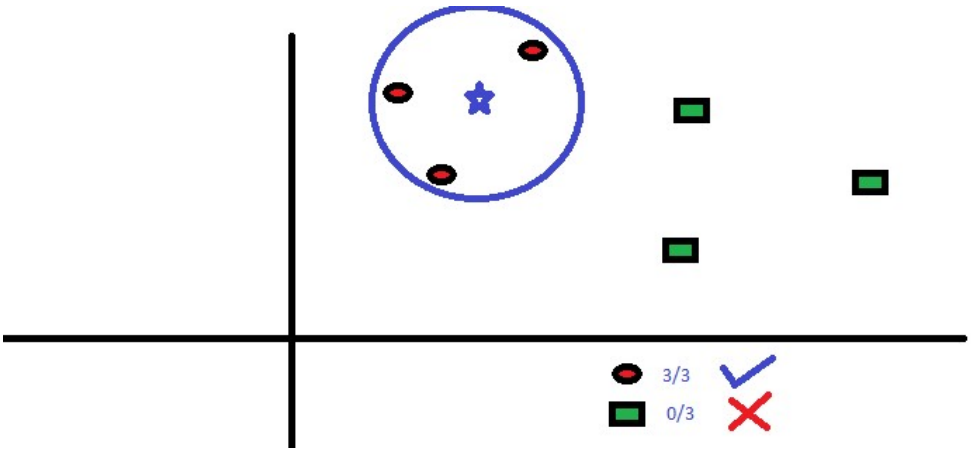
```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create SVM classification object
model = svm.svc() # there is various option associated with it, this is simple
for classification. You can refer link, for more detail.
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

**Код 6: Python код за Support Vector Machine (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

#### 3.4.5.2 KNN (k-nearest neighbors)

KNN алгоритмот е уште еден од алгоритмите што може да се користи за решавање на проблеми со класификација и со регресија, но најчесто се користи за решавање на класификациони проблеми во индустријата. KNN алгоритмот ги зачувува сите можни случаи и класифицира нови случаи од повеќето „гласови“ од неговите $k$ соседи (Слика 29). Понатаму случајот што се доделува на класата претставува најчестиот од најблиските $k$ соседи, измерено според функција за пресметување на далечина кај соседите (Guo, 2003).

Алгоритмот изработува претпоставки користејќи го тренирачкото множество. Претпоставките за нова променлива $x$ се пресметани со пребарување низ целото тренирачко множество за најсличните $k$ променливи (соседи) и ја сумира излезната променлива за дадените $k$ соседи. Излезната променлива за регресија може да биде средната вредност, додека пак во класификација може да биде вредноста на најчестата вредност на класата.




**Слика 29:** KNN алгоритам [28]

За одредување на тоа која од $k$ променливите во тренирачкото множество е најслична на новиот излез, се користи мерка за пресметување на растојанието; за влезни податоци со реални вредности, меѓу најпопуларните мерки за пресметување на растојание е Евклидовата функција за растојание.

Евклидовото растојание се пресметува преку следното равенство:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

каде што $x_i$ и $y_i$ претставуваат точките што треба да им се пресмета нивното растојание.

Во алгоритмот се користат и други мерки за пресметување на растојанието помеѓу точките, како што се:

- Хамингово растојание – се пресметува растојанието помеѓу бинарни вектори;
- Менхетен растојание – се пресметува растојанието помеѓу вектори со реални броеви преку пресметување на сумата на нивните апсолутни вредности;

[28] https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/

- Минкобски растојание – генерализација на Евклидовото и Менхетен растојанието.

Постојат и многу други мерки за пресметување на растојанието, но најдобро е да се избери посакуваната мерка врз база на параметрите во базата на податоци; со помош на експериментирање со различни мерки и различни вредности на променливите се добиваат различни модели кои понатаму подлежат на дополнителни анализи.

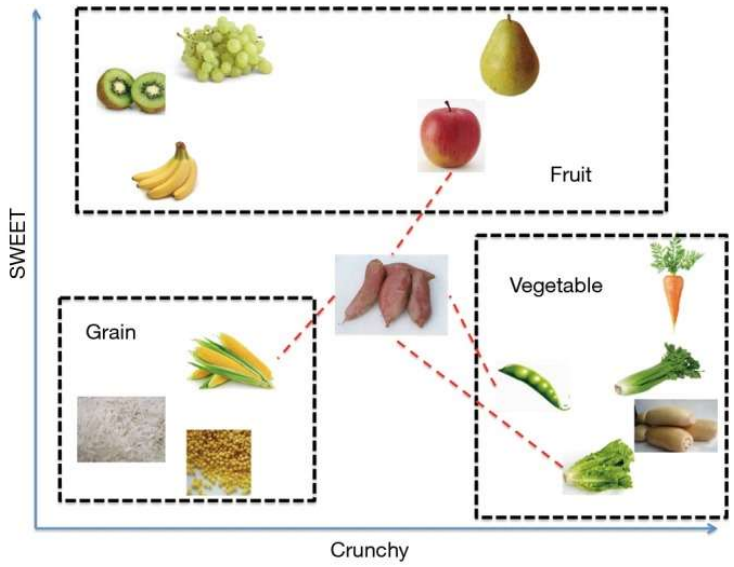
Евклидовото растојание е добра мерка што може да се користи доколку влезните податоци се слични меѓусебе во типот на податоци. Менхетен растојанието најчесто се користи доколку влезните податоци не се со сличен тип.

Вредноста $k$ може да добие со дополнително подесување на алгоритмот. Најчест пристап е да се тестира со различни вредности за $k$ и да се забележи која од вредностите е најдобро решение за проблемот. Комплексноста на алгоритмот се зголемува со зголемувањето на обемот на тренирачкото множество. Кај големите податоци, резултатот може да се добие стохастично преку одреден примерок од тренирачкото множество од каде што се пресметуваат најсличните $k$ вредности.

За решавање на класификациони проблеми, излезните податоци од алгоритмот се пресметуваат преку класата на највисоката фреквенција од најсличните $k$ променливи. Секоја променлива на некој начин ја избира класата, така што класата со најмногу „гласови“ се зема за претпоставка.

Претпоставките на класите се калкулираат со нормализирана фреквенција на примероци кои припаѓаат на секоја класа од најсличните $k$ променливи за новата променлива. На пример, овошјето, зеленчукот и житните производи може да се разликуваат според крцкавоста и присуството на шеќер, претставено на Слика 30. При прикажувањето на истите во дводимензионална рамнина се користат само две нивни карактеристики, но нормално, може да се користи повеќе карактеристики. Генерално овошјето е послатко од зеленчукот, додека пак житните производи го немаат својството на крцкавост, а и не се слатки. Целта во овој пример е да се утврди во која категорија спаѓа слаткиот компир. За овој случај се

избираат четири најблиски видови на храна: јаболко, зелен грав, марула и пченка. Бидејќи зеленчукот има најмногу гласови, слаткиот компир е доделен на класата зеленчук.




**Слика 30:** Пример за претпоставка на класата на одреден продукт [29]

Во примерот се посочени два битни концепти. Во првиот се користи метод за пресметување на растојанието помеѓу благиот компир и останатите типови на храна, додека пак во вториот се вклучува методот на Евклидово растојание за пресметување на растојанието помеѓу променливите.

KNN алгоритмот се користи кај следните типови на бази на податоци:

- Повторно скалирање на податоци – алгоритмот најдобро работи доколку сите податоци имаат се под еднакво скалабилни. Нормализирањето на податоците најчесто е во опсег од 0 до 1. Во овој случај најдобро е да се стандардизираат податоците доколку имаат Гаусова распределба.

[29] http://atm.amegroups.com/article/view/10170/html

- Адресирање на исчезнати податоци – исчезнатите податоци означуваат дека растојанието помеѓу примероците не може да се пресмета. Податоците може да се исклучат бидејќи вредностите што се исчезнати не може да се внесат.

- Пониска димензионалност – алгоритмот најдобро работи кај пониски димензионални бази на податоци.

Сепак, алгоритмот има и свои недостатоци и не може многу често да се корист, бидејќи:

- Не ги распознава позначајните атрибути – кога станува збор за пресметување на растојанието помеѓу точките, секој атрибут е со иста значајност од вкупното растојание. Ова значи дека помалку значајните атрибути имаат исто влијание на растојанието во споредба со позначајните атрибути.

- Не може да се справи со податоци кои недостасуваат – со алгоритмот не може да се изврши класификација доколку недостасуваат дел од податоците. Причината тука е дека растојанието останува недефинирано доколку недостасува еден или повеќе атрибути.

- Има бавно време на предвидување – кај големите податоци, излишно е да се користи алгоритмот доколку постојат голем број на податоци.

```
#Import Library
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and
x_test(predictor) of test_dataset
# Create KNeighbors classifier object model
KNeighborsClassifier(n_neighbors=6) # default value for n_neighbors is 5
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)
```

**Код 7: Python код за KNN (Pedregosa, Scikit-learn: Machine Learning in Python, 2011)**

# 4 ИМПЛЕМЕНТАЦИЈА НА АЛГОРИТМИ ЗА МАШИНСКО УЧЕЊЕ ЗА ПОДОБРУВАЊЕ НА ПЕРФОРМАНСИТЕ ПРИ ОБРАБОТКА НА ГОЛЕМИ ПОДАТОЦИ

## 4.1 АНАЛИЗА НА ПОДАТОЦИ СО APACHE SPARK

Моделот MapReduce е ограничен во обработката на различни типови на податоци што се должи на недоволната скалабилност и робустност (Karau, 2015). Поради овие причини, моделот е надограден во рамката Apache Spark, што ја нуди можноста за обработка на различни типови на податоци вклучувајќи интерактивни прашања заедно со стриминг пресметувања преку брза обработка и анализа на истите. Во споредба со MapReduce, брзината на процесирање на големи податоци е намалена од часови до неколку минути за обработка на податоците. Помеѓу подобрувањата на Apache Spark се вбројува и извршувањето на задачите во меморијата, а исто така, за разлика од MapReduce, рамката е способна за извршување на комплексни задачи и на диск.

Spark ја нуди можноста за покривање на голем опсег на задачи наместо дистрибуирано извршување на задачите и пак преку одделни системи како што се batch апликации, интерактивни алгоритми, прашања и стриминг. Сите овие задачи што се веќе вклучени во Apache Spark ја олеснуваат работата преку лесно и евтина комбинација на различни процесирачки типови.

Архитектурата на рамката овозможува лесен пристап кон многу вградени библиотеки и доста обични API-а претставени во Python, Java, Scala и SQL, вклучувајќи и интеграција со многу други алатки за обработка на големи податоци. Apache Spark исто така работи со Hadoop кластери и има пристап до било кој извор на податоци на Hadoop.

