УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

ПРИМЕНА ТЕХНИКА МАШИНСКОГ УЧЕЊА КОРИШЋЕЊЕМ АЛАТА „*JUPYTER NOTEBOOK*“

Дипломски рад

Ментор:

доц. др Дражен Драшковић,

доктор наука – електротехника и рачунарство

Кандидат:

Ђорђе Малешевић
0399/2015

Београд, Септембар 2020.г.

Захваљујем се свом академском ментору др. Дражену Драшковићу за помоћ и разумевање приликом израде дипломског рада. Велику захвалност дугујем и својој породици и блиским људима који ми пружају безусловну и безграничну подршку, пријатељима, колегама и професорима који су учествовали у мом школовању.

САДРЖАЈ

1. Увод

Овај документ представља опис начина примене техника машинског учења на проблеме регресије и класификације. Студент није имао претходна искуства са машинским учењем. Позната је чињеница да машинско учење представља једну од најузбудљивијих области рачунарства у данашњем времену. Компаније као што су *Google, Facebook, Apple, IBM* itd. инвестирају велике суме новца у истраживање и примену машинског учења. Ова област постала је неопходна и у свакодневном животу. Мобилни уређаји поседују сервисе засноване на машинском учењу као што је гласовни помоћник, компаније користе технике машинског учења како би унапредили своје пословање као што је систем за препоручивање производа купцима који је заснован на историји поруџбина тог купца, банке користе системе за спречавање превара кредитним картицама, системи за филтрирање спем порука из електронског поштанског сандучета, системи за дијагностику у медицинским установама такође су засновани на техникама машинског учења итд.

У општој перцепцији машинско учење представља дисциплину која се бави извођењем алгоритама из података, без експлицитног програмирања. Ипак, машинско учење има и своју фундаменталну димензију. Као што се логика бави проучавањем дедукције објашњавајући шта чини неки закључак потпуно оправданим и тиме формализује један важан вид људског закључивања, машинско учење се бави проучавањем индукције, односно генерализације и тиме формализује други вид људског закључивања – уопштавање од ограниченог броја узорака ка универзалним закључцима.

Машинско учење је техника анализе података која омогућава да рачунариураде оно што природно могу и људи и животиње, то је да уче из искуства. Машинско учење може да се подели у четири групе:

1. надгледано учење
2. ненадгледано учење
3. полунадгледано учење
4. учење са подршком

Надгледано учење је вероватно најзначајнија група. Основна карактеристика је да се подаци код надгледаног учења састоје из парова описа онога на основу чега се учи и онога што је потребно научити. Назив је мотивисан аналогијом са процесом учења при којем учитељ ученику задаје задатке, али му након његових одговора даје и тачне резултате како би ученик могао да упореди. Често се поистовећује са принципом учења код људи. Ако дете треба да научи да препознаје мачку на основу фотографије, треба му показати фотографије различитих животиња међу којима су и мачке, и рећи му шта јесте а шта није мачка. Што више фотографија дете види, то ће његова моћ расуђивања бити јача.

Ненадгледано учење се карактерише одсуством информације о томе шта је потребно научити. Ако бисмо објаснили ненадгледано учење на примеру детета које треба да научи како да препозна мачку на слици, онда детету не бисмо говорили, ни за једну фотографију

коју јој прикажемо, која животиња се налази на њој. Заправо, овај вид учења се обично бави проналажењем неке врсте структуре у подацима, а методе које на тај начин уче су обично направљене полазећи од конкретне врсте структуре која се тражи. Проблем кластеровања је један проблем налажења структуре у подацима – структуре група. У многим применама потребно је идентификовати групе података. Примера ради, могу се груписати слични текстови, сличне фотографије, акције чије се цене слично крећу на берзи итд.

Полунадгледано учење представља комбинацију надгледаног и ненадгледаног учења. У пракси има велику вредност јер је за многе проблеме тешко добити довољне количине података који имају информације о томе шта је потребно научити док се учење са подршком користи у ситуацијама када је потребно решити неки проблем предузимајући неке акције чијим се заједничким дејством долази до решења проблема. Избор правог алгоритма може да буде тежак проблем јер постоји велики број алгоритама машинског учења у свакој групи, и сваки од њих има различит приступ у учењу. У овом документу акценат се ставља на надгледано учење.

У поглављу 2 биће детаљно описана анализа проблема проблема и технологије које ће бити коришћене у реализацији. Анализа проблема представља процес машинског учења који се састоји од седам корака које је потребно реализовати како би се направио систем који је заснован на машинском учењу, док технологије представљају окружење помоћу ког ће сваки корак бити спроведен у дело. У поглављу 3 детаљно је описан појам надгледаног учења. Објашњени су појмови означених података који имају улазни/е и циљни/е атрибут/е. Потом су објашњена два типа проблема који постоје у оквиру надгледаног учења, то су класификација и регресија. Наводе се и најпознатији алгоритми надгледаног учења а детаљније су објашњена три алгоритме из те групе. У поглављу 4 кратко је описано помоћу чега и на који начин ће бити реализоване све фазе машинског учења док су у поглављу 5 приказани резултати рада, имплементација и изворни код и дискусија. Имплементирана су два система и за сваки од њих спроведене су у дело све фазе машинског учења: прикупљање података, припрема података, експлораторна анализа, модел, обучавање модела и евалуација. У поглављу шест дат је општи закључак.

2. АНАЛИЗА ПРОБЛЕМА И ТЕХНОЛОГИЈЕ

У сврху анализе проблема, дефиниција машинског учења може се представити на поједностављен начин као „коришћење података да би се дао одговор на одређено питање“. На основу ове врло поједностављене дефиниције могу се извући одређени закључци. Ако се дефиниција подели на два дела: „коришћење података“ и „одговор на одређено питање“, два дела широко осликавају две стране машинског учења, обе подједнако важне. „коришћење података“ је оно што се обично назива „тренирање“, док се „одговор на одређено питање“ назива „предикција“ или „закључивање“. Оно што спаја ова два дела назива се „модел“. Тренирамо модел помоћу скупа података како би модел дао све боље и корисније предикције. Овај предиктивни модел се тада може користити за предикцију над претходно невиђеним подацима. Кључна компонента овог процеса су подаци. Подаци су кључ за откључавање машинског учења, као што је и машинско учење кључ за откључавање знања скривеног у подацима. У овом документу потребно је направити два система где сваки од система треба да пружи одговор на одређено питање. Процес изградње таквих система назива се процес машинског учења.

Први систем служиће компанији која има своје производе које продаје искључиво путем интернета. Купци поручују производе компаније помоћу интернет апликације или помоћу мобилне апликације. Питање на које систем треба да одговори јесте „да ли компанија треба да се фокусира на развој мобилне апликације или интернет апликације како би унапредила своје пословање?“. Овај систем користиће податке везане за регистроване кориснике који користе интернет или мобилну апликацију. Други систем треба да одговори на питање „да ли пацијент има кардиоваскуларне проблеме?“ и користиће податке везане за пацијенте на основу којих ће дати одговор на постављено питање. Процес машинског учења може да се подели у неколико фаза:

1. прикупљање података
2. припрема података
3. експлоративна анализа података
4. модел
5. обучавање модела
6. евалуација

Прикупљање података је врло битна фаза. Квалитет и квантитет података који су прикупљени директно одређује квалитет предиктивног модела. Прикупљање података може бити спроведено помоћу затворених и отворених анкета, фокусне групе, директно посматрање итд. Прикупљени подаци врло често имају тенденцију да буду несавршени тј. да постоји доста непостојећих вредности, вредности које одступају као и вредности које се понављају.

Припрема података се може дефинисати као претпроцесирање сирових података. Сирови подаци су врло често пуни непостојећих и нетачних информација које утичу на

квалитет предикције, па се такви подаци обрађују како би се добила форма која може бити квалитетно анализирана. Кораци који се предузимају у припреми могу бити уклањање или модификација непостојећих вредности, дупликата, одступајућих вредности.

Експлоративна анализа података се користи како би што боље разумели поруке које носе подаци. Једна од метода којом се врши експлоративна анализа података јесте визуелизација података. Визуелизација омогућава да се уоче неке законитости између посматраних података, да се стекне увид у саме прикупљене податке итд. и да се на основу такве анализе што боље претпостави ефикасност предиктивног модела.

Модел представља избор алгоритма машинског учења а под обучавањем модела подразумева се оптимизација модела коришћењем расположивих тренинг података. Шта оптимизација тачно значи зависи од конкретног алгоритма машинског учења. При обучавању модела важно је да се избегну ситуације када модел није довољно прилагођен што значи да није довољно искористио тренинг податке и када је модел превише прилагођен што значи да се превише ослонио на тренинг податке.

Евалуација модела подразумева процес тестирања модела помоћу тест података на основу чега се помоћу одређене метрике одређује тачност предикције модела који се потом упоређује са тачностима других алгоритама који могу да се примене на посматрани проблем. На основу упоређивања тачности модела се бира модел који има највећу тачност.

Оба система биће реализована у интерактивном рачунарском окружењу под називом *Jupyter Notebook*. *Jupyter Notebook* окружење омогућава корисницима да креирају документе који укључују:

- изворни код
- интерактивне виџете
- форматирање текста
- исцртавање графика
- писање математичких формула
- фотографије
- видео

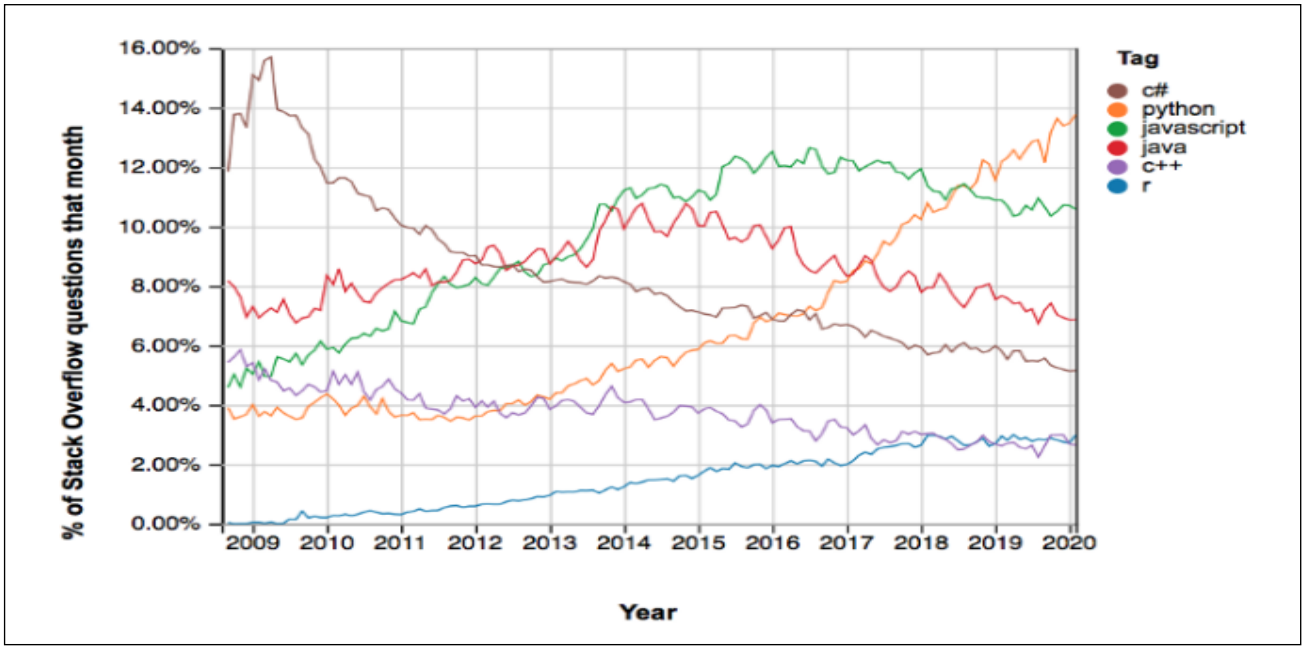
Jupyter Notebook састоји се од 3 компоненте:

1. интерактивна интернет апликација за интерактивно писање и покретање кода и креирање докумената
2. језга – процеси који покреће интерактивна интернет апликација која покреће кориснички код који може бити написан у једном од више програмских језика. Резултат који враћа језгро прослеђује се назад интерактивној интернет апликацији.
3. самостални документи који садрже приказ целокупног садржаја видљивог у интернет апликацији

Постоје разна окружења помоћу којих је могуће правити системе који су засновани на машинском учењу. Једно врло популарно окружење је *WEKA*. *WEKA* у себи садржи врло добре алате који су лаки за коришћење и који служе за претпроцесирање података,

имплементације неколико алгоритама машинског учења, неколико алата за визуелизацију и остало.

Језик који се користи у реализацији система је *Python*. *Python* је један од најпопуларнијих и најједноставнијих алата за аналитику података. Познат је као „швајцарски нож“ света програмирања, јер подржава структурирано програмирање, објектно-орјентисано програмирање, функционално програмирање и друго. Према неким истраживањима *Python* је најпопуларнији језик на планети и најпогоднији је за науку о подацима. Популарност програмског језика *Python* као и осталих програмских језика може се мерити на основу броја постављених питања о програмском језику у току месеца на популарној интернет заједници која се зове *Stack Overflow*, приказаној на слици 1.



Слика 1. Број постављених питања на *Stack Overflow* за различите програмске језике

Програмски језик *Python* дуго постоји па су инжињери имали довољно времена да обезбеде библиотеке за скоро сваку врсту проблема. Неколико популарнијих библиотека:

- *NumPy*
- *SciPy*
- *Pandas*
- *Scikit-Learn*
- *TensorFlow*
- *PyTorch*
- *Matplotlib*
- *Seaborn*

3. АЛГОРИТМИ НАДГЛЕДАНОГ МАШИНСКОГ УЧЕЊА

У овом поглављу биће детаљније описан концепт надгледаног учења као и три алгоритма из скупа алгоритама који постоје код надгледаног учења. Биће објашњено шта је то надгледано учење и који проблеми постоје код таквог учења, такође биће објашњени и алгоритми линеарне регресије, стабла одлучивања и к најближих суседа. Обзиром на комплексност машинског учењна уопште, па и надгледаног учења у овом поглављу највише пажње посвећује се концептима који су коришћени за реализацију свих фаза машинског учења.

