УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Никола Карпић

Развој рјешења за предвиђање броја особа у просторији

дипломски рад

Бања Лука, новембар 2023.

Тема: **РАЗВОЈ РЈЕШЕЊА ЗА ПРЕДВИЂАЊЕ БРОЈА ОСОБА У ПРОСТОРИЈИ**

Кључне ријечи:

Машинско учење
Регресија
Класификација
LightGBM

Комисија:

проф. др Милош Љубојевић, предсједник
проф. др Зоран Ђурић, ментор
Александар Келеч, ма, члан

Уз рад је приложен CD.

Кандидат:
Никола Карпић

УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
КАТЕДРА ЗА РАЧУНАРСТВО И ИНФОРМАТИКУ

Тема: РАЗВОЈ РЈЕШЕЊА ЗА ПРЕДВИЂАЊЕ БРОЈА ОСОБА У ПРОСТОРИЈИ

Задатак: Машинско учење. Описати класификационе и регресионе алгоритме машинског учења и њихове типичне представнике. Припрема скупова података за тренирање, валидацију и тестирање. У практичном дијелу рада анализирати моделе за предвиђања броја особа у просторији креиране коришћењем неколико алгоритама (класификационих и регресионих). Моделе је потребно тренирати подацима који садрже температуру, влажност, ниво угљен-диоксида и др. За реализацију користити *Jupyter Notebook* алат. Извршити компаративну анализу перформаси добијених модела.

Ментор: проф. др Зоран Ђурић

Кандидат: Никола Карпић (1144/14)

Бања Лука, новембар 2023.

Садржај

1. Увод

У данашње вријеме се суочавамо са феноменом све веће производње дигиталних података у разним форматима. Количина података се експоненцијално повећава и ти подаци су огроман ресурс који остаје неискориштен. Због наглог повећања количине података, појавили су се могућност и потреба за обрадом тих података. Како је велики проценат тих података неструктурисан и некласификован, појавила се потреба и за креирањем ефикасних алгоритама и процеса за разврставање, именовање и анализу тих података. Једна од најзаступљенијих области вјештачке интелигенције (енг. *Artificial Intelligence*) која се бави разврставањем, именовањем и анализом података је машинско учење (енг. *Machine learning*).

Велике компаније као што су Google[1], Microsoft[2], Facebook[3] Amazon[4] и OpenAI[5] су почеле да улажу огромне напоре и средства у своје могућности чувања, обраде и класификације великих количина несређених података како би остале релевантне и остариле тржишну предност у данашњем брзорастућем и брзомијењајућем дигиталном пространству. Велика количина средстава која се улажу у ову област је довела до експлозије научних радова на тему машинског учења у посљедњих 15 година (слика 1.1).

Владе, војске, Министарства унутрашњих послова и остале институције разних држава улажу у ову област, такође, било због могућности да боље и ефикасније прате своје грађане и брже проналазе могуће терористе и преступнике, било због постизања предности над другим државама у војном или обавјештајном сектору.

Машинско учење је облик вјештачке интелигенције који омогућава систему да учи из података, а не путем експлицитног програмирања. Машинско учење користи низ алгоритама који итеративно уче из података да би побољшали, описали податке и предвидјели исходе. Како алгоритми уносе више података за учење, тако је могуће произвести све прецизније моделе засноване на тим подацима. [1]

Слика 1.1. Број научних радова на тему машинског учења у свијету од 2000. године[6]

[1] https://www.google.com/
[2] https://www.microsoft.com/
[3] https://www.facebook.com/
[4] https://www.amazon.com/
[5] https://openai.com/
[6] https://www.forbes.com/sites/louiscolumbus/2018/01/12/10-charts-that-will-change-your-perspective-on-artificial-intelligences-growth

Алгоритми машинског учења се убрзано увлаче у све сфере живота данашњих људи. Све је чешће да породице посједују дигиталног асистента који се контролише звуком или помоћу паметног телефона. Све су чешћи паметни кућански уређаји и имплементације концепата као што су паметне куће са разним сензорима (камере, микрофони, детектори пожара, нивоа влаге, угљен-диоксида, освијетљености, итд...) у свим просторијама.

Компаније често користе податке о броју особа у просторији, како би направиле уштеде на трошковима гријања и како би смањиле свој угљенични отисак. Смањењем угљеничног отиска компаније повећавају своју прихватљивост у очима све више еколошки освијештених потенцијалних нових клијената и генеришу милијарде долара годишње продаје. [2] Ти сензори могу да представљају проблем за осјећај приватности у дому или на радном мјесту.