Во јадрото на Spark постои т.н. пресметувачка машина што е одговорна за распределување и мониторирање на апликации што имаат многу задачи низ многу машини, односно пресметувачки кластер. Јадрото на Spark контролира повеќе компоненти на повисоки нивоа кои се специјализирани за различни задачи како што се SQL или пак машинското

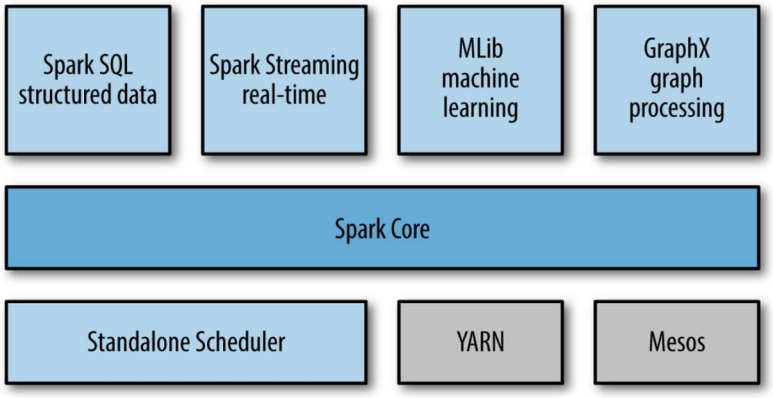
учење. Компонентите се дизајнирани за лесна интерполација со библиотеки за одреден проект.

Сите библиотеки и компоненти имаат голема корист од подобрувањата на пониските слоеви, како на пример, кога се додава или креира некоја оптимизација во јадрото, автоматски се подобрува работата на задачи во SQL и машинското учење. Од друга страна пак, на пример, трошоците со работење на меморијата се минимизирани, бидејќи наместо 5 или 10 индивидуални софтверски системи, компанијата може да работи само со еден систем. Останатите трошоци што се исто така минимизирани вклучуваат развој, одржливост, тестирање и корисничка поддршка. Со ова се покажува дека секогаш кога се додава нова компонента во Spark, секоја компанија што ја користи рамката веднаш ќе ја има можноста да ја додаде таа компонента. Трошоците се намалуваат на тој начин што се заштедува време системот да ја симнува, развива или пак да се надогради со цел да може да ја користи новата компонента.

Една од најголемите придобивки на Spark е дека преку Spark можно е и градење на апликации што се базираат на одредена комбинација на различни процесирачки модели. Како на пример, во Spark може да се креира апликација што користи одреден алгоритам за машинско учење во реално време на тој начин што податоците се добиваат инстантно од различни извори. Истовремено можно е да се добиваат анализи на податоците исто така во реално време.

### 4.1.1 Структура на Spark

Структурата на Spark е претставена на Слика 31:




**Слика 31:** Структура на Apache Spark [30]

Јадрото на рамката ги содржи основните функционалности на Spark, вклучувајќи компоненти за организирање на задачи, управување со меморија, оправување на грешки, интеракција со системи за складирање итн. Јадрото содржи т.н. еластични дистрибуирани множества на податоци (Resilient Distributed Datasets – RDDs). Овие множества претставуваат колекција на податоци што се дистрибуирани низ многу јазли и може да се обработуваат паралелно. Јадрото на Spark содржи многу други API-а за креирање и управување со различни колекции.

Spark SQL претставува пакет за работа со структурирани податоци, што овозможува обработка на прашања преку SQL или преку Apache Hive за SQL и поддржува различни типови на податоци како што се Hive табели, Parquet и JSON. Spark SQL им овозможува на развивачите на апликации да помешаат SQL прашања преку одреден програмски јазик како Python, Java или Scala во истата апликација за добивање на комплексни анализи.

Друга компонента на рамката е Spark streaming што овозможува процесирање на живи податоци. На пример, Spark обработува лог податоци што се генерирани од одреден сервер

[30] http://www.spark-stack.org

или нови статусни пораки што се генерирани од корисниците на одреден веб сервис. Spark стриминг нуди различни API-а за манипулација на живи податоци кои се поклопуваат со RDD на јадрото на рамката што го олеснува учењето на проектот и манипулација на податоци зачувани во меморија или пак во дискот кои стигнуваат во реално време. Овој дел од структурата на Spark е дизајниран за да им понуди на корисниците лесно решавање на грешки што е овозможено од самата скалабилност на јадрото на Spark.

Можеби еден од најбитните делови од структурата на Spark е што содржи MLlib, односно библиотека за машинско учење. MLlib им нуди на корисниците многу алгоритми за машинско учење, вклучувајќи алгоритми за класификација, регресија, кластерирање како и евалуација на самите модели што се креирани во Spark. Во основа содржи примитиви на машинско учење на ниско ниво и методи што се дизајнирани за лесно скалирање низ одредениот кластер.

GraphX е исто така библиотека за управување со графици, како на пример креирање на график за одреден пријател на одредена социјална мрежа, но исто така преку библиотеката се овозможува паралелна обработка на податоци. Оваа библиотека овозможува креирање и манипулација со графици како на пример subgraph или mapVerticles, но може и да се вклучат одредени библиотеки за алгоритми за графици, како на пример PageRank.

Spark е креиран со цел ефикасно скалирање од еден до илјадници јазли. За максимална флексибилност, рамката работи на различни менаџери за кластери, вклучувајќи Hadoop YARN, Apache Mesos и доста обичен менаџер на кластер што е вклучен во Spark што се нарекува Standalone Scheduler.

Освен тоа што Spark преставува рамка за пресметување на кластери, платформата се користи и за различни типови на апликации. Апликациите најчесто се поделени во две категории, наука за податоци и податочни апликации.

Самата структура на Spark им нуди на развивачите на апликации голем број на компоненти за интерактивни анализи на податоци преку Python или Scala. Бидејќи Spark SQL е веќе

вградено во Spark, им овозможува на корисниците да обработуваат податоци со стандардни SQL команди, на сличен начин како што е вградена MLlib библиотеката што им нуди голем избор на алгоритми за анализа на податоци. Spark ја има можноста за обработка и анализа на големи количества на податоци што им овозможува на корисниците решавање на проблемите преку алатки како што се R или Pandas.

Процесот при обработка на големи податоци започнува со фаза на истражување односно продолжена работа на веќе претходни истражувања со цел подобрување на грешките кои се појавиле во истражувањата и креирање на апликација за обработка на подароци за покомплексни анализи, што би се задоволила бизнис логиката на компанијата. На пример, не се исклучува можноста за креирање на дополнителен систем што има одредени претпоставки и се интегрира во одредена веб апликација за креирање на предлози за кои производи би биле најсоодветни за корисниците.

Рамката најчесто се користи за креирање на различни апликации за процесирање на податоци што се јавува потребата од солидни познавања на принципите од софтверското инженерство, како на пример енкапсулација, дизајн на интерфејс и објектно-ориентирано програмирање, како и креирање на софтверски системи што имплементираат најразлични студии на случај.

Како рамка за паралелизирање на апликации низ различни кластери, Spark ја крие комплексноста на програмирањето на дистрибуираните системи и комуникацијата низ мрежата. Овој систем им дава доволно контрола за мониторирање, истражување и подесување на апликациите за имплементирање на различни задачи.

Предностите на Spark вклучуваат широк опсег на функционалности, како што се лесна употреба, учење и доверливост.

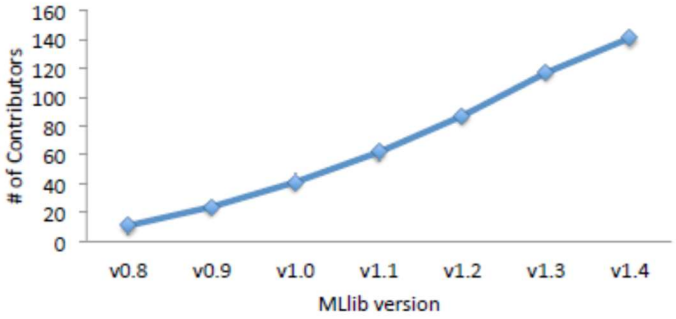
## 4.2 МАШИНСКО УЧЕЊЕ ПРЕКУ APACHE SPARK

Зголемувањето на комплексноста и големината на новите множества на податоци поттикнува развој на различни решенија за лесна и брза обработка и анализа на големи податоци преку користење на различни статистички методи. Во основа, постојат различни решенија преку генерализација на MapReduce за процесирање на големи податоци што вклучува и дополнителни функционалности за машинско учење.

Apache Spark како open-source рамка е толерантна на грешки и е ефикасна за итеративни пресметки за развој на апликации што користат алгоритми за машинско учење.

MLlib претставува најголемата библиотека што е вградена во Spark. Целта на MLlib е да ги таргетира својствата на големи податоци што се должи на паралелизмот на податоците и на моделите, што како параметри ги содржи самиот Spark. MLlib се состои од голем број на брзи и скалабилни алгоритми за учење поделени како алгоритми за класификација, регресија, колаборативно филтрирање, кластерирање и редукција на димензионалност. Исто така библиотеката содржи најразлични примитиви од статистика, линеарна алгебра и оптимизација испишани во Sacla и C++ библиотеки на секој јазол за линеарна алгебра.

Помеѓу повеќето предности на Spark е дека истиот е дизајниран за итеративни пресметки што овозможува развој на ефикасна имплементација на алгоритмите за машинско учење (Meng, 2016). Бидејќи кодот на Spark е отворен, лесно може да се допринесе во развојот на рамката и на библиотеката. И на крај, како дел од компонентите на Spark (Слика 32), MLlib им нуди на развивачите на апликации широк опсег на алатки со цел да го забрзаат развојот на цевководот на машинското учење.




**Слика 32:** Приказ на развивачи и подобрувачи на MLlib (Meng, 2016)

### 4.2.1 Карактеристики на MLlib

Меѓу повеќето карактеристики на библиотеката се вклучуваат и:

- Множество од алатки и методи – помеѓу повеќето алатки се вклучени алгоритмите за машинско учење, како на пример: различни линеарни модели, методот на наивен Баес и дрва на одлуки за решавање на проблеми со класификација и регресија, ALS алгоритмот (Alternating Least Squares - ALS) за колаборативно фитлрирање, K-means за кластерирање и анализа со цел намалување на димензионалноста, итн. Библиотеката содржи голем број на примитиви на ниско ниво и основни алатки за конвексна оптимизација, статистичка анализа, дистрибуирана линеарна алгебра, како и доста влезно-излезни формати.

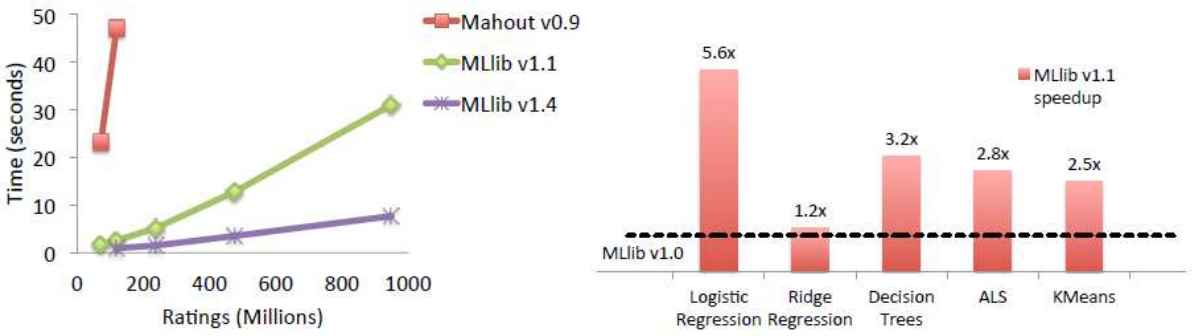
- Оптимизација на алгоритми – библиотеката вклучува најразлични оптимизации за подобрување на ефикасноста на дистрибуираното учење и претпоставките. На пример, ALS алгоритмот ги блокира непотребните променливи со цел намалување на garbage колекција со цел спречување на преоптоварување на различни оператори од линеарна алгебра. Дрвата на одлуки пак, користат најразлични препораки од проектот PLANET за дискретизација на параметарот на зависните податоци со цел намалување на трошоците за комуникација (Panda, 2009).