3.1. Увод у алгоритме надгледаног машинског учења

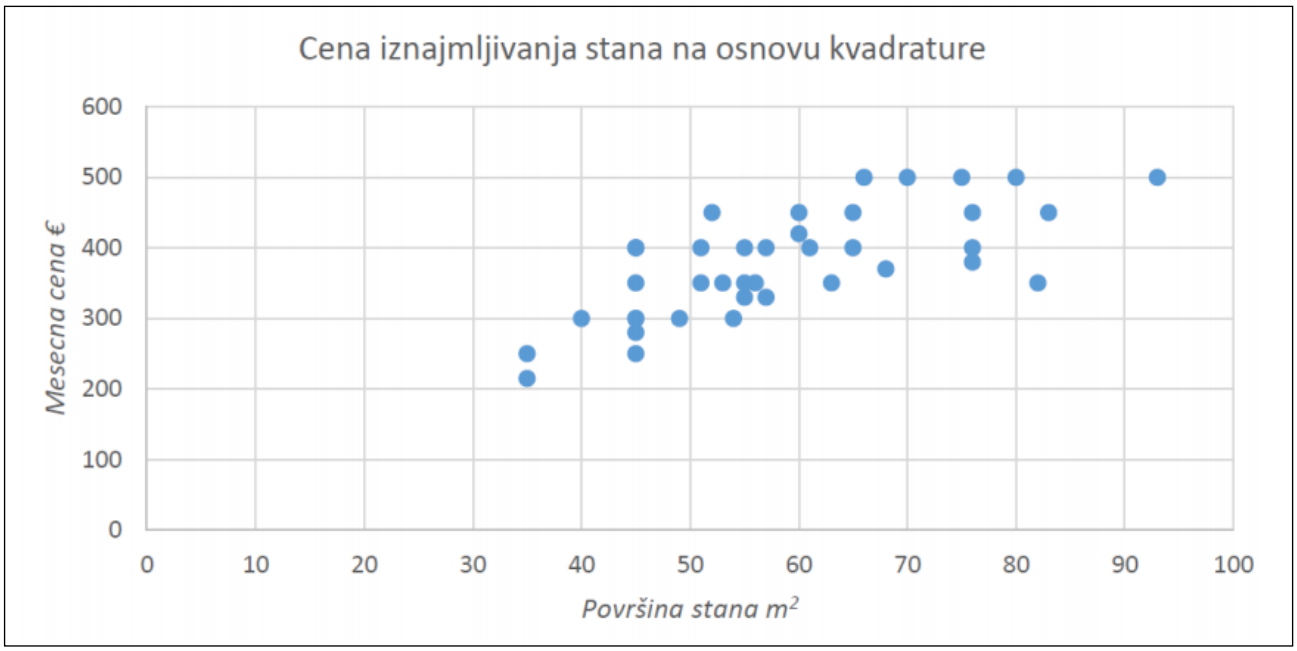
Надгледано учење је најчешће коришћена подгрупа машинског учења данас. Алгоритми надгледаног машинског учења засновани су на учењу на примерима. Надгледано учење представља облик машинског учења на основу обележеног скупа података за тренирање. Примерак из обележеног скупа података за тренирање састоји се од улазних вредности тј. улазних атрибута и жељене очекиване излазне вредности тј. циљног атрибута. Сврха алгоритама надгледаног машинског учења, који су тренирани на основу обележеног скупа података, јесте да израчунају излазну вредност за необележен скуп података тј. где не постоји вредност циљних атрибута тј. излазних вредности.

Циљни атрибут/излазна вредност може бити континуална вредност или дискретна/категоричка вредност. Уколико је излазна вредност контиунална тада то представља проблем регресије а ако је излазна вредност дискретна/категоричка онда то представља проблем класификације. Такође, излазна вредност може бити и структурирана вредност што представља проблем структурне предикције која неће бити разматрана у овом документу.

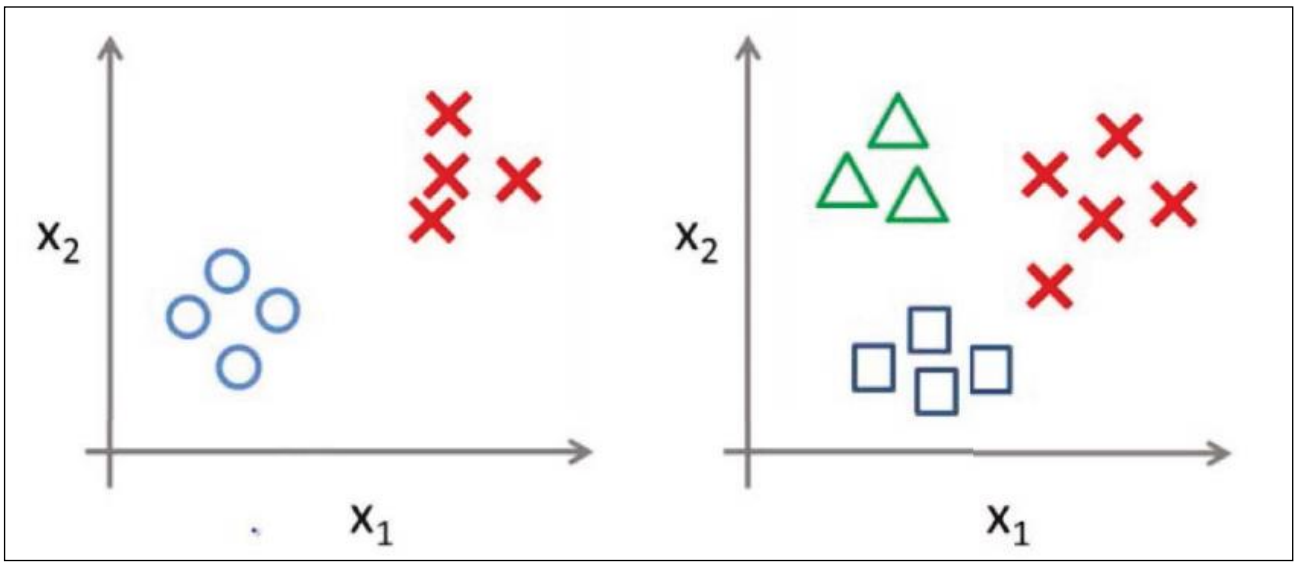
Пример проблема регресије може бити предвиђање месечних цена станова на основу површине стана. Цена стана је континуална вредност коју је потребно предвидети. У случају примера регресије улазни атрибут је површина стана док цена стана представља циљни атрибут тј излазну вредност. Пример обележеног скупа података за тренирање представљен је као дијаграм расипања месечне цене и површине стана приказан на слици 2.1., где свака тачка на графику представља примерак из обележеног скупа података за тренирање. Неки примери класификације су:

- медицинско тестирање ради утврђивања да ли пацијент има болест или не
- тестирање крвне групе које показује да ли пацијент има А, Б, АБ или О крвну групу
- одређивање да ли је нека електронска пошта спем или није

Медицинско тестирање ради утврђивања да ли пацијент има болест и одређивање да ли је електронска пошта спем представљају бинарну класификацију док тестирање крвне групе представља вишеструку класификацију. Примери бинарне класификације и вишеструке класификације приказани су на слици 2.2.



Слика 2.1. Пример обележеног скупа података за тренирање



Слика 2.2. Пример бинарне и вишеструке класификације

Неки од алгоритама надгледаног машинског учења су:

- наивни бајесовски класификатор
- линеарна регресија
- логистичка регресија
- метода потпорних вектора
- к најближих суседа
- стабла одлучивања

3.2. Линеарна регресија

Линеарна регресија може бити:

- једнострука – постоји само један улазни атрибут
- вишеструка – постоји више улазних атрибута
- универијантна – постоји само један циљни атрибут
- мултиваријантна – постоји више циљних атрибута

Линеарна регресија нам говори о повезаности улазног/улазних атрибута са циљним атрибутом/атрибутима. Та повезаност се представља математичким регресионим моделом. Постоје два типа модела:

1. детерминистички
2. стохастички

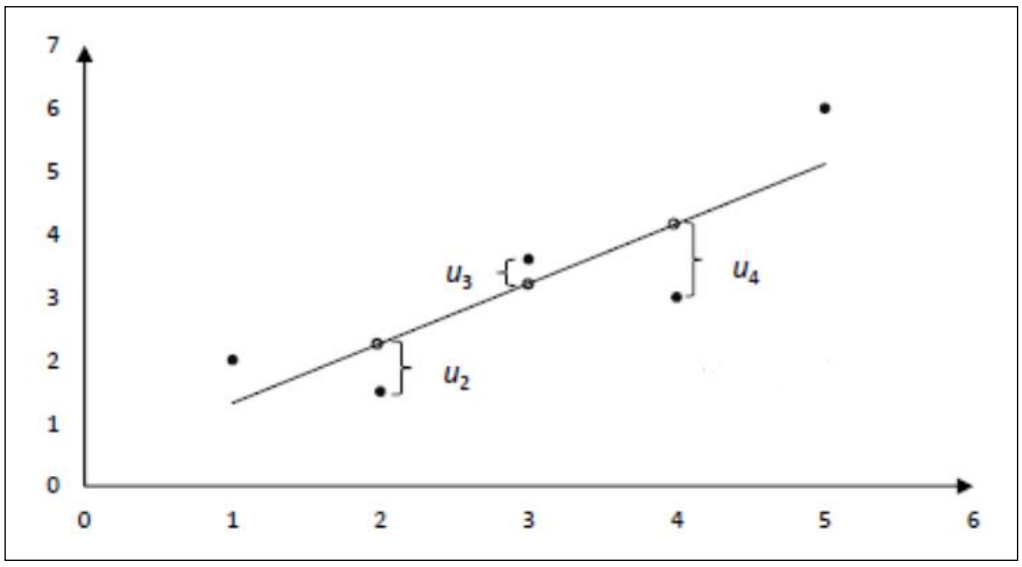
Запис детерминистичког модела може се датим следећим изразом:

$$y = a_0 + a_1 x$$

Модел представља једноставну линеарну регресиону праву. Променљива y представља циљни атрибут док променљива x представља улазни атрибут. Овај модел описује тачну зависност улазног атрибута и циљног атрибута, односно доказује како је циљни атрибут одређен тачно једним улазним атрибутом. Запис стохастичког модела дат је следећим изразом:

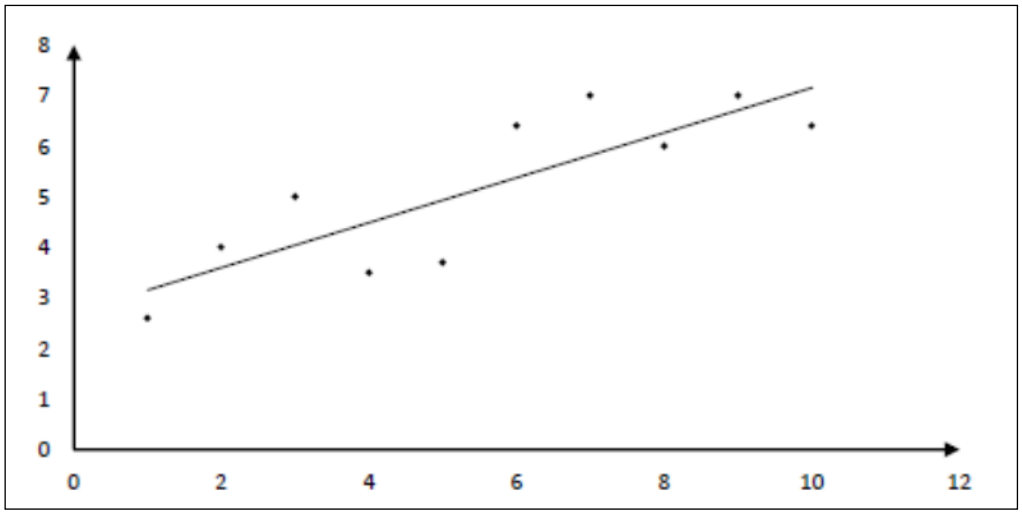
$$y = a_0 + a_1 x + \mu$$

Код стохастичког модела постоји додатна променљива, која фигурише у изразу, μ која представља мерну несигурност. Та мерна несигурност приказује разлику између експериментално измерених или утврђених вредности променљиве величине и теоријског предвиђања вредности те променљиве величине. Како би се лакше разумела, мерна несигурност приказана је на слици 2.3.



Слика 2.3. Мерне несигурности

Применом метода најмањих квадрата добија се најмањи утицај мерне несигурности на циљни атрибут. Метода најмањих квадрата једна је од најважнијих метода за обраду експериментално добијених резултата. Регресионим једначинама, регресионом анализом, настоји се да се у дијаграм расипања резултата уцрта оптимална права која ће најбоље описати однос улазног атрибута и циљног атрибута. Дијаграм расипања са оптималном правом приказан је на слици 2.4.



Слика 2.4. Дијаграм расипања са регресионом правом

Процес проналажења регресионе праве заправо представља обучавање модела. У пракси се за обучавање модела најчешће користи метод градијентног или стохастичког спуста. То је итеративни метод тражења којим се добија најмањи утицај мерне несигурности на циљну променљиву. За реализацију алгоритма линеарне регресије касније у документу, коришћена је *Python* библиотека *Scikit-Learn* која има имплементиран алгоритам линеане регресије заснован на методи најмањих квадрата.