- Цевководен API – цевководот на машинското учење вклучува секвенца на предпроцесирање на податоци, екстракција на параметри, подготовка на моделот

и одредени степени на валидација. Повеќето од библиотеките за машинско учење не содржат основна поддршка за различни функционалности во цевководот. MLlib содржи најразлична поддршка што адресира до повеќето функционалности во цевководот преку пакетот познат како spark.ml[31].

- Интеграција со Spark – библиотеката има најразлични бенефити од многуте компоненти што се вградени во екосистемот на Spark. Јадрото на Spark ја нуди можноста до пристап од преку 80 оператори за трансформација на податоци.

- Документација, Spark заедница и зависности – од страна на Spark заедницата (Слика 33), развиена е документација којашто им помага на корисниците лесно да пристапат до опис на саканите методи и алатки преку примери со код[32]. Документацијата исто така содржи најразлични зависности на кодот од MLlib, како на пример библиотеките Breze, netlib-java и NumPy. Заедницата пак многу често ги известува корисниците преку најразлични пораки за новостите од библиотеката и има развиено систем за поддршка и усовршување на кодот на библиотеката.




**Слика 33:** а) Временски развој на библиотеката MLlib б) Развој на алгоритмите од MLlib (Meng, 2016)

[31] https://spark.apache.org/docs/1.2.2/ml-guide.html
[32] https://spark.apache.org/docs/latest/ml-guide.html

## 4.3 Анализа и подобрување на алгоритмите за машинско учење

### 4.3.1 Модел 1: Оптимизација на моделот преку споредување на параметрите со примена на алгоритмот регресија на случајни шуми

#### 4.3.1.1 Цел за креирање и оптимизација на моделот

Со цел креирање, анализа и оптимизација на моделот, се вклучува MLlib библиотеката за користење на регресија на случајни шуми алгоритмот. Целта при креирањето на моделот е да се генерираат и оптимизираат резултати во форма на предвидени вредности што понатаму се споредуваат со резултатите кои се пресметани преку емпириски пат од базата на податоци. Преку алгоритмот се прикажани и најзначајните карактеристики коишто влијаат врз целната колона, подредени по значајност. Моделот се евалуира со помош на одредени метрики за евалуација.

#### 4.3.1.2 Опис на податоците

За целите на моделот, како база на податоци се користи „Choose Maryland: Compare States - Quality Of Life“ (Health Data, 2015), со лиценца за користење (ODbL, 2016). Базата е претставена во .csv формат и содржи податоци за секоја Американска држава вклучувајќи множество од параметри (Number of America's Byways, Number of National Parks, Number of National Historic Landmarks, National Register of Historic Places Listings …). Базата содржи 51 редица и 7 колони, каде што емпириски е пресметан квалитетот на живот (Well-being index) на граѓаните за секоја Американска држава.

#### 4.3.1.3 Трансформација на податоците

Пред-процесирање на податоците е клучен чекор за извршување на понатамошна обработка на податоците, односно, потребна е претходна подготовка на податоците. Често пати се јавува потребата од приказ на податоците во суров формат што понатаму преку дополнителна обработка би се добиле податоци кои би биле најсоодветни за креирање на моделот.

За вчитување на базата на податоци, се користи методот *ReadDataFrame*.

Следниот чекор во креирање на моделот е дополнителна обработка и анализа на типот на податоци. Доколку податоците се од текстуален тип, вредностите треба да се кодираат во нумерички тип, со цел да се добие индексот од текстуалната колона. За таа цел се користи методот *StringIndexer*. Методот ги трансформира лабелите од текстуален тип во лабели со индекси, каде што најфреквентните лабели добиваат индекс $0$. Во моделот се индексираат колоните *ArtsPerCapita* и *InternetHouseholds*. Со помош на овој метод се добиваат две нови колони со префиксот *indexed.*

Методот *OneHotEncoder*[33] се користи за креирање на бинарен вектор од индексираните податоци. Методот се извршува на двете индексирани колони, при што новите вредности се запишуваат во две нови колони со префиксот *encoded* (Табела 1).

**Табела 1: Додавање на нови колони: индексирана колона, кодирана колона и колона со вектор на карактеристики**

| State | Byways | Landmarks | Listings | Arts per capita | Internet Households | Employment | Professionally Active Physicians | Well-being index | Index Arts per capita | Index Internet Households | Encoded Index Arts per capita | Encoded Index Internet Households | Features |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alabama | 4 | 7 | 38 | 1280 | $0.97 | 68.3% | 2.33 | 61 | 30 | 39 | (45, [30], [1.0]) | (45, [39], [1.0]) | [4.0, 7.0, 38.0, 1280.0, 8.63, 2.33] |

За понатамошна анализа на податоците, односно тренирање на моделот, потребно е користење на методот *VectorAssembler* со кој се соединуваат повеќе колони во единствена вектор колона, резултирајќи во нова колона т.н. *features* што ги содржи вредностите од сите колони трансформирани во вектор.

[33] https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

По завршување на трансформацијата на моделот, истиот е подготвен за додатна оптимизација со помош на користење на алгоритмот регресија на случајни шуми и користење одредени регресиони метрики за одредување на карактеристики кои најмногу влијаат врз *WellBeingIndex.* Со помош на евалуационите метрики се пресметува грешката при предвидување, односно за почеток се пресметуваат предвидените вредности за *WellBeingIndex*, а понатаму се споредуваат со емпириските вредности на моделот.

#### 4.3.1.4 Оптимизација на моделот со користење на алгоритмот регресија на случајни шуми

Регресијата како процес овозможува предвидување на вредноста на една колона врз база од вредностите на повеќето колони. Во моделот, со користење на регресијата од алгоритмот регресија на случајни шуми се пресметуваат претпоставените вредности на колона *WellBeingIndex* кои се базираат на вредностите од колоната *features*. Со понатамошни анализи се согледува значењето на секоја карактеристика врз вредностите на колоната *WellBeingIndex*. Алгоритмот регресија на случајни шуми има различни параметри кои може да се прилагодат за подобрување на веродостојноста на резултатите.

За предвидување на најточните параметри на алгоритмот се користи методот *GridSearch*, што дозволува специфицирање на множество параметри од предвидувачот и евалуаторот. Методот користи на хиперпараметри (Pedregosa, Scikit-learn: Machine Learning in Python, 2011). Оваа оптимизација го претставува проблемот со избор на множество од оптимални хиперпараметри за алгоритмот за учење. Потребно е и дополнително прилагодување на параметрите за постигнување на најдобро решение на соодветниот проблем со алгоритмот за машинско учење.

Методот *GridSearch*, како еден од најосновните методи за оптимизација и подесување на хиперпараметарот, гради модел за секоја можна комбинација од сите вредности на хиперпараметарот, евалуирајќи го секој модел и ја избира архитектурата која дава најдобри резултати.

Доколку, на пример се креира листа за параметрите `number_of_estimators` и `max_depth`, методот ќе креира модел за секоја можна комбинација, односно доколку:

```
number_of_estimators = [10, 50, 100, 200]
max_depth = [3, 10, 20, 40]
```

се креираат следните модели:

```
RandomForestClassifier(number_of_estimators=10, max_depth=3)
RandomForestClassifier(number_of_estimators=10, max_depth=10)
RandomForestClassifier(number_of_estimators=10, max_depth=20)
RandomForestClassifier(number_of_estimators=10, max_depth=40)

RandomForestClassifier(number_of_estimators=50, max_depth=3)
RandomForestClassifier(number_of_estimators=50, max_depth=10)
RandomForestClassifier(number_of_estimators=50, max_depth=20)
RandomForestClassifier(number_of_estimators=50, max_depth=40)

RandomForestClassifier(number_of_estimators=100, max_depth=3)
RandomForestClassifier(number_of_estimators=100, max_depth=10)
RandomForestClassifier(number_of_estimators=100, max_depth=20)
RandomForestClassifier(number_of_estimators=100, max_depth=40)

RandomForestClassifier(number_of_estimators=200, max_depth=3)
RandomForestClassifier(number_of_estimators=200, max_depth=10)
RandomForestClassifier(number_of_estimators=200, max_depth=20)
RandomForestClassifier(number_of_estimators=200, max_depth=40)
```

За секоја комбинација од вредности се евалуира валидациската функција на грешка и се одбира минималната вредност од функцијата.

Самата природа на методот е во детализирањето, односно „рачното“ пребарување низ одредено подмножество од просторот на хиперпараметри на алгоритмот за машинско учење. Успешноста на методот се мери со помош на вкрстена валидација на тренирачкото множество или евалуација на валидираното множество. Во моделот, стратегијата за поделба на подможества е автоматски поставена што значи дека бројот на поделби за секој јазол од дрвото е автоматски.

Како влезни параметри во методот се внесуваат методите *RandomForestRegression* и *RegressionEvaluator*. *GridSearch* содржи најразлични параметри кои можат рачно да се променуваат, како што се: *Maximum Depth, Maximum Bins, Minimum instances per node, Maximum Memory, Cache node IDs, Checkpoint interval, Regression impurity, Sampling Rate, Seed, Number of trees* и *Feature Subset strategy*. Во соодветниот модел, како најоптимални

параметри за модификација се избрани: *Maximum Depth*, *Maximum Bins* и *Number of trees*. За секое од овие три множества може да се внесат 3 податоци, одделени со запирка, при што се добиваат 27 модели ($3\ x\ 3\ x\ 3$) кои треба да бидат тренирани, а потоа евалуирани.

*GridSearch* се користи за прикажување на најоптималните параметри за моделот, за избраната метрика за евалуација (Табела 2).

**Табела 2: Вредности на параметрите на GridSearch**

| | | | |
|---|---|---|---|
| **max bins** | 32 | 40 | 60 |
| **max depth** | 10 | 20 | 30 |
| **number of trees** | 10 | 50 | 100 |

По тестирањето на сите 27 модели со сите параметри кои се зададени во *GridSearch*, се избираат следните вредности од параметрите за најсоодветни за моделот (Табела 3).

**Табела 3: Оптимални вредности на параметрите на GridSearch**

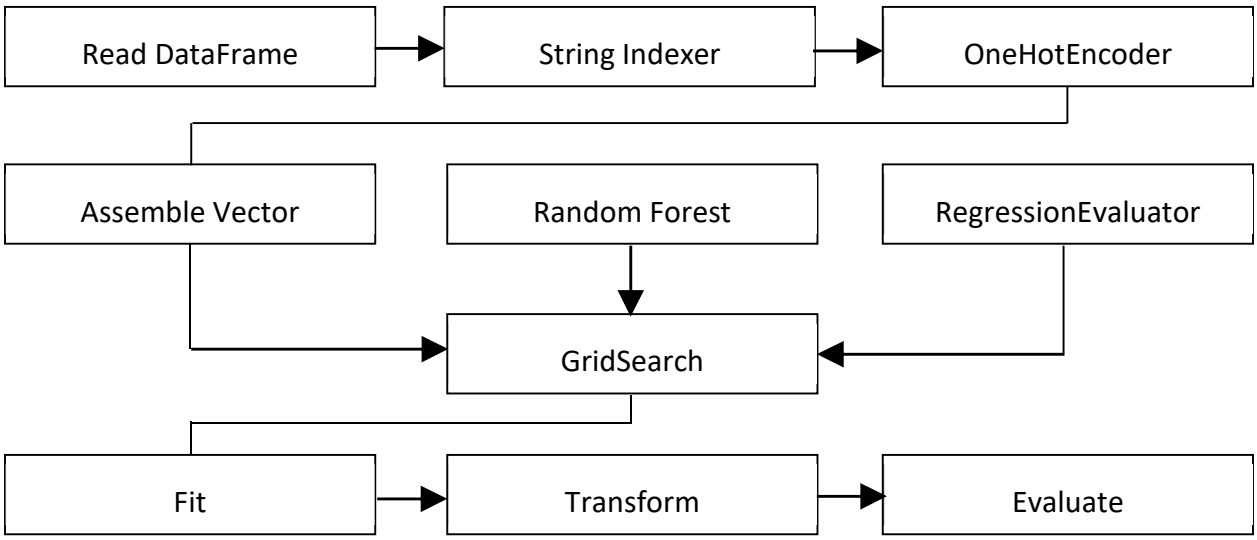
| | |
|---|---|
| **max bins** | 32 |
| **max depth** | 10 |
| **number of trees** | 50 |

За подетални анализи бројот на дрва може да се зголеми, бидејќи самата природа на регресија на случајни шуми алгоритмот покажува дека колку повеќе дрва се користат, толку поточни се резултатите од регресијата.

#### 4.3.1.5 Тренирање на моделот

За одредување на точноста на параметрите на моделот се користи методот *Fit* за тренирање. Најсоодветните параметри кои се добиени од *GridSearch* (max bins, max depth и number of trees) се внесуваат во методот со цел да се истражи која колона, односно кој параметар најмногу влијае врз колоната *WellBeingIndex*. Процесот за развој на моделот е прикажан на

Слика 34. По тренирањето на моделот, карактеристиките што го дефинираат Well-being индексот се подредени и евалуирани по значајност.




**Слика 34:** Процес на развој на моделот 1

### 4.3.2 Модел 2: Оптимизација и верификување на ефективноста на моделот преку користење на алгоритмите регресија на случајни шуми и логистичка регресија

#### 4.3.2.1 Цел за креирање и оптимизација на моделот

Целта за оптимизација на креираниот модел е примена на два различни алгоритми за машинско учење преку креирање на различни под-модели за оптимизација и верификација на резултатите од моделот. Користејќи иста база, трансформација на податоците и различен алгоритам се утврдува точноста и влијанието на резултатите од моделот. Со помош на алгоритмот регресија на случајни шуми се согледуваат најзначајните колони кои делуваат врз резултатот на целната колона, додека пак со помош на алгоритмот логистичка регресија се согледува веродостојноста на конечните вредности на целната колона.

#### 4.3.2.2 Опис на податоците

Во моделот се користи слободната база на податоци „Supplier Directory Data“, претставена во .csv format каде што е претставена листа на снабдувачи и испораки превезени до одредена локација во Американска држава (Health Data, 2015). Базата на податоци е 174MB

и содржи $9$ колони (Country, DBA Name, Address, City, State, Zip, Phone, Product Category Name, Competitive bid) и $716.681$ ред.

#### 4.3.2.3 Трансформација на податоците

Податоците подлежат на дополнителна обработка со цел да се подобри резултатот од оптимизацијата на моделот. Истите се вчитуваат со *ReadDataFrame* методот. Процесот на пред-процесирање вклучува трансформација на податоците од текстуален тип во индексирани податоци, користејќи го *StringIndexer* методот. Во моделот, сите колони од текстуален тип се индексираат, односно колоните во опсег $0$ до $8$, креирајќи нови колони со префиксот *indexed.*

За понатамошна обработка на податоците и на индексираните колони се користи методот *OneHotEncounter* за креирање на вектор од индексираните податоци. Методот врши трансформација на редните вредности во форма на бинарен вектор. Со овој метод се вклучуваат сите индексирани колони, освен целната колона *Competitve_Bid*.

Бидејќи целната колона има Булови вредности (true или false), потребно е истите да се трансформираат во нова колона (*Competitve_Bid_label*) со нумерички вредности, каде што точните вредности добиваат вредност $1$, а грешните вредности $0$.

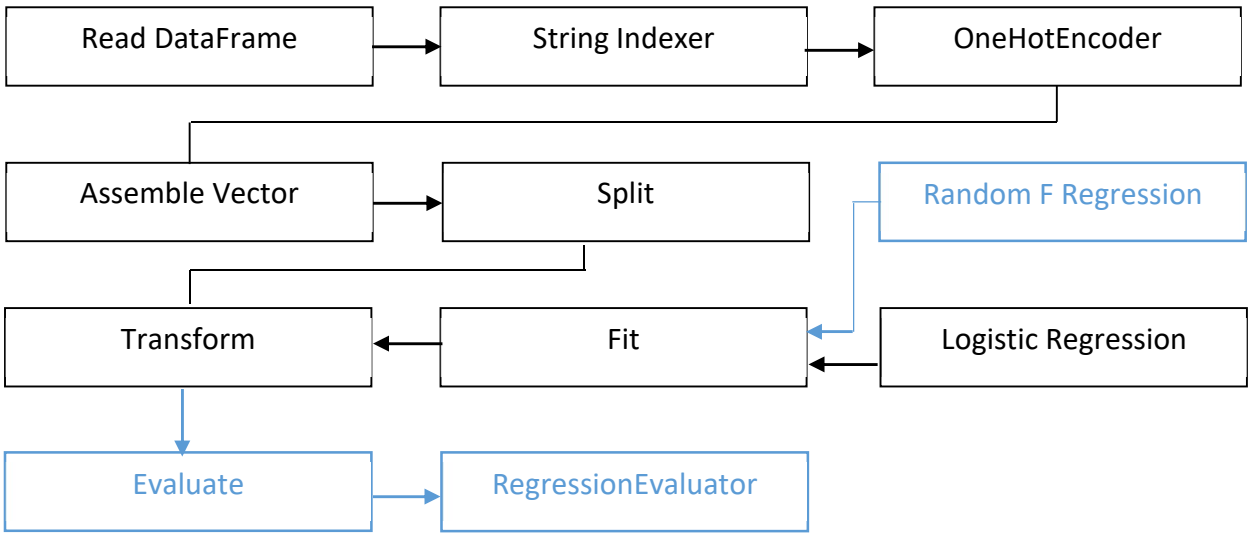
На крај од трансформацијата, сите трансформирани колони се соединуваат во една колона, наречена *features* за понатамошна оптимизација и тренирање на моделот со користење на *VectorAssembler* методот, прикажано на Табела 4.

**Табела 4: Трансформација на вредностите од моделот**

| Competitive_Bid | Competitive_Bid _label | features |
|---|---|---|
| false | 0 | (23306, [(227, 1), (3450, 1), (6718, 1), (12577, 1), (14195, 1), (15373, 1), (18157, 1), (23304, 1)]) |
| true | 1 | (23306, [(175, 1), (3228, 1), (6724, 1), (12368, 1), (14196, 1), (14794, 1), (18195, 1), (23300, 1), (23305, 1)]) |

#### 4.3.2.4 Тренирање и оптимизација на моделот со користење на регресија на алгоритмот случајни шуми и логистичка регресија алгоритмот

Процесот на оптимизација на моделот продолжува со поделба на податоците од моделот на тренирачко и тестирачко множество. Поделбата се извршува со помош на *Split* методот со однос од $0.75$, односно $75\%$ од податоците се во тренирачкото множество, додека пак $25\%$ припаѓаат во тестирачкото множество. Доколку се избери помал процент на податоци во тренирачкото множество, параметрите за проценка би имале поголемо отстапување. Од друга страна пак, со помалку податоци во тестирачкото множество резултатите од моделот би имале поголемо отстапување. Податоците правилно е да се поделат на начин кој отстапувањето на резултатите нема да биде премногу високо. Дефинираниот сооднос е избран за да припаднат повеќе од податоците во тренирачкото множество, односно поголем дел од податоците да бидат тренирани, а да изврши тестирање врз помало множество од податоци. Во под-моделот, каде што се користи алгоритмот регресија на случајни шуми, се користи соодветниот метод *RandomForestRegression* за пресметување на најзначајните карактеристики што ја дефинираат целната колона, подредени по приоритет. Во под-моделот за дефинирање на веродостојноста на резултатите се користи *LogisticRegression* методот (Слика 35).




**Слика 35:** Процес на развој на моделот 2

Имајќи ги во предвид различните параметри на *RandomForestRegression* методот, во под-моделот се избрани вредностите прикажани на Табела 5.

**Табела 5: Оптимални вредности на параметрите на RandomForestRegression**

| | |
|---|---|
| **max bins** | 32 |
| **max depth** | 5 |
| **number of trees** | 20 |

Методот *LogisticRegression* исто така содржи параметри што може да се подесат, како на пример параметарот *elastic_net_mix*, параметарот за регулација, параметарот на толеранција итн. Како влезна колона на методот се користи *Competitive_Bid_label* и *features* колоната.

По соодветното подесување на параметрите, под-моделите се подготвени за проверка на ефективноста на моделот. За добивање на претпоставки користејќи го тренирачкото множество се користи методот *Transform*. Ефективноста на моделот се евалуира со споредба на предвидените резултати со вистинските резултати од базата на податоци.

### 4.3.3 Модел 3: Подобрување на претпоставките на модел со користење на алгоритмот линеарна регресија

#### 4.3.3.1 Цел за креирање и оптимизација на моделот

Креирање и оптимизирањето на соодветен модел со користење го алгоритмот линеарна регресија е со цел подобрување на скалабилноста и оптимизирање преку додавање на нови карактеристики на алгоритмот како влезни податоци. Моделот се евалуира со користење на неколку евалуциони метрики, доведувајќи до заклучок дека што повеќе карактеристики на моделот се користат за влез, толку е претпоставката поточна, што се потврдува преку споредба на резултатите од метриките за евалуација.

#### 4.3.3.2 Опис на податоците

Во моделот се користи „Household Living-costs Price Indexes: December 2017 quarter – Expenditure weights“ базата на податоци со 3179 редови и 13 колони. Базата на податоци е во .csv формат, каде што се претставени индексите на цените за животни трошоци за домаќинствата во Нов Зеланд преку следните колони: hlpi_name, year, hlpi, nzhec, nzhec_name, nzhec_short, level, weight, nzhec1, nzhec1_name, nzhec1_short, exp_pw и eqv_exp_pw (New Zeland Goverment, 2017). Примерок од базата на податоци е прикажан на Табела 6.