3.3. Стабло одлучивања

Стабло одлучивања је непараметризована метода надгледаног машинског учења које се може користи и у проблемима регресије и у проблемима класификације. Циљ стабла одлучивања јесте да се конструише модел/стабло који ће на основу неколико улазних атрибута предвидети циљни атрибут. Стабло се састоји од чворова и грана где сваки чвор који није лист стабла представља један улазни атрибут а гране чвора представљају неку информацију о вредности улазног атрибута. Стабло одлучивања се користи као класификатор тако што узима улазне атрибуте од улазног примерка на основу којих треба да класификује улазни примерак.

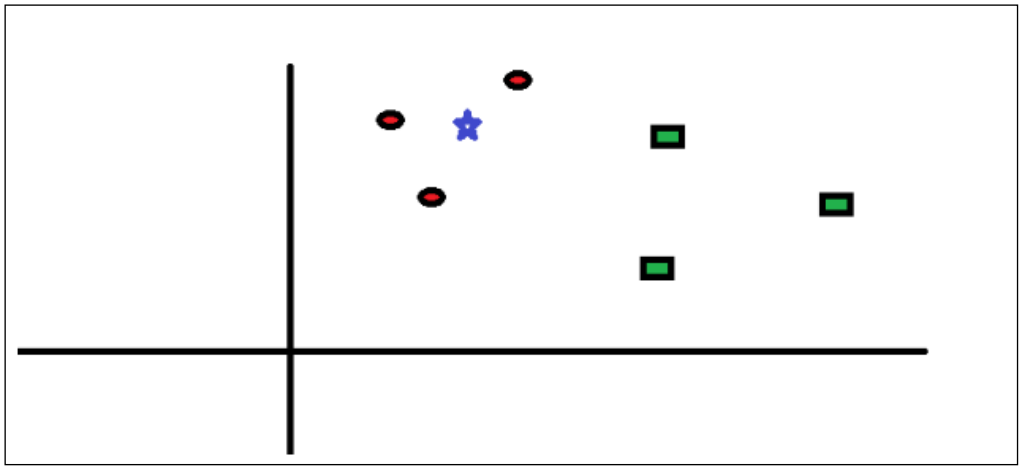
1. атрибут који је корени чвор стабла интерпретира се као питање, а одговор се одређује на основу вредности тог атрибута из улазног примерка
2. одговор одређује који чвор наследник се посећује
3. понављање све док се не дође до листа стабла

Стабла се формирају од корена ка листовима, тзв. „похлепним“ приступом тј. рекурзивним бинарним дељењем. Алгоритми за конструисање стабла одлучивања засновани на ентропији тј. теорији информација су:

- *ID3*
- *C5.0*
- *CART*

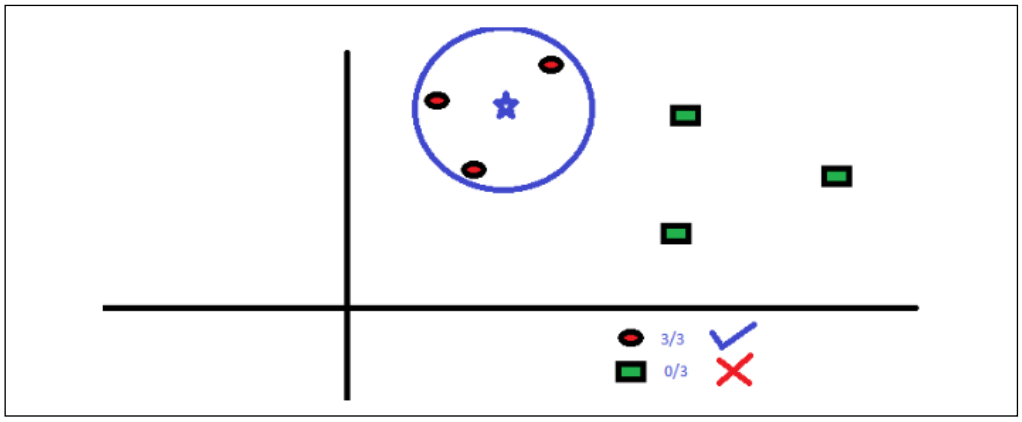
3.4. К најближих суседа

К најближих суседа је један од најлакших алгоритама који се користи и за проблем класификације и за проблем регресије. Да би се разумео алгоритам, наводи се пример на слици 2.5. где су приказани зелени квадратићи и црвени кругови. Потребно је пронаћи класу којој припада плава звездица са слике.



Слика 2.5. Дијаграм расипања црвених кругова и зелених квадратића

Фактор „к“ у алгоритму говори колики број најближих суседа плавој звездици се посматра да би се утврдило којој класи плава звездица припада. Ако се узме да је к = 3, тада би ситуација била као на слици 2.6.



Слика 2.6. Дијаграм расипања црвених кругова и зелених квадратића када је к = 3

Број најближих суседа који припадају класи црвених кругова је три док је број оних најближих суседа који припадају класи зелених кругова је нула. На основу претходног

разматрања закључује се да постоји велика шанса да плава звездица припада класи црвених кругова.

Може се закључити да промена вредности фактора к утиче на квалитет предикције с тога је битно изабрати вредност к тако да алгоритам да што квалитетнију предикцију. Процес избора праве вредности за фактор к назива се подешавање параметара. Што боље одредимо вредност фактора к, прецизност модела ће бити боља. Фактор к може да се изабере на следеће начине:

- помоћу „лакат“ методе тј. минимизацијом грешке над валидационим скупом
- к = $\sqrt{m}$ где је *m* укупан број свих примерака у скупу података

Фактор к се бира углавом да буде непарна вредност, да би се избегла конфузија у ситуацији када постоје две класе и где је број најближих суседа из обе класе међусобно исти. Што је већа вредност фактора к, смањује се вероватноћа за грешку. За проналажење најближих суседа користе се следеће метрике:

- еуклидска раздаљина
- менхетн раздаљина
- чеобишева раздаљина
- махаланобисова раздаљина

Еуклидско растојање представља растојање између тачке $A(x_1, y_1)$ и тачке $B(x_2, y_2)$ у дводимензионалном простору које се рачуна као:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

4. РЕАЛИЗАЦИЈА СИСТЕМА

У овом поглављу кратко се описује на који начин могу да се реализују одређени системи. Реализација система представља на који начин су реализовани процеси машинског учења који су примењени на реалне проблеме. Биће реализована два система заснована на машинском учењу:

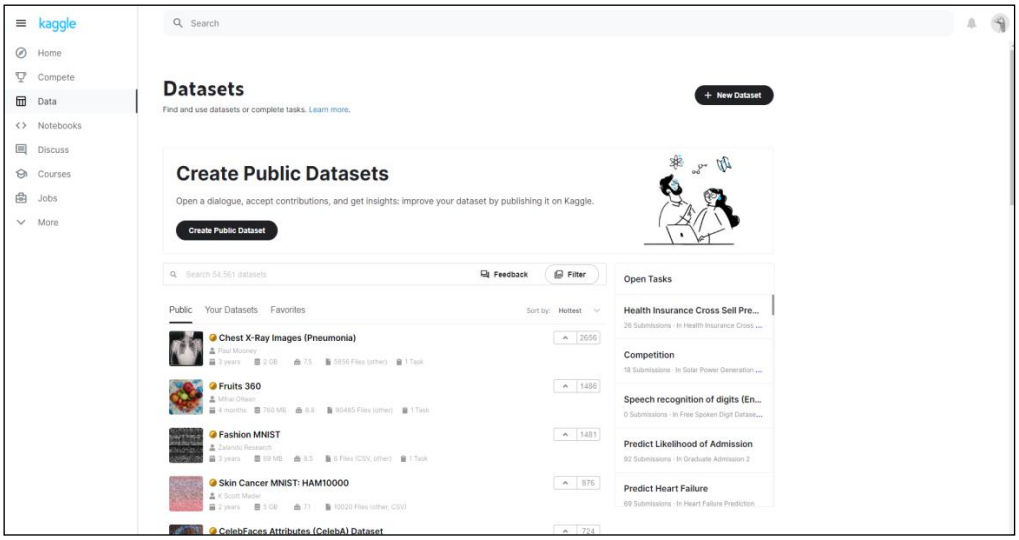
1. систем за помоћ при стратешком пословном одлучивању
2. систем за помоћ при одлучивању да ли пацијент има кардиоваскуларне проблеме

Процес машинског учења подељен је у шест фаза:

1. прикупљање података
2. припрема података
3. експлоративна анализа података
4. модел
5. обучавање модела
6. евалуација

4.1. Прикупљање података

Прикупљање података је битна фаза у процесу машинског учења. Од квалитета прикупљених података зависи ефикасност предиктивног модела и квалитет анализе података. Подаци који се користе у реализацији система преузети су са интернет заједнице људи који се баве науком о подацима и машинским учењем, која се назива *Kaggle*.



Слика 3.1. *Kaggle*

За потребе првог система подаци који су прикупљени изгледају као на слици 3.2.

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

Слика 3.2. Подаци о регистрованим корисницима

Прикупљени подаци састоје се од следећих атрибута:

- *Email* – електронска адреса корисника
- *Address* – физичка адреса корисника
- *Avatar* – Боја аватара корисника
- *Avg. Session Length* – просечно време изражено у часовима, које је корисник провео консултујући се са представницима компаније
- *Time on App* – просечно време изражено у часовима, које је корисник провео користећи мобилну апликацију
- *Time on Website* – просечно време изражено у часовима, које је корисник провео користећи интернет апликацију
- *Length of Membership* – колико дуго постоји корисник, изражено у годинама
- *Yearly Amount Spent* – просечна годишња потрошња корисника, изражена у доларима

За потребе другог система подаци који су прикупљени изгледају као на слици 3.3.

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

Слика 3.3. Подаци о пацијентима

Прикупљени подаци састоје се од следећих атрибута:

- *id* – идентификатор
- *age* – старост изражена у данима
- *gender* – пол, категорички атрибут 1=мушко 2=женско
- *height* – висина изражена у центиметрима
- *weight* – тежина изражена у килограмима
- *ap_hi* – систолни/горњи притисак
- *ap_lo* – дијастолни/доњи притисак
- *cholesterol* – ниво холестерола, категорички атрибут где је 1=нормалан; 2=изнад нормале; 3=доста изнад нормале
- *gluc* – ниво глукозе, категорички атрибут где је 1=нормалан; 2=изнад нормале; 3=доста изнад нормале

- *smoke* – да ли пацијент конзумира цигарете, категорички атрибут где је 0=не; 1=да
- *alco* – да ли пацијент конзумира алкохол, категорички атрибут где је 0=не; 1=да
- *active* – да ли је пацијент физички активан, категорички атрибут где је 0=не; 1=да
- *cardio* – да ли пацијент има кардиоваскуларних проблема, категорички атрибут где је 0=не; 1=да

4.2. Припрема података

Припрема података подразумева процес „чишћења“ података. Такав процес је често дуготрајан али и пресудан. Битни задаци у таквом процесу су:

- уклањање сувишних података и одступања
- попуњавање недостајућих вредности
- прилагођавање података
- трансформација података

Подаци који се припремају, обрађују се помоћу *Python* библиотеке *Pandas*.

4.3. Експлораторна анализа података

Експлораторна анализа података ће бити спроведена помоћу визуелизације података. Биће посматрани униваријантни, биваријантни и мултиваријантни графици на основу којих ће се долазити од одређених закључака како би се што боље разумели подаци који су прикупљени. За визуелизацију података користе се пајтонове *Matplotlib* и *Seaborn* библиотеке.

4.4. Модел

Због броја алгоритама који постоје у машинском учењу, избор алгоритма може да буде захтеван посао. Изабрани алгоритам за први ситем је линеарна регресија док се у другом систему примењују алгоритми стабла одлучивања и к најближих суседа. За реализацију ове фазе користи се *Python* библиотека *Scikit-Learn*.

4.5. Обучавање модела

Обучавање модела биће засновано на тренинг подацима.За реализацију ове фазе користи се *Python* библиотека *Scikit-Learn* која има имплементиране разне алгоритме машинског учења. Обучавање се обавља коришћењем методе *fit(..)* која као резултат враћа истренирани модел.

4.6. Евалуација

Евалуација модела подразумева процес тестирања модела помоћу тест података на основу чега се помоћу одређене метрике одређује тачност предикције модела који се потом упоређује са тачностима других алгоритама који могу да се примене на посматрани проблем. На основу упоређивања тачности модела се бира модел који има највећу тачност.

5. РЕЗУЛТАТИ РАДА И ДИСКУСИЈА

У овом поглављу детаљно је приказан и описан начин реализације свих фаза у процесу примене техника машинског учења на два различита проблема. Детаљно ће бити представљено на који начин су реализовани систем за помоћ при одлучивању да ли пацијент има кардиоваскуларне проблеме и систем за помоћ при стратешком пословном одлучивању, коришћењем програмског језика *Python* у окружењу *Jupyter Notebook*. Детаљно се пролази кроз све фазе процеса машинског учења за сваки од система:

- прикупљање података
- припрема података
- експлораторна анализа података
- модел
- обучавање модела
- евалуација

5.1. Систем за помоћ при одлучивању да ли пацијент има кардиоваскуларне проблеме

Систем треба да предвиди на основу стања (улазних атрибута) пацијента да ли пацијент можда има кардиоваскуларне проблеме. Прво наводимо библиотеке које ће нам бити потребне за почетак.

Biblioteke

```
In [44]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

Слика 4.1. *Python* библиотеке.

Потом је потребно учитати скуп података из датотеке и приказати основне информације о учитаним подацима као на слици 4.2., име датотеке из које се читају подаци о пацијентима је „*cardio_train3.csv*“:

```
In [4]: patients = pd.read_csv('cardio_train3.csv')

In [6]: patients.head()
Out[6]:
```

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

```
In [449]: patients.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           70000 non-null  int64
 1   age          70000 non-null  int64
 2   gender       70000 non-null  int64
 3   height       70000 non-null  int64
 4   weight       70000 non-null  float64
 5   ap_hi        70000 non-null  int64
 6   ap_lo        70000 non-null  int64
 7   cholesterol  70000 non-null  int64
 8   gluc         70000 non-null  int64
 9   smoke        70000 non-null  int64
 10  alco         70000 non-null  int64
 11  active       70000 non-null  int64
 12  cardio       70000 non-null  int64
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

Слика 4.2. Информације о учитаним подацима

У скупу података постоји 70000 примерака/пацијената, укупно 13 атрибута/колона и 700 ненултих вредности у свакој колони.