**Табела 6: Примерок од базата на податоци за модел 3**

| Hlpi name | year | hlpi | nzhec | Nzhec name | Nzhec short | level | weight | nzhec1 | nzhec1 name | nzhec1 short | Exp pw | Eqv exp pw |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All households | 2008 | allhh | 1 | Food | Food | group | 18 | 1 | Food | Food | 146.35 | 86.76 |
| All households | 2008 | allhh | 1.1 | Fruit and vegetables | Fruit & veg | subgroup | 2.6 | 1 | Food | Food | 21.14 | 12.53 |
| All households | 2008 | allhh | 1.2 | Meat, poultry and fish | Meat | subgroup | 3.1 | 1 | Food | Food | 25.2 | 14.94 |

#### 4.3.3.3 Трансформација на податоците

Трансформацијата на податоците од моделот започнува со вчитување на базата на податоци користејќи го методот *ReadDataFrame*. По вчитување на податоците се повикува *AssembleVector* методот што извршува соединување на повеќе колони во форма на вектор. Во моделот со помош на овој модул се креира нова колона наречена *features* што ги содржи вредностите од колоните *nzec_short* и *level*. Целната колона е *ex_pw* чија вредност е споредена по евалуацијата и добиените резултати од моделот.

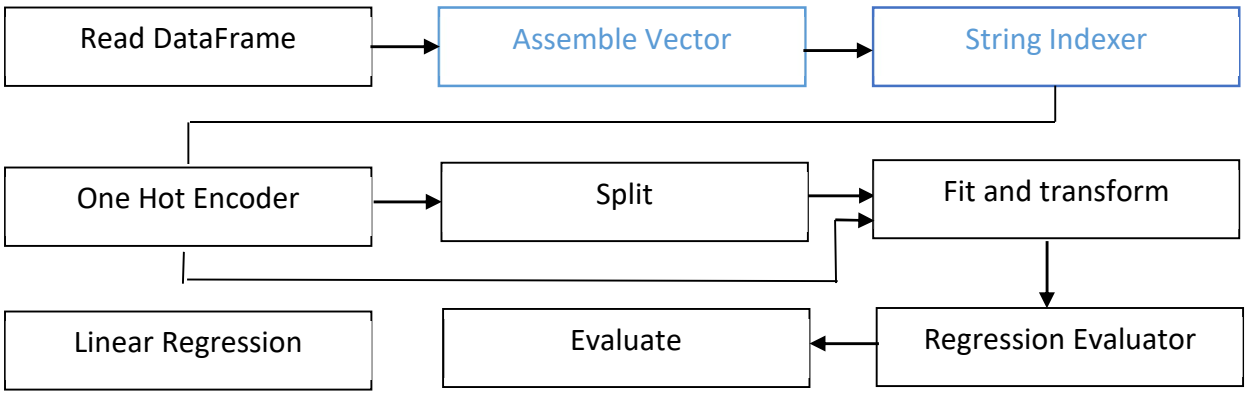
Поради самата природа на алгоритмот линеарна регресија, вредностите од колоната *nzec_short* се трансформираат од текстуален во нумерички формат. За таа цел, при трансформацијата на податоците се користи методот *StringIndexer*.

#### 4.3.3.4 Тренирање на моделот

Податоците од моделот се делат на тренирачко и тестирачко множество со помош на *Split* методот во сооднос од 0.5, односно 50% од податоците би припаднале на тренирачкото множество. Сооднесот е избран со цел добивање на поголем простор за тестирање на податоците. Во овој метод почетниот параметар е поставен на $0$, што значи дека се користи методот на случајни броеви за влез во генераторот на случајни броеви, бидејќи користењето на фиксна вредност на овој параметар би значело добивање на неверодостоен резултат од тренирањето.

Методот *OneHotEncoder* се користи за креирање на вектор од секоја вредност на оригиналната колона. Понатаму, колоните се трансформираат во колона со бинарен вектор (колона *features*) користејќи го методот *AssembleVector*. Моделот е поделен на тренирачко и тестирачко множество за креирање на претпоставки со помош на алгоритмот линеарна регресија.

Методот *LinearRegression* се користи за креирање на претпоставки од поставените податоци на моделот. Како влез во методот се внесува карактеристиката *features,* што понатаму се креира нова колона со предвидени вредности, односно *prediction*. Како клучен параметар во методот се додава карактеристиката *ex_pw* што е потребна за евалуација на моделот. Моделот е прикажан на Слика 36.




**Слика 36:** Процес на развој на моделот 3

#### 4.3.3.5 Оптимизација на моделот со користење на алгоритмот линеарна регресија

Оптимизацијата на моделот подлежи кон додавање на останатата карактеристика од базата на податоци, односно вредностите од карактеристиката *hlpi*. Поради самата природа на алгоритмот линеарна регресија, вредностите од оваа карактеристика се трансформираат од текстуален во нумерички формат, користејќи го методот *StringIndexer*.

Со помош на методот *OneHotEncoder* се креира вектор од индексираните колони. Понатаму, колоните се трансформираат во колона со бинарен вектор, со користење на *AssembleVector* методот.

# 5 АНАЛИЗА И РЕЗУЛТАТИ ОД ОБРАБОТКАТА НА МОДЕЛИТЕ СО ПОМОШ НА АЛГОРИТМИТЕ ЗА МАШИНСКО УЧЕЊЕ

## 5.1 МОДЕЛ 1: ОПТИМИЗАЦИЈА НА МОДЕЛОТ ПРЕКУ СПОРЕДУВАЊЕ НА ПАРАМЕТРИТЕ СО ПРИМЕНА НА АЛГОРИТМОТ РЕГРЕСИЈА НА СЛУЧАЈНИ ШУМИ

### 5.1.1 Анализа и резултати

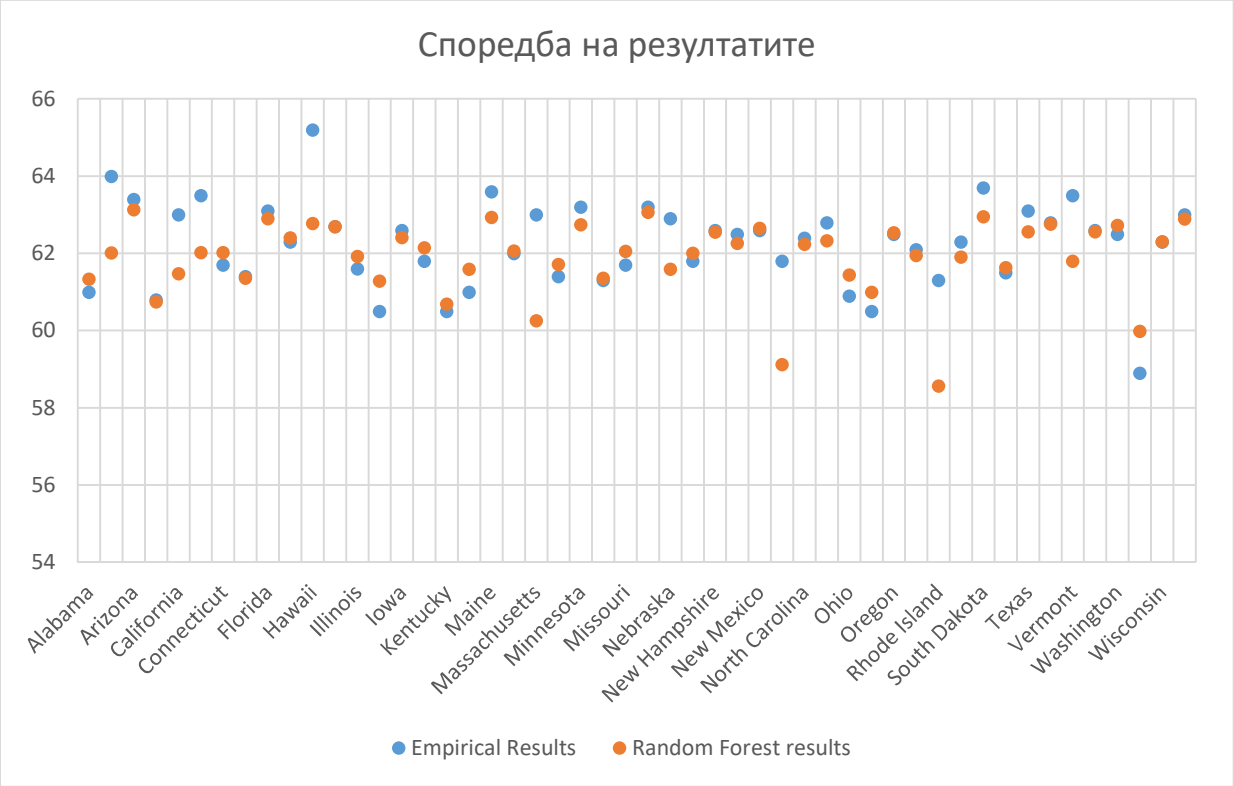
Резултатот од моделот е прикажан во форма на вектор каде што се прикажува значајноста на параметрите кои делуваат на квалитетот на живот на Американците по држави, односно карактеристики кои го дефинираат Well-being индексот. Вредностите од $n$-тата ќелија на излезот покажува колку $n$-тата карактеристика влијае врз целокупната претпоставка, претставено на Табела 7.

**Табела 7: Табеларен приказ на резултатот за најзначајните карактеристики кои делуваат на Well-being Index, подредени по значајност**

| Карактеристика | Резултат |
|---|---|
| Arts, Entertainment, Sports and Media Employment per 1000 | 0.306288755 |
| Total Professionally Active Physicians per 1000 Population | 0.260692449 |
| National Register of Historic Places Listings | 0.136332315 |
| Number of National Historic Landmarks | 0.135836522 |
| Number of America's Byways | 0.088064347 |
| Number of National Parks | 0.072785611 |

Резултатите прикажани во Табела 7 покажуваат дека карактеристиката „*Arts, Entertainment, Sports and Media Employment per 1000*“ најмногу влијае врз квалитетот на живот на Американците во сите држави. Вредностите подлежат на подобрување доколку се направат дополнителни тестови со менување на параметрите во *Fit* методот.

По добиените резултати, моделот подлежи на тестирање со цел да се добие претпоставка за квалитетот на живот на Американците врз основа на својствата кои се тренираа со помош на *Fit* методот, прикажано на Слика 37.




**Слика 37:** Споредба на емпириските резултати со резултатите добиени од моделот

Резултатите прикажани на Слика 37 се добиени со помош на користење на стандардна девијација од грешките на претпоставките, односно со помош на квадратниот корен од средната грешка (RMSE). Грешките од претпоставките се мерка за максималната разлика од регресијата во споредба со емпириските вредности. Во резултатите добиени од моделот може да се забележи помалку значајно отстапување на резултатите, односно резултатот на грешката не е поголем од $2.79$. Како додатна метрика во моделот се користи и квадратниот корен од средната грешка (MAE), а вредноста добиена со оваа метрика, односно апсолутната вредност на разликите помеѓу претпоставената вредност и вистинската вредност е $1.178$.

**Табела 8: Табеларен приказ на резултатот од Студентовиот t-тест**

| | **Променлива 1** | **Променлива 2** |
|---|---|---|
| Mean | 62.248 | 61.91153902 |
| Variance | 1.224587755 | 0.917541127 |
| Observations | 50 | 50 |
| Pearson Correlation | 0.614898048 | - |
| Hypothesized Mean Difference | 0 | - |
| df | 49 | - |
| t Stat | 2.598112557 | - |
| P(T<=t) one-tail | 0.006173105 | - |
| t Critical one-tail | 1.676550893 | - |
| P(T<=t) two-tail | 0.012346211 | - |
| t Critical two-tail | 2.009575237 | - |

Како додатна евалуциона метрика се користи студентовиот тест „Paired Two sample for Means“ (Statistics - how to official website, 2018) со претпоставка дека средната разлика помеѓу емпириските вредности и резултатите добиени од моделот немаат значителна разлика. Според резултатите до тестот (Табела 8), сè додека „t Stat“ вредноста е поголема од вредноста на „t Critical“, нултата хипотеза е одбиена со степен на доверба од 95% што покажува дека разликата на емпириските со вредностите добиени од моделот немаат значителна разлика.

Исто така, бидејќи вредноста на $p$ е помала од вредноста на $\alpha$ ($\text{p-value} < 0.05 = \alpha$ ) се извлекува истиот заклучок (сличен резултат се добива со користење на $\alpha = 0.1$).

### 5.1.2 Дискусии и препораки за подобрување на моделите

Целта на оптимизацијата на моделот е да се прикаже како примената на алгоритмот за машинско учење регресија на случајни шуми може да се искористи за решавање на одредени проблеми преку моделирање, анализа и визуелизирање на големи податоци со користење на Spark. Главната цел на користењето на алгоритмите за машинско учење е подобрување на квалитетот на резултатите и брзината на обработката на самите податоци.

Основна цел на алгоритмот е предвидување на најзначајните карактеристики што ја опишуваат вредноста на претпоставка, подредени по значајност. Во моделот, трите најзначајните карактеристики што влијаат на квалитетот на живот на Американците се: „Arts, Entertainment, Sports and Media Employment per 1000“, „Total Professionally Active Physicians per 1000 Population“ и „National Register of Historic Places Listings“.

Во процесот на тренирање на моделот се користи методот за оптимизација *GridSearch*, преку кој се избрани најоптималните вредности на хиперпараметрите кои понатаму се искористени за тестирање на моделот.

По тестирање на моделот, резултатите покажуваат дека предвидената вредност, односно вредноста добиена од моделот е блиска со емпириските резултати. Се разбира, истите може да се тестираат, односно подобрат во иднина преку подесување на параметрите од алгоритмот преку поставување на друга техника за пресметување на резултатите, како што е на пример RandomSearch или пак SmartHyperparametarTuning.

Степенот на грешка пресметана со евалуционата метриката RMSE е $2.79$, што ја покажува максималната разлика од емпириските и пресметаните вредности во моделот. Исто така, апсолутната вредност помеѓу прогнозираните вредности и емпириските вредности е $1.178$, пресметана со помош на евалуционата метрика MAE.

Во иднина, моделот може да се анализира во детали со цел минимизирање на стапката на грешка и да се истражи проблемот што се јавува кај големите девијации на некои вредности. Овие девијации треба да се оптимизираат со помош на дополнително тренирање на моделот со додатни вредности кај параметрите за тестирање, но потребно е да се обрни посебно внимание во бројот на параметрите, конкретно кај *GridSearch* методот, бидејќи доколку креираат модели со многу повеќе параметри, резултатите добиени од тестирањето на основниот модел може да бидат и неизводливи.

## 5.2 Модел 2: Оптимизација и верификување на ефективноста на моделот преку користење на алгоритмите регресија на случајни шуми и логистичка регресија

### 5.2.1 Анализа и резултати

Методот *Fit* се користи за враќање на модел за тренирање. Во под-моделот каде што се користи алгоритмот регресија на случајни шуми, како излез се добива вектор со значајност. Векторот со значајност ги покажува најзначајните карактеристики што ја дефинираат *CompetitiveBid* колоната, подредени по значајност и се прикажани на Табела 9.

**Табела 9: Табеларен приказ на резултатот за најзначајните карактеристики кои делуваат на Competitive Bid, подредени по значајност**

| Карактеристика | Резултат |
|---|---|
| State | 0.0415439 |
| City | 0.000420416 |
| Address | 0.000284102 |

Резултатите од табелата покажуваат дека карактеристиката *State* најмногу влијае на целната колона. Вредноста може да се подобри доколку со тестирање на различни вредности од методот.

Под-моделот регресија на случајни шуми се евалуира користејќи стандардна девијација од предвидените грешки со користење на RMSE метриката за евалуација. Грешката на предвидување покажува дека максималната разлика на регресијата во споредба со резултатите на моделот е со помалку значајно отстапување, односно $0.121$.

Излезот од *LogisticRegression* методот креира три нови колони: *raw_prediction*, што претставува колона со ниво на доверба, *probability* – колона за предвидени условни веројатности и *prediction* колона што се креира за време на постигнувањето на моделот. Користејќи SQL трансформации се комбинираат вредностите од колоната *prediction* и *competitive_bid_label*. Извештајот од моделот покажува дека не постојат резултати каде што вредноста на *prediction* е $0$ и вредноста од *competitive_bid_level* е $1$, односно резултати каде што вредноста на *prediction* е $1$ и вредноста на *competitive_bid_level* е $0$, што значи дека моделот дава добри резултати, односно:

$$prediction = competitive\_bid\_label$$

### 5.2.2 Дискусии и препораки за подобрување на моделите

Целта на оптимизацијата на моделот е да се докажи дека со комбинација на два алгоритми за машинско учење, не само што се верификуваат резултатите, туку и се покажуваат главните фактори коишто делуваат на резултатите односно на целната колона, што остава простор за поширока дискусија.

По својата природа, алгоритмот регресија на случајни шуми со сите свои можни параметри директно индицира врз причинителите на резултатот од целната колона. Со овој алгоритам се покажа дека карактеристика *State* најмногу и најдиректно влијае врз предвидената карактеристика на моделот, односно врз *CompetitiveBid*. Резултатите од овој под-модел се евалуирани со помош на квадратниот корен од средна грешка. Вредноста произлезена од метриката е $0.121$ што ја покажува ниска максимална разлика помеѓу емпириските вредности и вредностите добиени од моделот.

Со помош на алгоритмот логистичка регресија се верификуваат резултатите од моделот, односно од вредностите на целната колона. Резултатите од алгоритмот покажуваат дека не постои разлика во вредностите помеѓу реалните вредности и претпоставените вредности добиени со алгоритмот, што може да се заклучи дека извршувањето на моделот е солидно.

Во иднина, моделите може да подлегнат кон дополнителна оптимизација преку додатно тестирање со различни комбинации за оценување на перформансите на секој модел. Доколку би се оптимизирал моделот во тренирачкото множество, резултатите би биле многу подобри во истото множество, но нема да има влијание врз новите податоци, односно врз податоците од тестирачкото множество. Односно, доколку перформансноста на моделот се подобри во тренирачкото множество, но не се подобри во тестирачкото множество, проблемот само би се зголемил. За надминување на овој проблем, во иднина потребно е користење на вкрстена валидација на целосниот модел со цел подобра оптимизација на моделот.

## 5.3 Модел 3: Подобрување на претпоставките на модел со користење на алгоритмот линеарна регресија

### 5.3.1 Анализа и резултати

Како последниот чекор за креираниот модел е евалуацијата на резултатите од моделот. Имајќи ги во предвид метриките за евалуација квадратниот корен од средната грешка и средна апсолутна грешка, резултатите од моделот каде што се користат помалку карактеристики се прикажани во колоната *prediction* (Табела 10).

**Табела 10: Резултати од предвидувањата во моделот со помалку карактеристики**

| Hlpi name | year | hlpi | nzhec | … | nzhec1 short | Exp pw | Eqv exp pw | Features | Prediction |
|---|---|---|---|---|---|---|---|---|---|
| All households | 2008 | allhh | 1.1 | … | Food | 21.14 | 12.53 | (69, [(9, 1), (18, 1), (68, 2.6)]) | 25.7937 |
| All households | 2008 | allhh | 2.2 | … | Alcohol and tobacco | 17.89 | 10.6 | (69, [(9, 1), (59, 1), (68, 2.2)]) | 21.1445 |
| All households | 2008 | allhh | 3 | …. | Clothing and footwear | 27.64 | 16.39 | (69, [(9, 1), (64, 1), (68, 3.4)]) | 32.9329 |

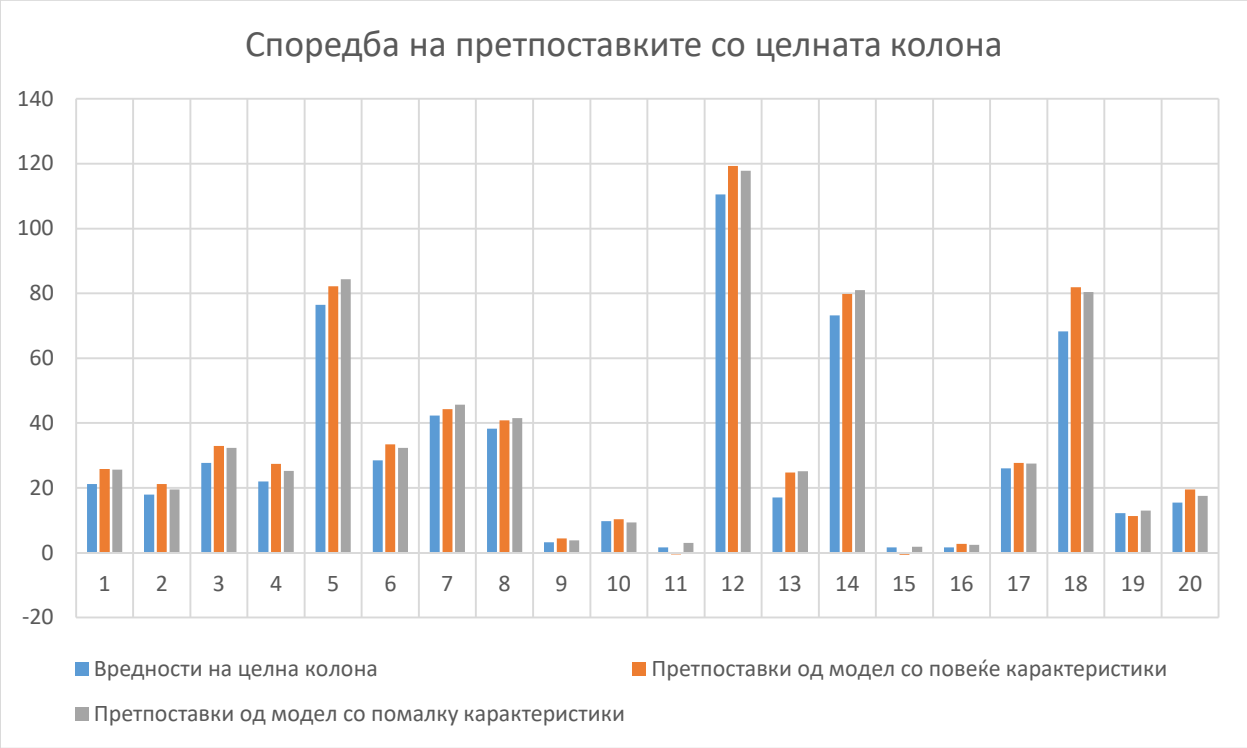
Резултатот со претпоставки каде што се користи додатната карактеристика *hlpi* пресметани со метриките за евалуација се прикажани на Табела 11.