Након учитавања и приказивања основних информација из скупа података следи фаза „припрема података“. Први корак је провера да ли има непостојећих података, дупликата или великих одступања у скупу података. Слика 4.3. представља проверу да ли има непостојећих вредности и дупликата док слика 4.4. представља опис података који се налазе у скупу на основу којих се закључује да ли постоје одступања. Опис података подразумева неке карактеристике као што су просечна вредност за сваки атрибут, број примерака за сваки атрибут, минималне и максималне вредности атрибута итд.

```
In [450]: # Da li postoje Null vrednosti
          print(patients.isnull().sum(), '\n')
          print('Postoji bar jedna Null vrednost:',patients.isnull().values.any(), '\n')
          # Ukupan broj Null vrednosti u Dataset-u
          print('Ukupan broj Null vrednosti:', patients.isnull().sum().sum())

id             0
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64

Postoji bar jedna Null vrednost: False

Ukupan broj Null vrednosti: 0

In [451]: print("Broj duplikata: {}".format(patients.duplicated().sum()))

Broj duplikata: 0
```

Слика 4.3. Провера да ли има непостојећих вредности и дупликата у скупу података

```
In [452]: patients.describe()
```

Out[452]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	49972.419900	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.053771
std	28851.302323	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.225568
min	0.000000	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000
25%	25006.750000	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000
50%	50001.500000	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000
75%	74889.250000	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000
max	99999.000000	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000

Слика 4.4. Опис података

На основу слике 4.3. види се да нема непостојећих вредности и да нема дупликата, подаци су „чисти“. Након разматрања слике 4.4. долази се до закључка да колоне *ap_hi* и *ap_lo* имају примерке који доста одступају. Емпиријски се показало да је вероватноћа да се појави пацијент са систолним/горњим крвним притиском изнад 200 врло мала јер то представља изузетно тешку хипертензију, тако је и са дијастолним/доњим притиском који углавном не иде испод 30. У опису података са слике 3.5. примећује се следеће:

- минималне вредности у колонама *ap_lo* и *ap_hi* износе -70 и -150 респективно
- максималне вредности у колонама *ap_lo* и *ap_hi* износе 11000 и 16020 респективно

На основу претходних закључака може се рећи да у скупу података постоје примерци/пацијенти који одступају и које је потребно уклонити или модификовати. У овом случају због количине података која је велика, примерци који одступају биће уклоњени. Прво се постављају „границе“ за колоне *ap_lo* и *ap_hi* који служе као филтер да се узму у обзир само они примерци који имају вредности колона *ap_lo* и *ap_hi* у оквирима „граница“. Потом је потребно елиминисати примерке који не задовољавају те „границе“ као на слици 4.5.

In [9]:
```
bounds = pd.DataFrame(index=['donja granica', 'gornja granica'])
bounds['ap_hi'] = [80, 200]
bounds['ap_lo'] = [30, 120]
bounds
```

Out[9]:

	ap_hi	ap_lo
donja granica	80	30
gornja granica	200	120

In [10]:
```
ap_hi_filter0 = (patients["ap_hi"] > bounds["ap_hi"][1])
ap_hi_filter1 = (patients["ap_hi"] < bounds["ap_hi"][0])

ap_lo_filter0 = (patients["ap_lo"] > bounds["ap_lo"][1])
ap_lo_filter1 = (patients["ap_lo"] < bounds["ap_lo"][0])

outliers = (ap_hi_filter0 | ap_lo_filter0 | ap_hi_filter1 | ap_lo_filter1)
print("Postoji {} primeraka koji odstupaju\n".format(patients[outliers]["cardio"].count()))
# Eliminacija primeraka koji odstupaju
patients = patients[~outliers]
patients.info()
```

```
Postoji 1375 primeraka koji odstupaju

<class 'pandas.core.frame.DataFrame'>
Int64Index: 68625 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           68625 non-null  int64
 1   age          68625 non-null  int64
 2   gender       68625 non-null  int64
 3   height       68625 non-null  int64
 4   weight       68625 non-null  float64
 5   ap_hi        68625 non-null  int64
 6   ap_lo        68625 non-null  int64
 7   cholesterol  68625 non-null  int64
 8   gluc         68625 non-null  int64
 9   smoke        68625 non-null  int64
 10  alco         68625 non-null  int64
 11  active       68625 non-null  int64
 12  cardio       68625 non-null  int64
dtypes: float64(1), int64(12)
```

Слика 4.5. Елиминација примерака који одступају

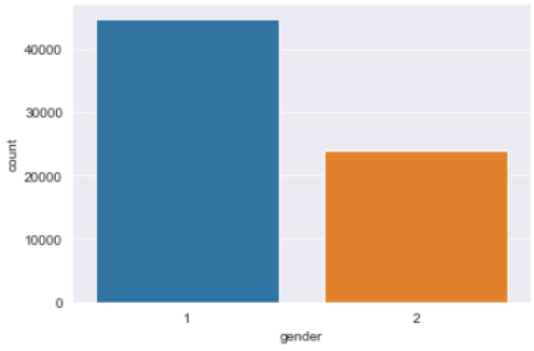
Експлораторна анализа података за овакав проблем састојаће се из 3 фазе:

1. униваријантна анализа података
2. биваријантна анализа података
3. мултиваријантна анализа података

Циљ експлораторне анализе је да се стекне што бољи увид у податке са којима се ради.

Униваријантна анализа података представља анализу података која атрибуте разматра појединачно. Први атрибут који гледамо је *gender* тј. пол пацијента. Пошто је категорички атрибут у питању најпогодније је користити стубичасти графиком на ком ће бити приказан број примерака који припада свакој од категорија атрибута *gender*, слика 4.6.:

```
In [13]: #sns.set_theme(style="darkgrid")
         sns.set_style(style='darkgrid')
         sns.countplot(x='gender', data=patients)

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1c8e2fdf220>
```

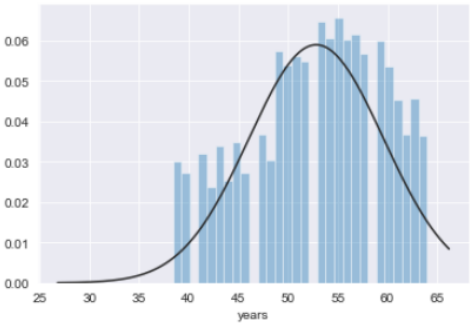


Слика 4.6. Број примерака који припадају женском(1) и мушком(2) полу

Може се приметити да се у скупу података налази око 60% женских пацијената на основу чега би могло да се каже да ће овај систем говорити више о кардиоваскуларним проблемима код жена. Ово закључивање може бити подложно променама у складу са корелацијом између атрибута *cardio* и *gender*.

Други атрибут који посматрамо је атрибут *age* тј. старост за који ће бити посматрана расподела представљена хистограмом на графику, као на слици 4.7.

```
In [18]: from scipy.stats import norm
         years = patients['age'] /365
         patients['years'] = years.astype(int)
         sns.distplot(patients['years'], bins=41, fit=norm, kde=False)

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1c8e412ae80>
```

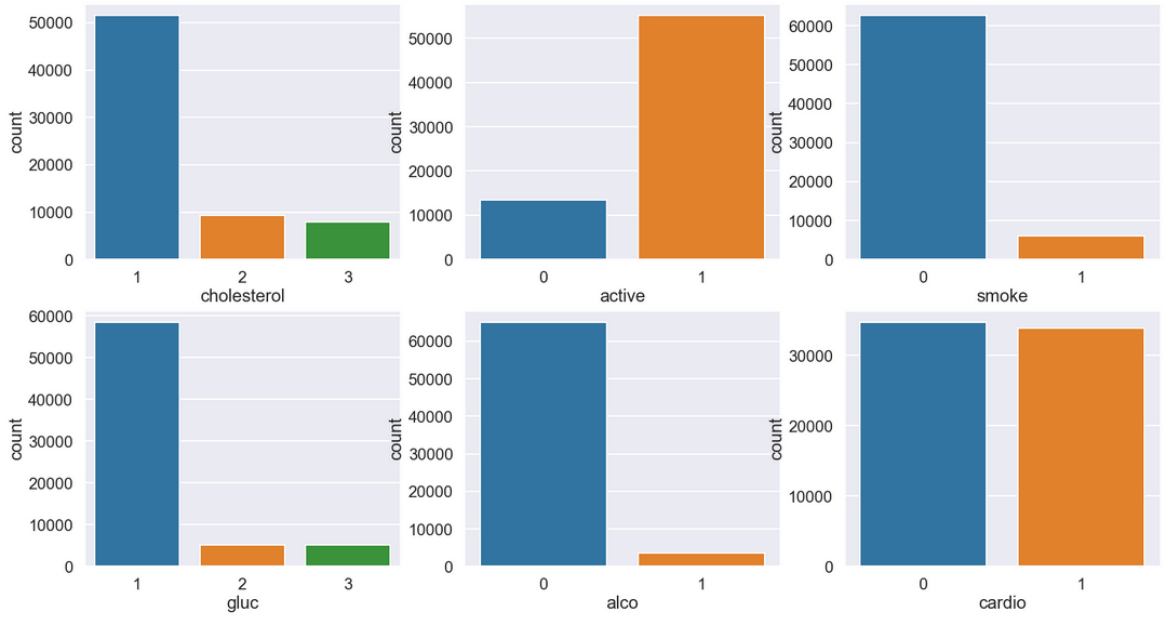


Слика 4.7. Расподела примерака по годинама

Може се видети да је расподела нормална као и што се очекује да буде. Такође, године свих пацијената из скупа података не прелазе границе испод 40 и изнад 65 година на основу чега се може закључити да модел система можда неће давати тако добру предикцију за пацијенте који по годинама не припадају опсегу (40, 65).

Сада се разматрају сви остали категорички атрибути по истом принципу као што је то рађено за атрибут *gender*. На слици 4.8. приказани су стубичасти графици који приказују број примерака за сваку од категорија посматраног атрибута.

```
In [28]: atts = ['cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio']
         sns.set_context("poster", font_scale = 1)
         fig, ax =plt.subplots(2,3,figsize=(22,12))
         fig.tight_layout()
         sns.countplot(x='cholesterol', ax=ax[0,0], data=patients)
         sns.countplot(x='active', ax=ax[0,1], data=patients)
         sns.countplot(x='smoke', ax=ax[0,2], data=patients)
         sns.countplot(x='gluc', ax=ax[1,0], data=patients)
         sns.countplot(x='alco', ax=ax[1,1], data=patients)
         sns.countplot(x='cardio', ax=ax[1,2], data=patients)

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1c8e6766f10>
```

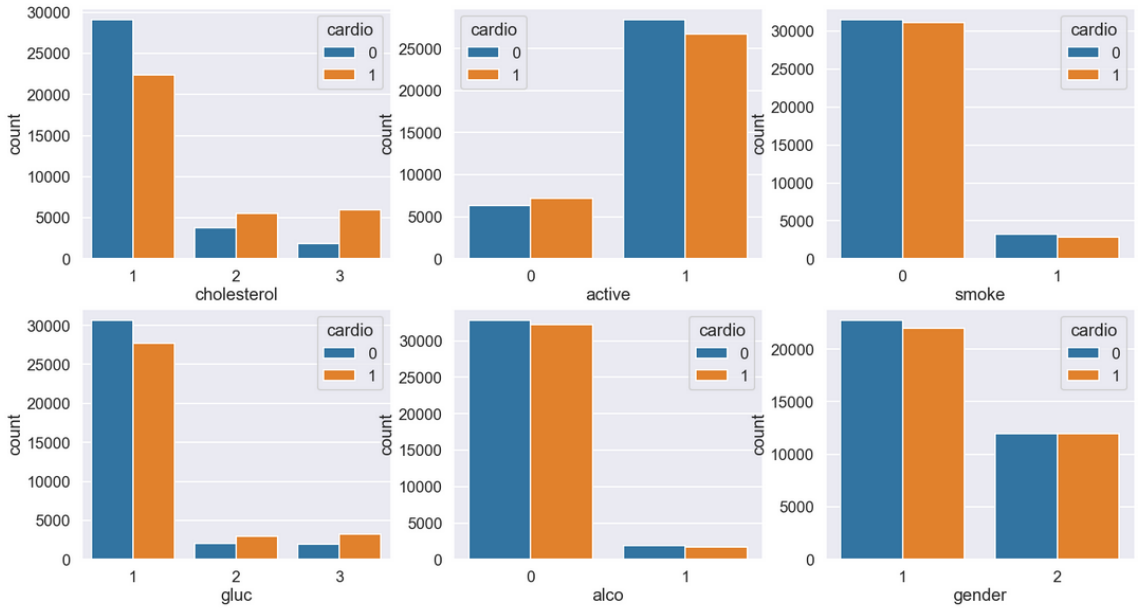


Слика 4.8. Сви категорички атрибути

На основу слике 4.8. потребно је доћи до некаквог закључка гледајући број примерака на основу категорија одређених атрибута као што су холестерол, ниво глукозе и информације о активностима и навикама пушења и конзумирања алкохола пацијената. Може се закључити да већина пацијената има нормалан ниво холестерола и глукозе и мање пушача и оних који конзумирају алкохол. Такође је очигледно да је већина пацијената физички активна и да је број пацијената који имају и немају кардиоваскуларне проблеме поприлично уравнотежен.