**Табела 11: Резултати од предвидувањата во моделот со додатни карактеристики**

| Hlpi name | year | hlpi | nzhec | … | nzhec1 short | Exp pw | Eqv exp pw | Features | Prediction |
|---|---|---|---|---|---|---|---|---|---|
| All households | 2008 | allhh | 1.1 | … | Food | 21.14 | 12.53 | (57, [(5, 1), (55, 1), (56, 2.6)]) | 25.5843 |
| All households | 2008 | allhh | 2.2 | … | Alcohol and tobacco | 17.89 | 10.6 | (57, [(46, 1), (55, 1), (56, 2.2)]) | 19.4571 |
| All households | 2008 | allhh | 3 | …. | Clothing and footwear | 27.64 | 16.39 | (57, [(51, 1), (56, 3.4)]) | 32.3704 |

Метриките за евалуација во моделот ја пресметуваат грешката на моделот користејќи ги карактеристиките *prediction* и *exp_pw* како параметри што ја мерат грешката помеѓу двете карактеристики.

Резултатите од карактеристиката *prediction* каде што се користат повеќе и помалку карактеристики се споредени со резултатната колона и прикажани на Слика 38.




**Слика 38:** Споредба на резултатите од претпоставките со вредностите од целната колона за првите 20 резултати

Точноста на алгоритмот линеарна регресија се основа врз вредностите од грешките на евалуацијата. Резултатите од тестовите покажуваат дека со користење на трите регресиони метрики за пресметка на грешката на моделот, секоја додатна карактеристика во моделот значи и негово подобрување. Евалуацијата на моделот покажува дека постои мала, но сепак значителна доколку се користат помалку или повеќе карактеристики во моделот, прикажано на Табела 12.

**Табела 12: Резултати од евалуацијата на моделот**

| | **RMSE** | **MAE** |
|---|---|---|
| **Модел со помалку карактеристики** | 24.4205 | 13.0028 |
| **Модел со повеќе карактеристики** | 19.1327 | 11.236 |
| **Δ** | 5.2878 | 1.7668 |

### 5.3.2 Дискусии и препораки за подобрување на моделот

Целта на алгоритмот линеарна регресија е да се минимизира вредноста на сумата од квадратот на остатоците, со други зборови, минимизирање на разликата помеѓу предвидените и надгледаните вредности. Иако моделите што се креираат со помош на алгоритмот се доста обични, кога станува збор за креирање и оптимизација на модели со големи податоци самата природа на алгоритмот има потреба од користење на минимална меморија.

Резултатите од алгоритмот во моделите покажуваат дека моделот е многу ефективен и точен во предвидувањето на карактеристиката *exp_pw*. Доколку истиот алгоритам се примени на поголема база на податоци, бројот на карактеристики како влезни параметри значително може да ја подобри вредноста на претпоставките.

Алгоритмот може да има широка примена кај големите податоци и алгоритмите за машинско учење бидејќи самиот алгоритам е толерантен на грешки што ја подобрува скалабилноста на моделите.

Во иднина кај овој алгоритам потребно е да се посвети повеќе внимание во минимизирање на сумата од квадратни девијации помеѓу податоците од тренирачкото и податоците од тестирачкото множество. Користење на различен метод за оптимизација на моделот можеби директно би влијаел врз резултатите од оптимизацијата на моделот. Исто така, изготвувањето на стратегија за подесување на параметрите на моделот би водела кон подобрување и кон подобра оптимизација на моделот.

# 6 ДИСКУСИЈА НА ЗАКЛУЧНИТЕ СОГЛЕДУВАЊА И ПРЕПОРАКИ ЗА ИДЕН РАЗВОЈ НА АЛГОРИТМИТЕ ЗА КЛАСИФИКАЦИЈА НА ГОЛЕМИТЕ ПОДАТОЦИ

Секоја година се адресираат нови отворени проблеми и области во обработката и анализата на податоци, каде како можно решение е искористување на алгоритмите за машинско учење. Самата природа на науката за големи податоци е интердисциплинарна и секојдневно се надоградува, односно се зголемува спектарот на проблеми и можни решенија. Бидејќи податоците секојдневно се зголемуваат, потребно е целосно да се смени процесот на нивна анализа, со други зборови, потребно е да се размислува пошироко од системите со затворена архитектура.

Обработката на податоците, нивната анализа и визуелизација со користење на различни алгоритми за машинско учење не треба да значи само нивно искористување само во големите системи, туку и искористување во обичните апликации за секојдневна употреба. Со денешниот развој на технологијата постојат најразлични open-source сервиси за анализа и обработка на податоци, со што се прошируваат можностите за нивна надоградба и изнаоѓање на полесни решенија за одредени проблеми при креирање на модели за различни проблеми.

Во процесот на обработка на податоците, самото додавање на повеќе колони во множеството од податоци, но не и повеќе редови веќе станува опасно. Со зголемувањето на бројот на променливите, сè повеќе карактеристики стануваат зависни една од друга што води кон слаба перформансност на развиениот модел. Податоците потребно е да се се истражат во целост, со други зборови, не треба да се прекине кога ќе се добие успешен резултат, што е еден од најчестите случаи со големите податоци, водејќи кон погрешни резултати. Како што Nassim Taleb вика: „Не велам дека нема информации за големите податоци. Има премногу податоци. Проблемот е дека потрагата по иглата во сено се наоѓа во сè поголем стог од сено“[34].

[34] "Beware the Big Errors of Big Data" Wired, February 2013

Алгоритмите за надгледувано учење се можеби основната и најбараната методологија во машинското учење. Техниките развиени за овие алгоритми се посуериорни од техниките што се користат кај алгоритмите за ненадгледувано учење, бидејќи со нив се добиваат јасни критериуми од тренирачкото множество за оптимизација на различни модели. Еден од најголемите предизвици кај овие алгоритми е минимизирање на ризикот за задоволување на одредени критериуми при процесот на оптимизација на моделите.

Машинското учење, како дисциплина, сè уште привлекува голем интерес при анализата на големите податоци и при искористувањето на нејзините алатки од страна на компаниите. Иако алгоритмите во оваа област се покажуваат како доста скалабилни во обработката на големи податоци кои се солидно теоретски поткрепени, тие сè уште претставуваат предизвик за нивно специфично искористување за одреден проблем. Оптимизацијата на моделите што користат алгоритми за машинско учење е критична за успешна анализа на податоците, тргнувајќи од обичните апликации, па сè до комплексните системи со големи податоци.

Рамката Apache Spark, како моќна алатка за интерактивно податочно рударење, е една од најефективните рамки за обработка на податоци користејќи различни методи за оптимизација и подобрување на модели. Можноста за повикување на веќе постоечки Java библиотеки и можноста за пристап до одреден Hadoop податочен систем претставува една од најдобрите карактеристики на Spark. Во таа насока претстои период на развој и имплементација на скалабилни алгоритми за машинско учење за реални проблеми за веќе имплементираните алгоритми во реални апликации, како и подобрување на постоечките методи за анализа, нова обработка и нова анализа на нивните податоци. Исто така, претстои период на креирање на нова модуларна платформа за обработка на големи податоци, што ќе се основа на Apache Spark, со додатна визуелизација на податоците. Можната оптимизација и адаптација на одреден алгоритам за машинско учење за одредени студии на случај, води кон подобрени и подетални анализи.

Науката за податоци не подразбира само градење на модели. Градењето на модели е само дел од цевководот на науката за податоци, кој исто така вклучува: идентификација на целите на проектот/бизнисот, разбирање на податоците преку нивно собирање и преглед, подготовка на податоците преку нивна селекција и прочистување, моделирање, евалуација и анализа.

Со оваа дисертација, со практична примена на неколку студии на случај во модуларна тестна и лесно скалабилна околина којашто во основа ја има рамката Apache Spark, со користење на слободни бази на податоци и алгоритмите за класификација на податоци развиени кај машинското учење, направени се анализи во насока на подобрување на перформансите на големите податоци во процесот на нивна обработка.

Изборот на алгоритам за машинско учење во решавање на одреден проблем пред сè зависи од повеќе фактори, меѓу кои: големината, квалитетот и природата на базата на податоци, достапното време за пресметување и што е најбитно, што треба да се постигне со податоците. Како и да е, постоечките алгоритми за машинско учење продолжуваат да се развиваат и еволуираат, а нивната примена најмногу зависи од излезот на податоците.

Алгоритмите за машинско учење се покажаа како доста скалабилни користејќи ја Apache Spark рамката. Истите доста лесно може да се искористат во градењето на модели. Согледувајќи ја самата околина и резултатите од моделите може да се заклучи дека постои потенцијал во скалирање на комплексноста на текот на апликациите. Бидејќи се користи open-source апликација основана на Spark, времето на обработка на податоците за нивна анализа потенцијално може да се зголеми доколку се користи чистата околина како Spark, на начин што ќе се искористи целокупниот потенцијал на рамката.

Резултатите добиени од истражувањето и алгоритмите за машинско учење покажуваат кон лесна и сеопфатна примена за обработка на податоци и креирање на препораки за подобрување на веќе постоечките модели во подобрување на решавањето на практични проблеми за одредени студии на случај.

Параметрите за оптимизација кај алгоритмите за машинско учење најчесто содржат стандардни вредности кои веќе се зададени во рамката за големи податоци или пак може лесно да се подесат со нови вредности. Вредностите на параметрите имаат големо влијание врз перформансите за предвидување на моделите.

Користењето на одредени методи за оптимизација директно влијаат врз резултатите од моделот. Преку примената на различни алгоритми на различни модели, оптимизацијата значи и подобрување на резултатите, потврдено преку процесот на евалуирање на самите модели. Примената на одредени метрики за евалуација врз моделите не значи само потврдување на точноста од резултатите на моделот. Резултатите од евалуацијата може да водат и кон дополнително тестирање на моделот, вклучувајќи и тестирајќи го истиот со нови методи за оптимизација, или пак комплетно трансформирајќи го истиот со цел добивање на сè поточни резултати.

Дел од процесот на оптимизација и подобрување прикажан во дисертацијата преку подесување на различните параметри кои се составен дел од алгоритмите претставува засебна конфигурација чија вредност не може да се предвиди од типот на самите податоци. Исто така, најточната вредност за параметрите на моделот за одреден проблем не може лесно да се дознае. Кога станува збор за подесување на одредени параметри во алгоритмите за машинско учење, подесувањето на параметрите исто така значи откривање на параметрите од моделот што резултираат во најдобрите претпоставки за одреден проблем. Најчесто параметрите од моделот се проценуваат со помош на користење на соодветна стратегија за оптимизација на алгоримот, што претставува ефективен тип на пребарување низ можните вредности од параметрите.