Биваријантна анализа података представља анализу података која разматра два атрибута. Циљни атрибут у овом систему је атрибут *cardio* тј. да ли пацијент има кардиоваскуларних проблема па ће ова анализа посматрати само циљни атрибут у односу на остале (улазне) атрибуте. На слици 4.9. приказани су графици који говоре колики број пацијената има кардиоваскуларне проблеме, за сваку од категорија посматраног атрибута пацијента.

```
In [29]: atts = ['cholesterol', 'gluc', 'smoke', 'alco', 'active', 'gender']
         sns.set_context("poster", font_scale = 1)
         fig, ax =plt.subplots(2,3,figsize=(22,12))
         fig.tight_layout()
         sns.countplot(x='cholesterol', ax=ax[0,0], data=patients, hue='cardio')
         sns.countplot(x='active', ax=ax[0,1], data=patients, hue='cardio')
         sns.countplot(x='smoke', ax=ax[0,2], data=patients, hue='cardio')
         sns.countplot(x='gluc', ax=ax[1,0], data=patients, hue='cardio')
         sns.countplot(x='alco', ax=ax[1,1], data=patients, hue='cardio')
         sns.countplot(x='gender', ax=ax[1,2], data=patients, hue='cardio')

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1c8e67cc040>
```



Слика 4.9. Сви категорички атрибути са груписањем по атрибуту *cardio*

Може се приметити да је доста пацијената који имају кардиоваскуларне проблеме активно, нормалног нивоа холестерола и глукозе, непушачи и не конзумирају алкохол, што је врло изненађујуће и лоше по квалитет предиктивног модела. У оваквим ситуацијама долази до изражаја важност фазе прикупљања података. Од квалитета прикупљених података доста зависи ефикасност предиктивног модела.

На слици 4.10. су приказане две расподеле пацијената по годинама. Расподела обележена плавом бојом представља пацијенте који немају кардиоваскуларне проблеме док расподела обележена наранџастом бојом представља пацијенте који имају кардиоваскуларне проблеме. Види се да је старосна граница пацијената који имају кардиоваскуларне проблеме већа од старосне границе пацијената који немају кардиоваскуларне проблеме што је и очекивано.

Мултиваријантна анализа података представља анализу података која разматра три или више атрибута. Први пример мултиваријантне анализе биће посматрање три атрибута пацијената. Посматрају се пацијенти који имају кардиоваскуларне проблеме у односу на године при томе да су пацијенти груписани по вредности категоричког атрибута *cholesterol*, слика 4.11. на којој је приказан дијаграм расипања атрибута *age* и *cholesterol* који су груписани по вредности атрибута *cardio*.

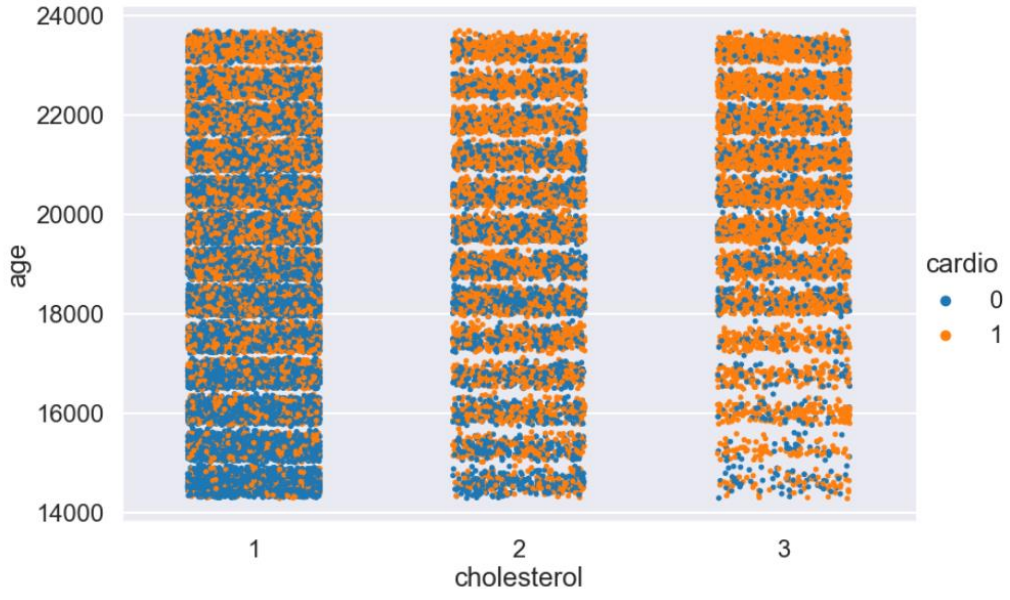
Плаве тачкице на слици представљају пацијенте који немају кардиоваскуларне проблеме док наранџасте тачкице представљају пацијенте који имају кардиоваскуларне проблеме. Са слике је лако закључити да што више година пацијент има већа је вероватноћа

да има кардиоваскуларне проблеме као и да што већи ниво холестерола пацијент има већа је вероватноћа да пацијент има кардиоваскуларне проблеме.

```
In [464]: pat_cardio0 = (patients['cardio'] == 0)
          pat_cardio1 = (patients['cardio'] == 1)

          pat_years_cardio0 = patients[pat_cardio0]
          pat_years_cardio1 = patients[pat_cardio1]

In [465]: f, ax = plt.subplots(figsize = (10,7))
          sns.distplot(pat_years_cardio0['years'], hist=False) #plava linija na grafiku
          sns.distplot(pat_years_cardio1['years'], hist=False) #crvena linija na grafiku

Out[465]: <matplotlib.axes._subplots.AxesSubplot at 0x13804d64b80>
```

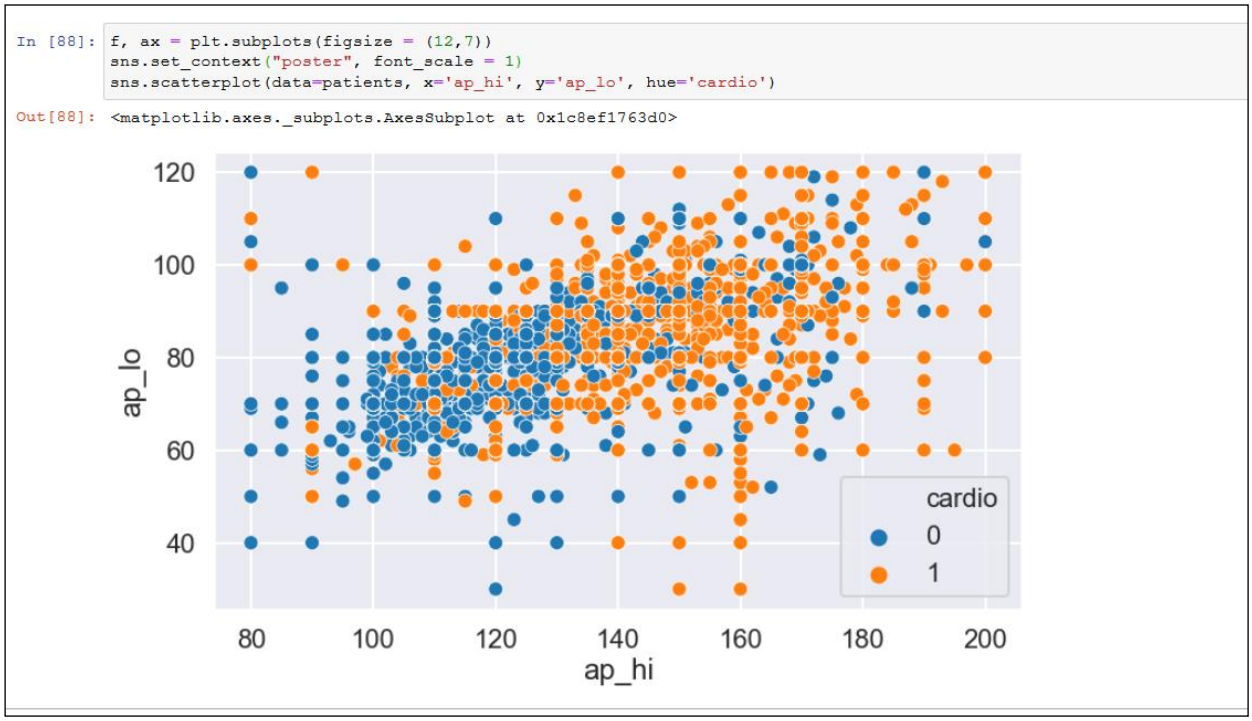
Слика 4.10. Две расподеле по годинама пацијената

```
In [87]: sns.set_context("poster", font_scale = 1)
         sns.catplot(x='cholesterol', y='age', hue='cardio', data=patients, jitter='0.25', height=8, aspect=1.5)
         #sns.stripplot(x="cholesterol", y="years", data=patients, jitter=0.55)

Out[87]: <seaborn.axisgrid.FacetGrid at 0x1c8ed8279a0>
```



Слика 4.11. Дијаграм расипања пацијената по годинама, нивоу холестерола и да ли има срчаних проблема

По сличном принципу представљена је мултиваријантна анализа података за атрибуте *ap_lo* , *ap_hi* и *cardio*, на слици 4.12., лако се примећује да што већи притисак пацијент има, веће су шансе да има кардиоваскуларне проблеме.



Слика 4.12. Дијаграм расипања пацијената по горњем и доњем притиску и да ли има срчаних проблема

Након експлоративне анализе података прелази се на избор модела, раздвајање атрибута пацијената по циљном атрибуту и по улазним атрибутима, раздвајање скупа података на тренинг и тест податке који се користе у обучавању и евалуацији модела респективно, нормализација вредности свих атрибута као и само обучавање модела. У овом проблему алгоритми који ће бити коришћени су:

- стабла одлучивања
- к најближих суседа

Циљни атрибут је *cardio* док остале колоне представљају улазне атрибуте а скуп података подељен је тако да је 30% од укупног броја података узето за тестирање док 70% чине подаци за тренинг тј. обучавање модела. Подела података постигнута је коришћењем библиотеке *Scikit-Learn* која пружа врло лак начин за поделу скупа. За реализацију процеса обучавања модела такође је коришћена библиотека *Scikit-Learn.*

На слици 4.13. приказана је реализација поделе скупа података, избора алгоритма и обучавање модела.

```
In [469]: y = patients['cardio']

In [470]: X = patients[['age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo',
                        'cholesterol', 'gluc', 'smoke', 'alco', 'active']]

In [471]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

In [472]: #normalizacija atributa
          from sklearn.preprocessing import normalize
          X_train = normalize(X_train)
          X_test = normalize(X_test)

In [473]: from sklearn.tree import DecisionTreeClassifier
          from sklearn.neighbors import KNeighborsClassifier

In [474]: dtree = DecisionTreeClassifier(criterion='entropy')
          knn = KNeighborsClassifier(n_neighbors=100)

In [475]: modeli = {"Stabla odlucivanja" : dtree,
                    "KNN" : knn}
          scores = {}

In [495]: dtree.fit(X_train, y_train)
          knn.fit(X_train, y_train)
```

Слика 4.13. Реализација фаза модела и обучавање модела

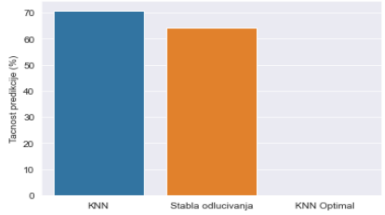
Променљиве *dtree* и *knn* представљају класификаторе тј. моделе који су инстанцирани позивањем конструктора *DecisionTreeClassifier(criterion='entropy')* и *KNeighborsClassifier(n_neighbors=100)* из библиотеке *Scikit-Learn*. *dtree* је модел заснован на стаблу одлучивања које користи ентропију као критеријум за одлуку. *knn* је модел заснован на алгоритму к најближих суседа где је вредност фактора к узета произвољно.

Након обучавања модела потребно је одрадити предикцију и евалуацију која служи као увид у ефикасност предиктивних модела. Евалуација се ради на основу тест података. Тест подаци су скуп примерака на основу којих модел после обучавања покушава да предвиди класу којој припада сваки од тих примерака, након чега се те предвиђене класе упоређују са стварним класама којима примерак припада. Постоји више метода упоређивања стварних и предвиђених вредности. Метода модела *score(...)* која прима два аргумента: тест податке и њихове стварне вредности циљног атрибута, врши предвиђање тест података и упоређињање предвиђене и стварне вредности циљног атрибута где је упоређивање је засновано на статистичкој мери која се назива R^2. Резултат методе је број у интервалу (0, 1) који говори колико је процентуално модел дао тачну предикцију, слика 4.14.

```
In [496]: scores['Stabla odlucivanja'] = dtree.score(X_test, y_test)*100
          scores['KNN'] = knn.score(X_test, y_test)*100
          scores['KNN Optimal'] = 0

          scores_table= pd.DataFrame(scores, index=["Tacnost predikcije (%)"]).T
          scores_table.sort_values(by=["Tacnost predikcije (%)"], axis=0 ,ascending=False, inplace=True)
          sns.barplot(x=scores_table.index,y=scores_table["Tacnost predikcije (%)"])
          scores_table
```

Out[496]:

	Tacnost predikcije (%)
KNN	70.802529
Stabla odlucivanja	64.153696
KNN Optimal	0.000000



Слика 4.14. Тачност предикције алгоритама

Уочава се да алгоритам к најближих суседа даје боље резултате при предвиђању са процентуалном тачношћу од 70,8% док стабло одлучивања даје тачну предикцију у 64,15% случајева. Проценат предикција није лош, али није ни превише добар, што је последица лоше прикупљених података који имају нешто већу неодређеност него што би требало.

Како би к најближих суседа било што ефикасније потребно је пронаћи одговарајућу вредност фактора к неким методом. Метод који је овде приказан назива се „*лакат*“ метод који итеративно за различите вредности фактора к обучава модел који потом рачуна проценат грешке на основу предвиђања вредности циљног атрибута за тест податке који се упоређују са стварним вредностима циљног атрибута из тест података. Када се итерације заврше на основу графика зависности фактора к и процента грешке могуће је наћи такво к да је проценат грешке при предвиђању најмања. „Лакат“ алгоритам приказан на слици 4.15. итерира за све вредности фактора к између 1 и 200 док за овај проблем вредност фактора к иде од 1 до 350. Због брзине извршавања кода „лакат“ алгоритма, опсег у ком итерира фактор к је подељен на опсеге од 50 до 100 итерација.

```
In [485]: error_rate = []

          for i in range(1,200):

              knn = KNeighborsClassifier(n_neighbors=i)
              knn.fit(X_train,y_train)
              pred_i = knn.predict(X_test)
              error_rate.append(np.mean(pred_i != y_test))
```

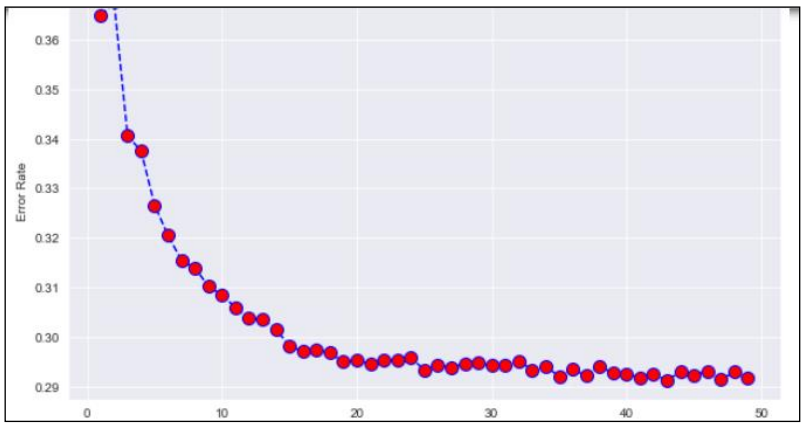
Слика 4.15. „Лакат“ алгоритам

Код за приказ зависности фактора к и процента грешке приказан је на слици 4.16.

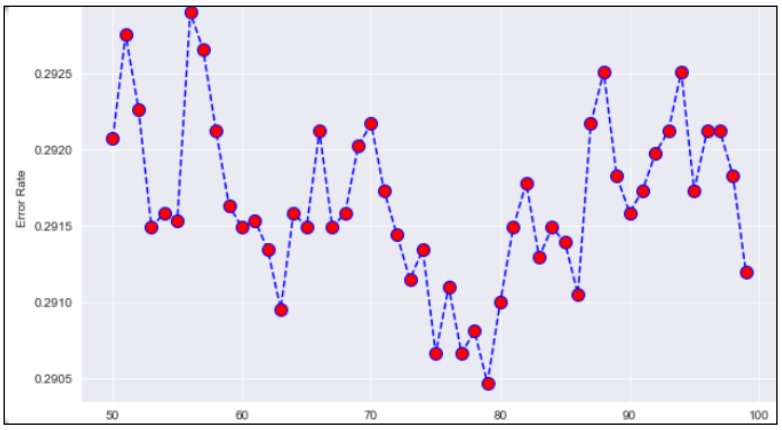
```
In [492]: plt.figure(figsize=(10,6))
          plt.plot(range(1,50),error_rate[0:49],color='blue', linestyle='dashed', marker='o',
                   markerfacecolor='red', markersize=10)
          plt.title('Error Rate vs. K Value')
          plt.xlabel('K')
          plt.ylabel('Error Rate')
```

Слика 4.16. Код за приказ зависности

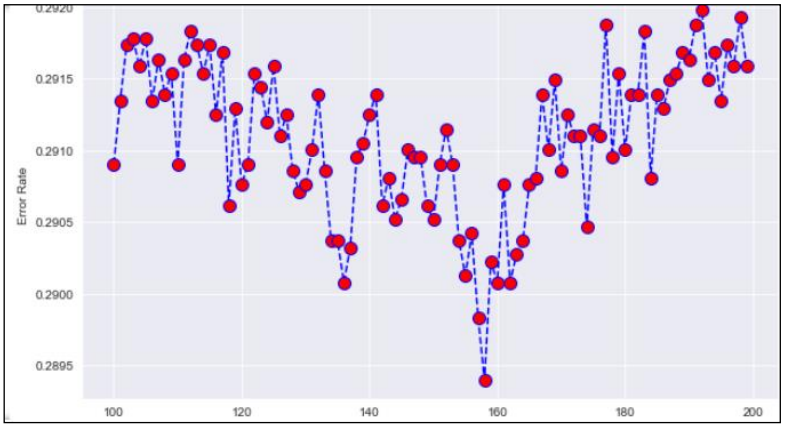
Графици зависности фактора к и процента грешке приказани су на сликама 4.17.1-4.17.5



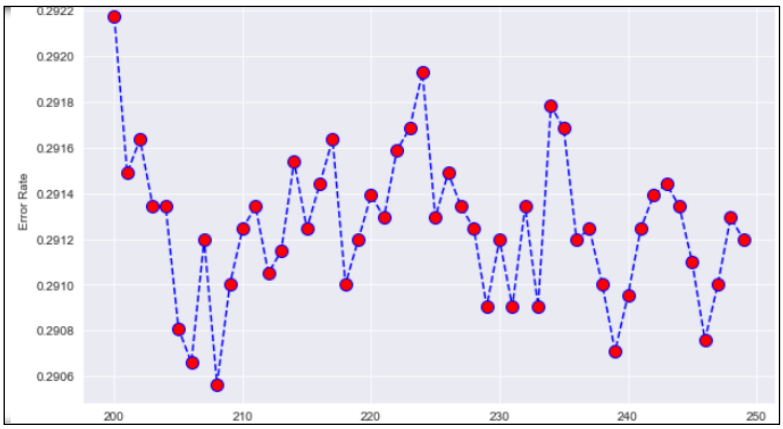
Слика 4.17.1. Фрафик зависности к из интервала (1-50)



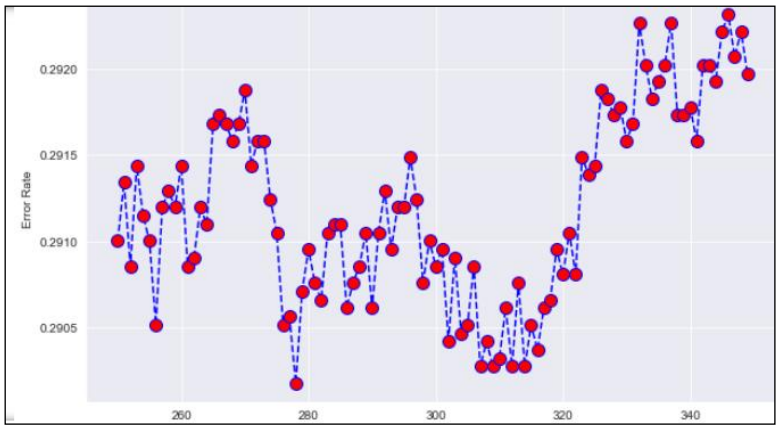
Слика 4.17.2. График зависности к из интервала (50-100)



Слика 4.17.3. График зависности к из интервала (100-200)



Слика 4.17.4. График зависности к из интервала (200-250)



Слика 4.17.5. График зависности к из интервала (250-350)

На основу графика 4.17.1 – 4.17.5, вредност фактора к за коју је проценат грешке најмањи износи 158. Препоручено је да фактор к буде непаран број како би се избегла ситуација када је број најближих суседа из сваке категорије изједначен.

Инстанцира се нови класификатор позивањем конструктора и прослеђивањем вредности за фактор к након чега следи обучавање модела позивањем методе *fit(...)* па након тога одредити тачност предикције модела на основу *score(...)* методе, као на слици 4.18.

```
In [493]: knnOpt = KNeighborsClassifier(n_neighbors=157)
          knnOpt.fit(X_train, y_train)

Out[493]:

In [526]: scores_table.loc['KNN Optimal'] = knnOpt.score(X_test, y_test)*100
          scores_table
```

Out[526]:

	Tacnost predikcije (%)
KNN	70.802529
Stabla odlucivanja	64.153696
KNN Optimal	71.060311

Слика 4.18. Кнн када је к=157

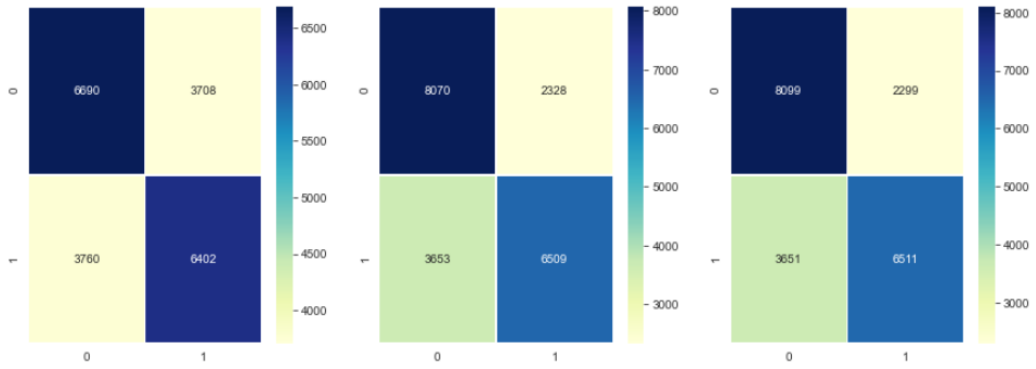
KNN Optimal представља ефикасност алгоритма к најближих суседа са оптималним фактором к = 157. У 71,06% случајева модел предвиди тачну вредност.

Како би се стекао мало бољи увид у ефикасност алгоритама, згодно је приказати бројчано колико пута је модел био у праву за сваку категоричку вредност. Користећи метрику из библиотеке *Scikit-Learn* која се зове *confusion_matrix* која се потом визуализује у виду матрице уз помоћ *heatmap(...)* графика из библиотеке *Seaborn*, као на слици 4.19.

Број који се у матрици са слике налази на позицији *(x, y)* представља колико пута је модел тачну предикцију што је слулај када су *x* и *y* једнаки, или није дао тачну предикцију што је случај ако је *x* различито од *y*. Први график с лева на десно на слици 3.20. представља матрицу за стабло одлучивања, други представља кнн када је к = 100 и трећи који представља кнн када је к = 157.

Збир бројева на позицијама (0, 0) и (1, 1) представља колико пута је модел дао тачну предикцију, бројеви 0 и 1 означавају вредост категоричког атрибута у овом случају то је *cardio* а збир бројева на позицијама (0, 1) и (1, 0) говоре колико пута модел није дао тачну предикцију. Види се да је алгоритам кнн са к = 157 најефикаснији.

```
In [527]: from sklearn.metrics import confusion_matrix

          knnOpt_predictions = knnOpt.predict(X_test)
          dtree_predictions = dtree.predict(X_test)
          knn_predictions = knn.predict(X_test)

          cm_dtree = confusion_matrix(y_test, dtree_predictions)
          cm_knn = confusion_matrix(y_test, knn_predictions)
          cm_knnOpt = confusion_matrix(y_test, knnOpt_predictions)

In [529]: f, ax = plt.subplots(1,3,figsize=(15,5))
          fig.tight_layout()
          sns.heatmap(cm_dtree,fmt=".0f", annot=True,linewidths=0.5, linecolor="white", ax=ax[0], cmap="YlGnBu")
          sns.heatmap(cm_knn,fmt=".0f", annot=True,linewidths=0.5, linecolor="white", ax=ax[1], cmap="YlGnBu")
          sns.heatmap(cm_knnOpt,fmt=".0f", annot=True,linewidths=0.5, linecolor="white", ax=ax[2], cmap="YlGnBu")

Out[529]: <matplotlib.axes._subplots.AxesSubplot at 0x1386848fac0>
```



Слика 4.19 *confusion_matrix*

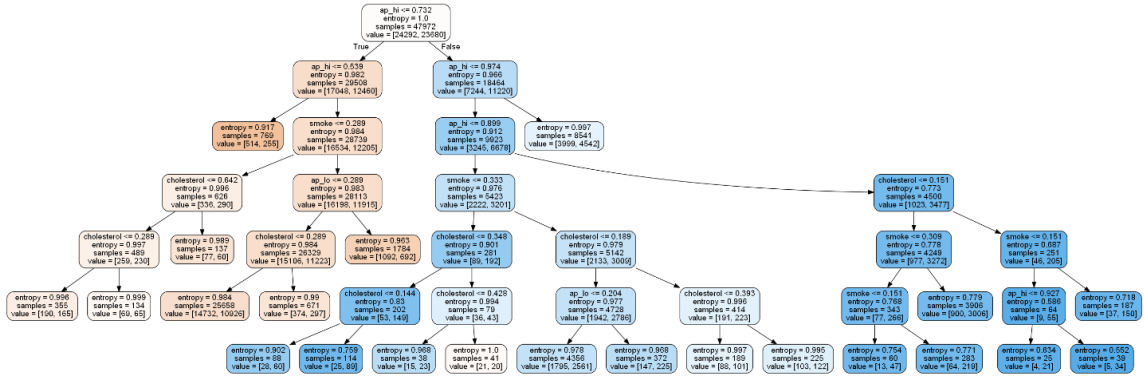
Визуелизација стабла одлучивања које је коришћено у овом проблему немогуће је приказати због великих димензија стабла. Што више атрибута фигурише при одлучивању то ће стабло бити веће, у ту сврху број атрибута који фигуришу при одлучивању смањен је на 4 категоричка атрибута а исцртавање стабла ради се уз помоћ библиотеке *Graphviz*, као на сликама 4.20.1. – 4.20.2.

```
In [129]: from IPython.display import Image
          from io import StringIO
          from sklearn.tree import export_graphviz
          import pydot
          from sklearn import tree

          X = patients[['ap_hi', 'ap_lo', 'cholesterol', 'smoke']]
          features = list(X.columns)
```

Слика 4.20.1. 4 категоричка атрибута

```
In [128]: dot_data = StringIO()
          export_graphviz(dtree, out_file=dot_data,feature_names=features,filled=True,rounded=True)

          graph = pydot.graph_from_dot_data(dot_data.getvalue())
          Image(graph[0].create_png())
```

Out[128]:



Слика 4.20.2. Стабло одлучивања

5.2. Систем за помоћ при стратешком пословном одлучивању

Систем треба да користи компанији која продаје своје производе исључиво путем интернет апликације и путем мобилне апликације. Компанија ће да користи овај систем при доношењу стратешких пословних одлука. Компанија треба да одлучи да ли треба да се фокусира на развој интернет апликације и или мобилне апликације.