За идна работа, во поглед на подобрување на моделите што ги користат алгоритмите за машинско учење, потребно е да се развие конкретна стратегија и избор на соодветен метод за оптимизација за идентификување и подесување на параметрите во зависност од природата на алгоритмот за машинско учење и во зависност од проблемот на кој се работи, што ќе води кон подобро предвидување на резултатите и кон подобрување на конкретниот

модел. Исто така, во иднина потребно е да се креира нова модуларна платформа која ќе се основа на автоматско градење на стратегија за оптимизација на моделите и добивање на оптимални резултати, односно претпоставки.

Се разбира, ова истражување претставува почеток на низа истражувања во областа на алгоритмите за класификација на големи податоци за подобрување на перформансите при нивната обработка.

## Библиографија

[1] *Amazon Web Services, Inc. or its affiliates*. (2018). Retrieved from https://aws.amazon.com/

[2] Berkhin, P. (2006). A survey of clustering data mining techniques. *Grouping multidimensional data* , 25-71.

[3] Breiman, L. F. (1984). *Classification and Regression Trees.* Wadsworth, Belmont, CA,: Chapman & Hall, New York.

[4] Cade, M. (2010). Google behavioral ad targeter is a Smart Ass. *The Register*.

[5] Cárdenas, A. A. (2013). *Big data analytics for security intelligence.* University of Texas at Dallas@ Cloud Security Alliance.

[6] *Cloudera*. (2018). Retrieved from https://www.cloudera.com/products.html

[7] Cunningham, P. C. (2008). Supervised learning. Berlin, Heidelberg: Machine learning techniques for multimedia, Springer.

[8] Das, S. R. (2013). *Data Science: Theories, Models, Algorithms and Analytics, a web book.*

[9] Durgesh, K. S. (2010). Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology 12.1*, 1-7.

[10] Elgendy, N. (2013). *Big Data Analytics in Support of the Decision Making Process, MSc Thesis.*

[11] Elgendy, N. a. (2014). Big data analytics: a literature review paper. *Industrial Conference on Data Mining, Springer, Cham*.

[12] Folkers, C. (2016). *Scalable machine learning algorithms on a big data infrastructure.*

[13] GC, D. (2016). A Critical Comparison of NOSQL Databases in the Context of Acid and Base. *Culminating Projects in Information Assurance*.

[14] *Google drive official website*. (2018). Retrieved from https://www.google.com/drive/

[15] Guo, G. e. (2003). KNN model-based approach in classification. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 986-996.

[16] Gupta, P. A. (2016). Scalable machine-learning algorithms for big data analytics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 6.6*, 194-214.

[17] *Hackernoon website*. (2017). Retrieved from https://hackernoon.com/

[18] *Health Data*. (2015). Retrieved from A federal government website managed by the U.S. Department of Health & Human Services: https://www.healthdata.gov

[19] *Hortonworks* . (2017). Retrieved from https://hortonworks.com/products/data-platforms/hdp/

[20] *IBM Official website*. (2017). Retrieved from https://www-01.ibm.com/software/in/data/bigdata/enterprise.html

[21] IntellifySolutions. (2017). *Intellify Solutions*. Retrieved from http://intellifysolutions.com/?p=2469

[22] *IoT one*. (2017). Retrieved from https://www.iotone.com/vendor/ibm-watson-ibm/casestudies/v928

[23] Kantardzic, M. (2011). *Kantardzic, Mehmed. Data mining: concepts, models, methods, and algorithms.* John Wiley & Sons.

[24] Karau, H. K. (2015). *Learning spark: lightning-fast big data analysis.* O'Reilly Media, Inc.

[25] Labrinidis, A. a. (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment, 5(12)*, 2032-2033.

[26] Lewis, R. J. (2000). An Introduction to Classification and Regression Tree (CART) Analysis. *Annual Meeting of the Society of Academic Emergency Medicine in*.

[27] Lugmayr, A. S. (2017). Cognitive big data: survey and review on big data research and its implications. What is really "new" in big data? *Journal of Knowledge Management 21(1)*, 197-212.

[28] *MapR Technologies, Inc. All Rights Reserved* . (2017). Retrieved from https://mapr.com/why-hadoop/why-mapr/enterprise-platform/

[29] Meng, X. e. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research 17.1*, 1235-1241.

[30] Murphy, K. P. (2012). *Machine learning, a probabilistic perspective.* Cambridge: The MIT Press.

[31] *New Zeland Goverment*. (2017). Retrieved from https://www.stats.govt.nz

[32] *ODbL*. (2016). Retrieved from http://opendefinition.org/licenses/odc-odbl/

[33] Panda, B. H. (2009). Planet: massively parallel learning of tree ensembles with mapreduce. *Proceedings of the VLDB Endowment 2.2*, 1426-1437.

[34] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from Naive Bayes: http://scikit-learn.org/stable/modules/naive_bayes.html

[35] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from Decision tree: http://scikit-learn.org/stable/modules/tree.html

[36] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from Random Forest Rregression: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[37] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from SVM: http://scikit-learn.org/stable/modules/svm.html

[38] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from Grid Search: http://scikit-learn.org/stable/modules/grid_search.html

[39] Pedregosa, e. a. (2011). *Scikit-learn: Machine Learning in Python*. Retrieved from KNN: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[40] Pedregosa, e. a. (2017). *Scikit-learn: Machine Learning in Python*. Retrieved from Logistic Regression: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[41] Schneider, A. H. (2010, November). Linear regression analysis: part 14 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 776. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2992018/

[42] Schubert, L. K. (1988). Cheeseman: A travesty of truth. Computational Intelligence. *Computational Intelligence*, 118-121.

[43] Services, E. E. (2012). *EMC: Data Science and Big Data Analytics.* Canada: Wiley.

[44] Shalev-Shwartz, S. a.-D. (2014). Understanding machine learning: From theory to algorithms. *Cambridge university press*.

[45] *Statistics - how to official website*. (2018). Retrieved from http://www.statisticshowto.com/t-test

[46] *Towards Data Science*. (2017). Retrieved from https://towardsdatascience.com/

[47] Willmott, C. J. (2005). dvantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, 79-82.

[48] Zikopoulos, P. (2015). *Big data beyond the hype: A guide to conversations for today's data center.* Mc Graw Hill Education.

[49] Zikopoulos, P. a. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data.* McGraw-Hill Osborne Media.

[50] Zou, K. H. (2003). Correlation and simple linear regression. *Radiology*, 617-628.

## Листа со табели

## Листа со слики

## Листа со кодови

# Додатоци

## Додаток 1: Околина за тестирање

Во дисертација како околина за тестирање се користеше апликацијата Seahorse[35] што претставува визуелна рамка за креирање на Apache Spark апликации преку визуелен пристап. Секој метод што е овозможен од Apache Spark во Seahorse е претставен преку објект, што преку визуелен пристап лесно може да се креираат различни модели кои понатаму подлежат кон дополнителни анализи.

Освен како самостојна апликација, меѓу предностите на Seahorse се вбројува и тоа што апликациите, односно моделите, може да се креираат и преку веб апликација, преку пребарувачи како што се Chrome или Mozilla. Seahorse исто така овозможува поврзување до било кој кластер, како што е YARN[36], Mesos[37] или Spark Standalone. Seahorse нуди алатки преку кои може да се решаваат и анализираат проблеми со големи податоци благодарение на визуелниот и лесен за користење интерфејс.

Со помош на оваа визуелна рамка можно е да се креираат комплексни протоци на податоци за екстракција, прочистување, трансформација и читање на податоци (ECTL) и искористување на алгоритмите за машинско учење дури и без напредно познавање на Spark. Во Seahorse може да се користат алатки за решавање на проблеми со големи податоци што е овозможено преку едноставниот кориснички интерфејс.

Во Seahorse се користи визуелниот пристап на програмирање, така што секоја апликација и модел креиран преку рамката е лесно разбирлив. Визуелното програмирање како метод за креирање на модели не дава ограничување при користењето на нови или пак специфични акции за самите модели. Акциите или пак трансформациите што не се

[35] https://github.com/deepsense-io/seahorse-workflow-executor
[36] https://yarnpkg.com/lang/en/docs/install/
[37] https://mesosphere.com/

овозможени од оваа визуелна рамка најчесто може да се испишат со користење на Python или пак R.

Апликациите креирани преку визуелната рамка и едноставниот веб интерфејс се претставени преку дијаграм од операции што се нарекува работен тек (workflow). Најчеста сесија што се креира преку Seahorse се состои од три фази: додавање на оператори во работниот тек, извршување на делот што е веќе креиран и експортирање на резултатите од извршувањето. Со ова се овозможува интерактивен процес за време на креирање на апликациите и лесен преглед на корисниците да ги следат своите чекори. По креирањето на работен тек на апликацијата, истата може да се експортира и да се постави како самостојна Spark апликација на сервер.

Апликацијата нуди можност за вчитување на податоци од различни извори: податоци коишто се локално поставени, бази на податоци поставени на сервер и бази поставени на cloud, како што се HDFS податочниот систем, или Google Drive (Google drive official website, 2018).

Преку оваа визуелна рамка може да вчитуваат различни типови на податоци: од mySql табели, па се до csv или JSON бази на податоци. При вчитувањето на базата на податоци потребно е да се внимава, на пример доколку базата е на сервер, со колкава брзина ќе биде вчитувањето на податоците, во кој формат ќе биде базата на податоци итн. Нормално, при вчитување на податоците, потребно е да се трансформираат, односно да се пречистат во таков формат што ќе биде најсоодветен за извршување на различни операции со податоците.

За дел од статистичките анализи кои не беа реализирани со апликацијата, креирањето на графиците, итн, се користешње Microsoft Excel 2016[38].

[38] https://products.office.com/en/excel

## Кратенки и акроними

RTAP – Real-time analytical processing

MLlib – Machine Learning Library

IoT – Internet of Things

JSON – JavaScript Object Notation

PB – Petabyte

ACID – Atomic Consistent Isolated Durable

BASE – Basic Availability Soft-stat Eventual consistency

IBM – International Business Machine

MAD – Magnetic Agile Deep

ETL – Extract Transform Load

ETCL – Extract Transform Clean Load

ELT – Extract Load Transform

DFS – Distributed File System

GPS – Global Positioning System

HDFS – Hadoop Distributed File System

MB – Megabyte

MNIST – Modified National Institute of Standards

YARN – Yet Another Resource Negotiator

CAP – Consistency Availability Partition tolerance

SQL – Structured Query Language

B-DAD – Big-Data Analysis and Decisions

EDW – Enterprise Data Warehouse

ADV – Advanced Data Visualization

SNA – Social Network Analysis

RDBMS – Relational Database Management System

API – Application Programming Interface

ALS - Alternating Least Squares

RMSE – Root Mean Square Error

MAE – Mean Absolute Error

SMBO – Sequential Model-Based Optimization