Прво наводимо библиотеке које ће нам бити потребне за почетак.

Biblioteke

```
In [44]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
```

Слика 5.1. *Python* библиотеке

Потом је потребно учитати скуп података из датотеке и приказати основне информације о учитаним подацима. Датотека из које се чита назива се „корисници“:

```
In [45]: cust = pd.read_csv('korisnici')
         # Opste informacije o Dataset-u
         cust.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Email                 500 non-null    object
 1   Address               500 non-null    object
 2   Avatar                500 non-null    object
 3   Avg. Session Length   500 non-null    float64
 4   Time on App           500 non-null    float64
 5   Time on Website       500 non-null    float64
 6   Length of Membership  500 non-null    float64
 7   Yearly Amount Spent   500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

Слика 5.2. Информације о учитаним подацима

У скупу података постоји 500 примерака, укупно 8 атрибута/колона и 500 ненултих вредности у свакој колони. Прве три колоне су типа *object* док су све остале типа *float64*. Величина датотеке је 31,4 килобајта. Првих 5 редова изгледа као на слици 5.3.:

	Email	Address	Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	mstephenson@fernandez.com	835 Frank Tunnel\nWrightmouth, MI 82180-9605	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	hduke@hotmail.com	4547 Archer Common\nDiazchester, CA 06566-8576	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	pallen@yahoo.com	24645 Valerie Unions Suite 582\nCobbborough, D...	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	riverarebecca@gmail.com	1414 David Throughway\nPort Jason, OH 22070-1220	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	mstephens@davidson-herman.com	14023 Rodriguez Passage\nPort Jacobville, PR 3...	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

Слика 5.3. Подаци о регистрованим корисницима

Након учитавања и приказивања основних информација из скупа података следи фаза „припреме података“. Први корак је провера да ли има непостојећих података и великих одступања у скупу података. Слика 5.4. представља проверу да ли има непостојећих вредност док слика 5.5. представља опис података који се налазе у скупу на основу којих се закључује да ли постоје одступања. Опис подразумева неке карактеристике, као што су просечна вредност за сваки атрибут, број примерака за сваки атрибут, минималне и максималне вредности атрибута итд., разматрају се само атрибути који су нумеричког типа.

```
In [48]: # Da li postoje Null vrednosti
         print(cust.isnull().sum(), '\n')
         print('Postoji bar jedna Null vrednost:',cust.isnull().values.any(), '\n')
         # Ukupan broj Null vrednosti u Dataset-u
         print('Ukupan broj Null vrednosti:', cust.isnull().sum().sum())

Email                   0
Address                 0
Avatar                  0
Avg. Session Length     0
Time on App             0
Time on Website         0
Length of Membership    0
Yearly Amount Spent     0
dtype: int64

Postoji bar jedna Null vrednost: False

Ukupan broj Null vrednosti: 0
```

Слика 5.4. Провера да ли постоје „*null*“ вредности у скупу података

```
In [47]: cust.describe()
```

Out[47]:

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

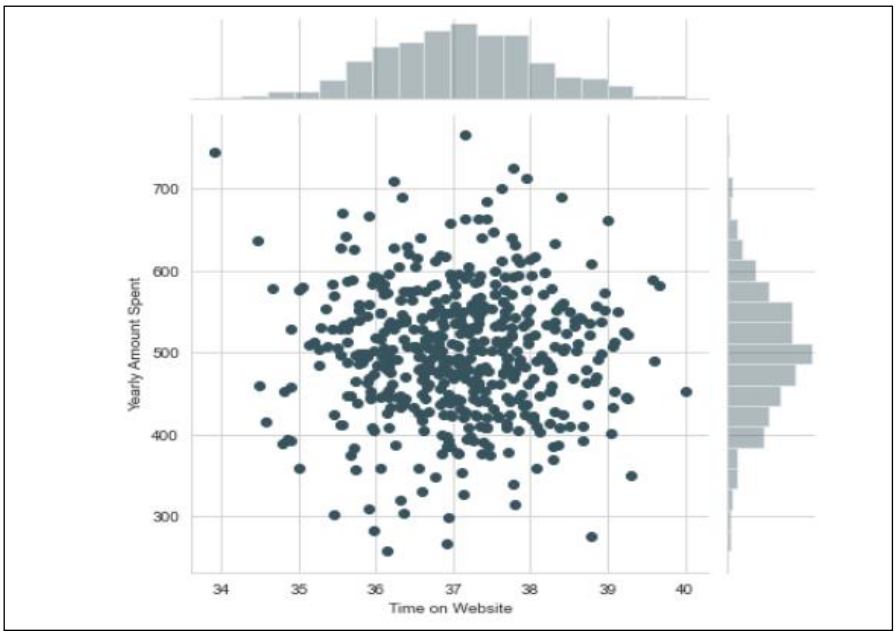
Слика 5.5. Опис података

Квалитетно прикупљени подаци дало је за последицу да у скупу података нема непостојећих вредности као ни великих одступања, с тога није погребно вршити припрему података.

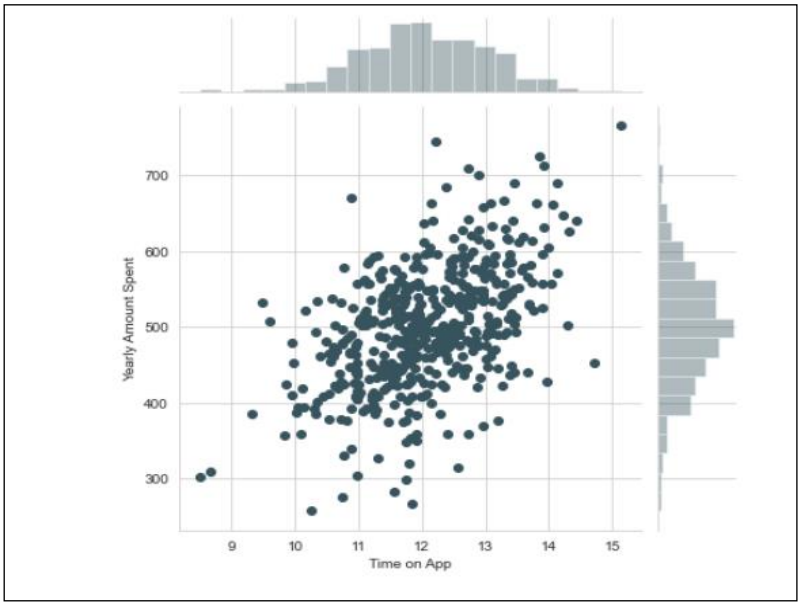
Следећа фаза јесте „експлораторна анализа података“. Циљ је на основу визуелизације података доћи до неких закључака. Пошто се ради о компанији која жели да развија неки тип апликације како би побољшала своје пословање што имплицира да компанија жели да унапреди своје услуге како би привукла више корисника и како би корисници што више новца потрошили користећи њихове апликације. Из тог разлога приказаће се зависност између одређених атрибута скупа података у виду дијаграма расипања са расподелом за сваки атрибут. За визуелизацију се користи *Seaborn* библиотека на начин приказан на слици 5.6. На слици 5.7. и 5.8. приказани су дијаграми расипања атрибута *Time on Website - Yearly Amount Spent* и *Time on App – Yearly Amount Spent* респективно.

```
In [50]: sns.set_palette("GnBu_d")
         sns.set_style('whitegrid')

         sns.jointplot(x='Time on Website',y='Yearly Amount Spent',data=cust)
         sns.jointplot(x='Time on App',y='Yearly Amount Spent',data=cust)
```

Слика 5.6. Исцртавање дијаграма помоћу *Seaborn* библиотеке

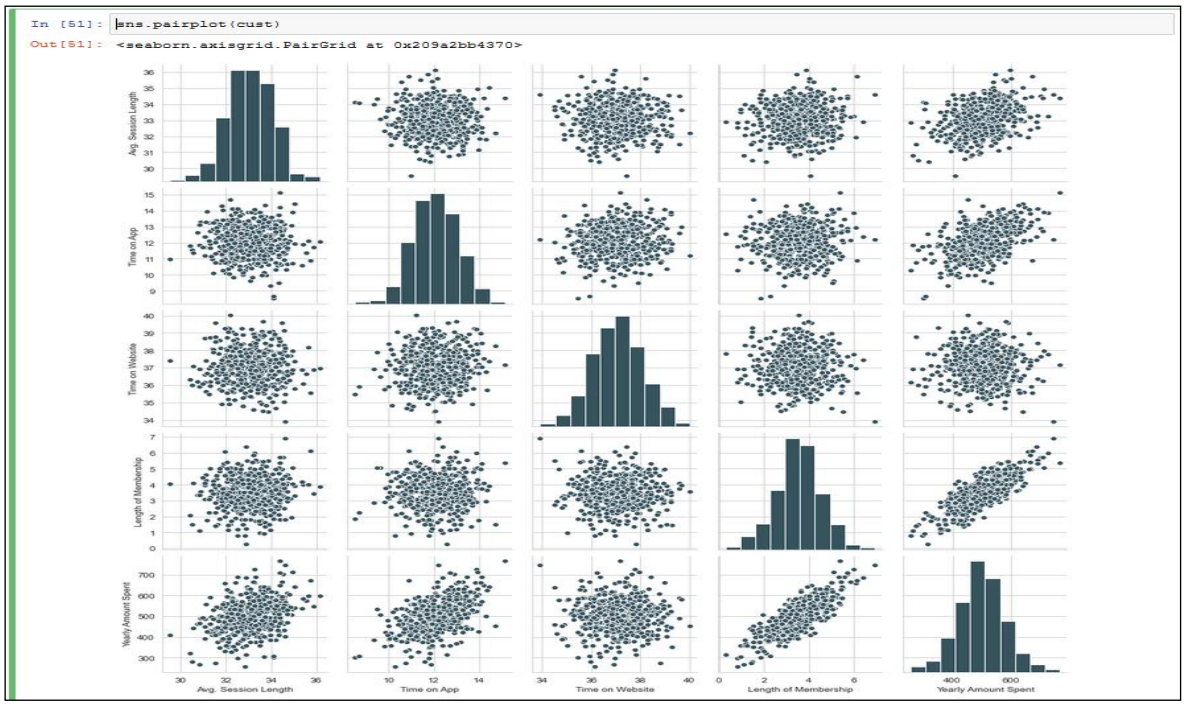


Слика 5.7. Дијаграм расипања атрибута *Time on Website* и *Yearly Amount Spent*

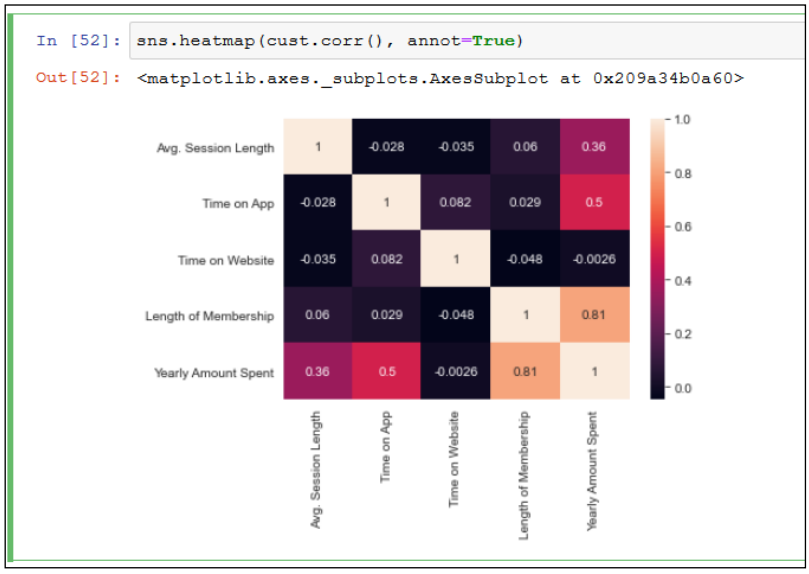


Слика 5.8. Дијаграм расипања атрибута *Time on App* и *Yearly Amount Spent*

На основу дијаграма расипања долази се до закључка да што више времена корисник проводи користећи мобилну апликацију то ће више новца потрошити што значи да је степен повезаности тј. степен корелације између атрибута са слике 5.8. већи него што је то случај са атрибутима на слици 5.7.



Слика 5.9. Дијаграми расипања свих парова атрибута из скупа података



Слика 5.10. Квантификовани степен корелације између свих парова атрибута

На слици 5.9. приказани су дијаграми расипања између свих парова атрибута из скупа података. Потребно је такође на неки начин квантификовати тај степен корелисаности међу атрибутима и визелизацијом приказати те квантификоване степене корелисаности у одговарајућој форми. Променљива *cust* у коју су смештени подаци учитани из датотеке, представља *DataFrame* из пајтонове библиотеке *Pandas* који у себи има методу која имплементира начин квантификовања степена корелације међу колонама *DataFrame*-а. Та метода се зове *corr()*. Помоћу „топлотне мапе“ из библиотеке *Seaborn* приказани су степени

корелисаности између парова атрибута из скупа података, слика 5.10.. Може се приметити да највећи степен корелације постоји између *Length of Membership* и *Yearly Amount Spent* што је и очекивано јер што дуже корисник постоји то ће више пара потрошити.

Након анализе података следећи битан корак у реализацији система јесте избор циљног атрибута и улазних атрибута као и подела скупа података на оне податке који служе за обучавање модела тзв. тренинг подаци и оне податке који служе за евалуацију модела тзв. тест подаци. За овај систем циљна променљива која је изабрана јесте *Yearly Amount Spent* јер је један од битних фактора успешности пословања компаније управо колико новца корисници потроше, док су улазни атрибути сви остали атрибути који су нумеричког типа. Од свих података из скупа 30% ће бити одвојени за тестирање, док ће 70% бити коришћено за обучавање тј. тренирање модела. Подела скупа података реализована је помоћу *Scikit-Learn* библиотеке као на слици 5.11.

```
In [54]: y = cust['Yearly Amount Spent']

In [55]: X = cust[['Avg. Session Length', 'Time on App','Time on Website', 'Length of Membership']]

In [56]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=101)
```

Слика 5.11. Циљни атрибут, улазни атрибути и подела података на тренинг и тест

Следећи корак ка реализацији система јесте „обучавање модела“ приказан на слици 5.12.. За обучавање модела користи се линеарни модел из библиотеке *Scikit-Learn* који се зове *LinearRegression.* Прво се инстанцира објекат *LinearRegression()* потом се позове метода за обучавање која се назива *fit(...)*. Ако је линеарни модел дат изразом:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$$

Тада је y - циљна променљиву, $x_1x_2x_3x_4$ – циљне променљиве, $b_1b_2b_3b_4$ – коефицијенти и b_0 – интерсептор. Циљ *fit(...)* функције јесте да пронађе коефицијенте и интерсептор на основу којих ће модел дати предикцију тј. циљни атрибут (y) за било које улазне атрибуте.

```
In [57]: from sklearn.linear_model import LinearRegression

In [58]: model = LinearRegression()
         model.fit(X_train, y_train)

Out[58]: LinearRegression()

In [59]: koef = pd.DataFrame(model.coef_,X.columns)
         koef.columns = ['Koeficijent']
         koef
```

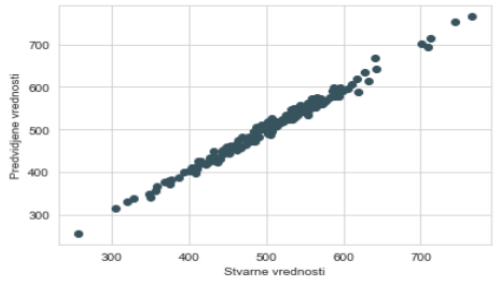
Out[59]:

	Koeficijent
Avg. Session Length	25.981550
Time on App	38.590159
Time on Website	0.190405
Length of Membership	61.279097

Слика 5.12. Циљни атрибут, улазни атрибути, подела података на тренинг и тест

У евалуацији модела прво се одрађује предикција над тест подацима. Тест подаци проследе се функцији *predict(...)* и предвиђене вредности се смештају у променљиву *pred.* Један од начина да се види колико је поуздан модел јесте закључивање на основу дијаграма расипања предвиђених вредности и стварних вредности. Што је већа линеарност на дијаграму то су одступања предвиђених вредности од стварних вредности мања што значи да је модел квалитетнији. Реализација овог корака представљена је на слици 5.13.

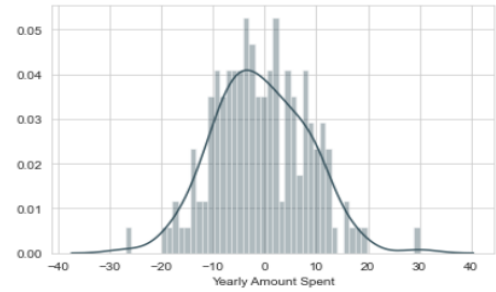
```
In [60]: pred = model.predict(X_test)

In [61]: plt.scatter(y_test,pred)
         plt.xlabel('Stvarne vrednosti')
         plt.ylabel('Predvidjene vrednosti')

Out[61]: Text(0, 0.5, 'Predvidjene vrednosti')
```



Слика 5.13. Предикција модела и дијаграм расипања стварних и предвиђених вредности

Такође, да би модел био што ефикаснији потребно је да разлика предвиђених и стварних вредности/атрибута има нормалну расподелу, у овом случају то је испуњено, што доприноси поузданости у квалитет модела. Расподела разлика предвиђених и стварних вредности приказана је на слици 5.14. одакле може да се закључи да је та расподела нормална.

```
In [63]: sns.distplot((y_test-pred),bins=50);
```



Слика 5.14. Расподела разлика између стварних и предвиђених вредности

Ефикасност модела потребно је на неки начин квантификовати, што ће бити урађено коришћењем *score(...)* методу из модела. *score(...)* је заснована на статистичкој мери која се назива R^2. Резултат методе је број у интервалу (0, 1) који говори колико је процентуално модел дао тачну предикцију, слика 5.15.

```
print(model.score(X_test, y_test)*100,"%")
```

```
98.90046246741232 %
```

Слика 5.15. Проценат тачности предикције модела

Још увек није одговорено на првобитно питање, а то је: „Да ли компанија треба да се фокусира на развој мобилне или интернет апликације?“. Ако се поново погледају вредности коефицијената вишеструког линеарног модела:

	Koeficijent
Avg. Session Length	25.981550
Time on App	38.590159
Time on Website	0.190405
Length of Membership	61.279097

Слика 5.16. Коефицијенти линеарног модела

Један од начина на који је могуће интерпретирати ове вредности јесте:

- ако се фиксирају све вредности атрибута осим *Avg. Session Length*, повећањем вредности овог атрибута за 1 резултираће повећањем циљног атрибута *Yearly Amount Spent* за 25.981550 долара
- ако се фиксирају све вредности атрибута осим *Time on App*, повећањем вредности овог атрибута за 1 резултираће повећањем циљног атрибута *Yearly Amount Spent* за 38.590159 долара
- ако се фиксирају све вредности атрибута осим *Time on Website*, повећањем вредности овог атрибута за 1 резултираће повећањем циљног атрибута *Yearly Amount Spent* за 0.190405 долара
- ако се фиксирају све вредности атрибута осим *Length of Membership*, повећањем вредности овог атрибута за 1 резултираће повећањем циљног атрибута *Yearly Amount Spent* за 61.279097 долара

Питање на које треба дати одговор може се посматрати на два начина:

1. Развијати интернет апликацију како би могла да сустигне мобилну апликацију
2. Развијати мобилну апликацију још више јер се показује као много боља од интернет апликације

6. Закључак

Циљ овог дипломског рада јесте примена алгоритама надгледаног машинског учења на неке реалне проблеме и упознавање са основним концептима машинског учења и науке о подацима. Прво је дат увод у област машинског учења где су представљени значај и употреба машинског учења као и описи група учења које постоје у машинском учењу. Потом је дат мало детаљнији увид у начин на који је могуће реализовати системе засноване на машинском учењу, тзв. процес машинског учења који се састоји из неколико фаза које треба да буду реализоване. Након описа процеса машинског учења прешло се на објашњавање алгоритама из групе надгледаног учења на фундаменталнијем нивоу, обрађени су алгоритми линеарне регресије, стабла одлучивања и к најближих суседа. Потом је дат кратак опис начина на који су реализоване све фазе машинског учења и помоћу којих алата је то спроведено у дело, након чега су детаљно представљени резултати рада који, подразумева детаљан опис процедура на основу којих је комплетан пројекат одрађен, и дискусија која вођена на основу резултата.

Међу најбитнијим стварима које су описане у овом документу јесу фундаментално разумевање концепата и поделе машинског учења и колико се такав вид индуктивног учења разликује од дедуктивног учења, опис рада алгоритама надгледаног машинског учења и технике из области науке о подацима коришћене при анализи података. Такође, битан концепт је и сама реализација система заснованих на машинском учењу који се састоје од 7 битних фаза као и врло погодни, лаки за разумевање алати који су се користили при реализацији тих фаза као и окружење *Jupyter* помоћу ког је врло лако водити евиденцију о раду. Тренутно, један од највећих проблема машинског учења је тај што не постоји пуно једноставног материјала који људи треба да користе како би стекли што ширу слику о проблематици. Материјали су често нестандардизовани и често се у њима проблему приступа из доста различитих углова што је углавном због особине интердисциплинарности самог машинског учења.

Два система која су реализована у оквиру овог рада, систем за подршку при одлучивању да ли пацијент има кардиоваскуларне проблеме и систем за подршку при стратешком пословном одлучивању засновани су на проблемима класификације и регресије, респективно. За проблем класификације коришћени су алгоритми стабло одлучивања и к најближих суседа док је за проблем регресије коришћена линеарна регресија. Избор алгоритама који су коришћени за реализацију оба система у оквиру овог рада заснован је на разлозима који су едукативне природе. Да би се неки проблем решио коришћењем алгоритама машинског учења потребно је добро истражити којој групи машинског учења проблем припада као и ком типу проблема, који постоје у оквиру одговарајуће групе машинског учења, посматрани проблем припада. Примера ради, у оквиру проблема регресије из групе надгледаног машинског учења постоји доста алгоритама који могу да се примене на такав тип проблема.

У систему за подршку при стратешком пословном одлучивању који представља проблем регресије, коришћен је само један алгоритам а то је линеарна регресија док је систем за подршку при одлучивању да ли пацијент има кардиоваскуларне проблеме

реализован помоћу два алгоритма. Системи могу да се оптимизује на више начина. Неки од њих подразумевају примену различитих алгоритама као и примену различитих имплементација алгоритама на исти проблем и на основу упоређивања резултата сваког од алгоритама и сваке од варијанти (начина имплементације) алгоритма изабрати онај алгоритам и ону варијанту алгоритма који дају најбоље резултате.

Такође, оно што значајно доприноси квалитету анализе и тачности система јесу подаци. У првом систему подаци који су прикупљени су „чисти“ што је добро, али их је мало а квантитет података може добро да доприноси квалитету предиктивног модела док су подаци за други систем нешто мање „чисти“ али их има јако пуно. Ни квантитет ни „чистоћа“ прикупљених података не играју пресудну улогу у доприносу квалитета предиктивног модела, често се деси да има много података и да су чисти али због самог садржаја тих података модел можда не може да да квалитетну предикцију. Експлораторном анализом може да се закључи колико су неки подаци квалитетно прикупљени. Што више атрибута подаци везани за одређени проблем имају, то је већа вероватноћа да се из таквих података могу извући корисне информације које могу бити кључне за посматрани проблем. Могуће је радити инжињеринг над атрибутима што између осталог подразумева тражење односа међу атрибутима и онда те односе користити као независан индикатор. Често се у природи налази да односи неких променљивих атрибута могу значајно да утичу на одређеност неких циљних променљивих.

На доста начина могуће је даљим истраживањем побољшавати и проширитвати могућности реализованих система. Код система за стратешке одлуке у пословању, примењени алгоритам машинског учења коришћен је само за једну врсту проблема тј. за питање да ли развијати интернет или мобилну апликацију. Пословање само по себи је често комплексно и од људи који управљају компанијама се очекује да доносе најбоље одлуке како би компанија била све више профитабилнија. Могуће би било дефинисати најкруцијалније аспекте пословања, тј. дефинисати факторе који највише утичу на профитабилност компаније, потом одредити која врста информација је потребна како би се стекао што бољи увид у сваки од фактора, потом прикупити те податке и изабрати одговарајући алгоритам машинског учења чији модел би био у стању да пружи најквалитетније и најпоузданије информације. Код система за одлучивање да ли пацијент има кардиоваскуларне проблеме виђено је да подаци који су прикупљени нису баш најквалитетнији па би од највећег значаја за квалитет предиктивног модела било управо прикупљање квалитетнијих података. Такође, када се ради са подацима о пацијентима, често је могуће одрадити квалитетан инжињеринг над атрибутима као нпр. формирање новог атрибута на основу два постојећа атрибута. Та два постојећа атрибута могла би бити *height* и *weight* из скупа података о пацијентима. Ако се погледа однос ова два атрибута добија се тзв. *BMI(Body Mass Index)* који одређује степен гојазности пацијента који може да послужи као добар индикатор за предикцију кардиоваскуларних проблема тј. ако неко има већи степен гојазности, постоји већа шанса да пати од кардиоваскуларних проблема. Свакако, систем би могао да се унапреди избором алгоритама који дају боље резултате и да се прошири у смислу да може да даје предикције и за неке друге врсте здравствених проблема.

РЕФЕРЕНЦЕ

[1] *Benedict Neo – Top 20 Websites for Machine Learning and Data Science in 2020*, https://medium.com/swlh/top-20-websites-for-machine-learning-and-data-science-d0b113130068 , прочитано 15.08.2020.г.

[2] Master akademske studije na modulu računarska tehnika i informatika, Elektrotehnički fakultet, Univerzitet u Beogradu – „Pronalaženje skrivenog znanja“ , http://rti.etf.bg.ac.rs/rti/ms1psz/

[3] Sebastian Raschka i Vahid Mirjalili – Mašinsko učenje i duboko učenje pomoću Pythona, scikit-learn biblioteke i TensorFlowa 2.

[4] *Gavin Edwards – Machine Learning | An introduction, www.towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590.com* , прочитано дана 28.08.2020.г.

[5] *Aidan Wilson – A Brief Introduction to Supervised Learning, https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590* , прочитано дана 28.08.2020.г.

[6] *Wikipedia – Data Analysis – https://en.wikipedia.org/wiki/Data_analysis*, прочитано 29.08.2020.г.

[7] *Wikipedia – Exploratory Data Analysis – https://en.wikipedia.org/wiki/Exploratory_data_analysis* , прочитано 30.08.2020.г.

[8] *Jung Wu – AI, Machine Learning, Deep Learning Explained Simply - https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960* , прочитано дана 30.08.2020.г.

[9] *Yufang G – The 7 Steps of Machine Learning, https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e*, прочитано дана 01.09.2020.г.

[10]*Stuart Russell, Peter Norvig – Artificial Intelligence: A Modern Approach*

[11] *TutorialsPoint – What is Weka? – https://www.tutorialspoint.com/weka/what_is_weka.htm* , прочитано 01.09.2020.г.

[12] *Nikhil Gupta – Why is Python Used for Machine Learning? - https://hackernoon.com/why-python-used-for-machine-learning-u13f922ug* , прочитано 01.09.2020.г.

[13] *Jupyter Team – What is Jupyter Notebook? – https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/What%20is%20the%20Jupyter%20Notebook.html*

СПИСАК СЛИКА