Слика 1.2. Паметни усисивач LUCY који посједује огроман број сензора (чак и камеру)

Из наведених разлога се тежи да се што више корисних података добија из сензора који не нарушавају приватност директно као што то раде микрофони и камере, па се прибјегава техникама које индиректно процјењују тражене величине из података које нам дају сензори који нису толико инвазивни у погледу приватности. Још једна брига је и очување пословних тајни у компанијама, пошто се помоћу микрофона и камера, које се могу користити за процјену броја особа у просторијама, релативно једноставно може доћи до пословних тајни изговорених на затвореним састанцима, ако постоје и најмањи сигурносни пропусти у сигурносном систему компаније.

У овом раду ће се обрађивати примјењивање одређеног броја класификационих и регресионих машинских алгоритама за процјену броја особа у просторији на основу датума, времена и података прикупљених сензорима за ниво угљен-диоксида у просторији, осивјетљеност просторије, количину влаге у просторији и сензорима за количину кретања у просторији. Алгоритми машинског учења које ћемо користити су: логистичка регресија, *Gaussian Naive Byes, K-Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, Support Vector Machine, LightGBM*.

У другој глави се даје објашњење вјештачке интелигенције и уопштено објашњење метода машинског учења и процеса примјене алгоритама машинског учења.

У трећој глави су детаљно објашњени кораци креирања модела машинског учења помоћу алгоритама машинског учења.

У четвртој глави су детаљније описани алгоритми који су кориштени у овом раду.

У петој глави се налазе појединости које се односе на практични дио. У њој су детаљно описани подаци и примјена сваког алгоритма, те упоредна анализе резултата. Сав код је написан у програмском језику Python.

У седмој глави се налази опис упоредне анализе резултата и коначан одабир најбољег алгоритма.

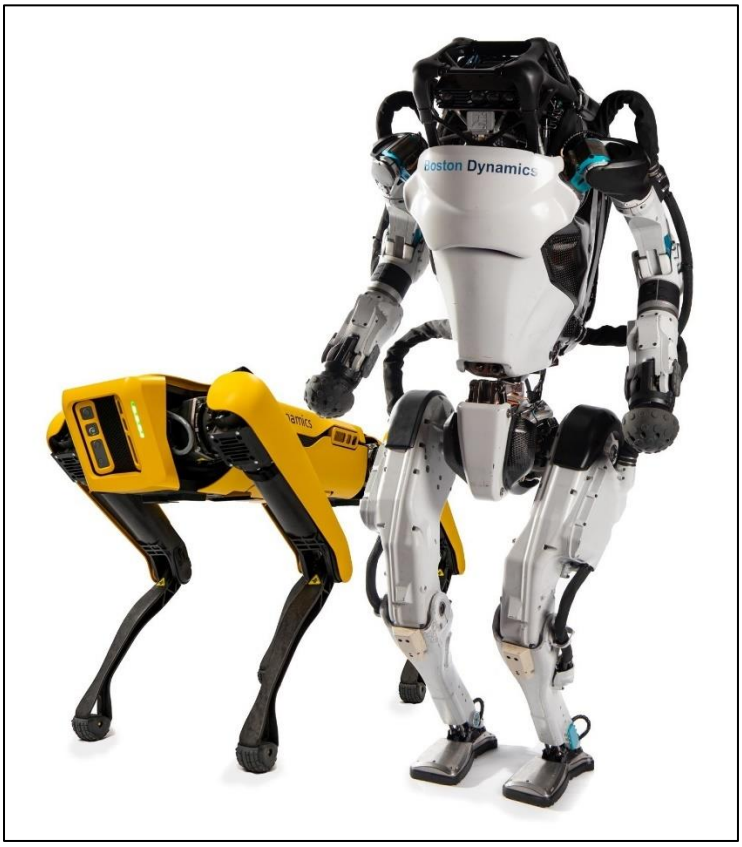
У шестој глави се налази закључак који је добијен у овом раду.

2. Машинско учење

Вјештачка интелигенција је једно од најновијих поља у науци и инжењерству. Ова област је почела озбиљно да се развија убрзо након Другог свјетског рата, а само име је настало 1956. Заједно са молекуларном биологијом, вјештачка интелигенција се редовно наводи као „поље у којем бих највише волио да будем“ од научника из других дисциплина.

Студент физике може оправдано да сматра да су све добре идеје већ преузели Галилео, Њутн, Тесла, Ајнштајн и остали. Вјештачка интелигенција, с друге стране, још увијек има мјеста за неколико нових Ајнштајна и Тесли. Вјештачка интелигенција тренутно обухвата огроман број области, у распону од општег (учење и перцепција) до специфичних, као што су играње шаха, доказивање математичких теорема, писање поезије, вожња аутомобила у препуној улици и дијагностиковање болести. Вјештачка интелигенција је релевантна за било који интелектуални задатак; то је заиста универзално поље. [3]

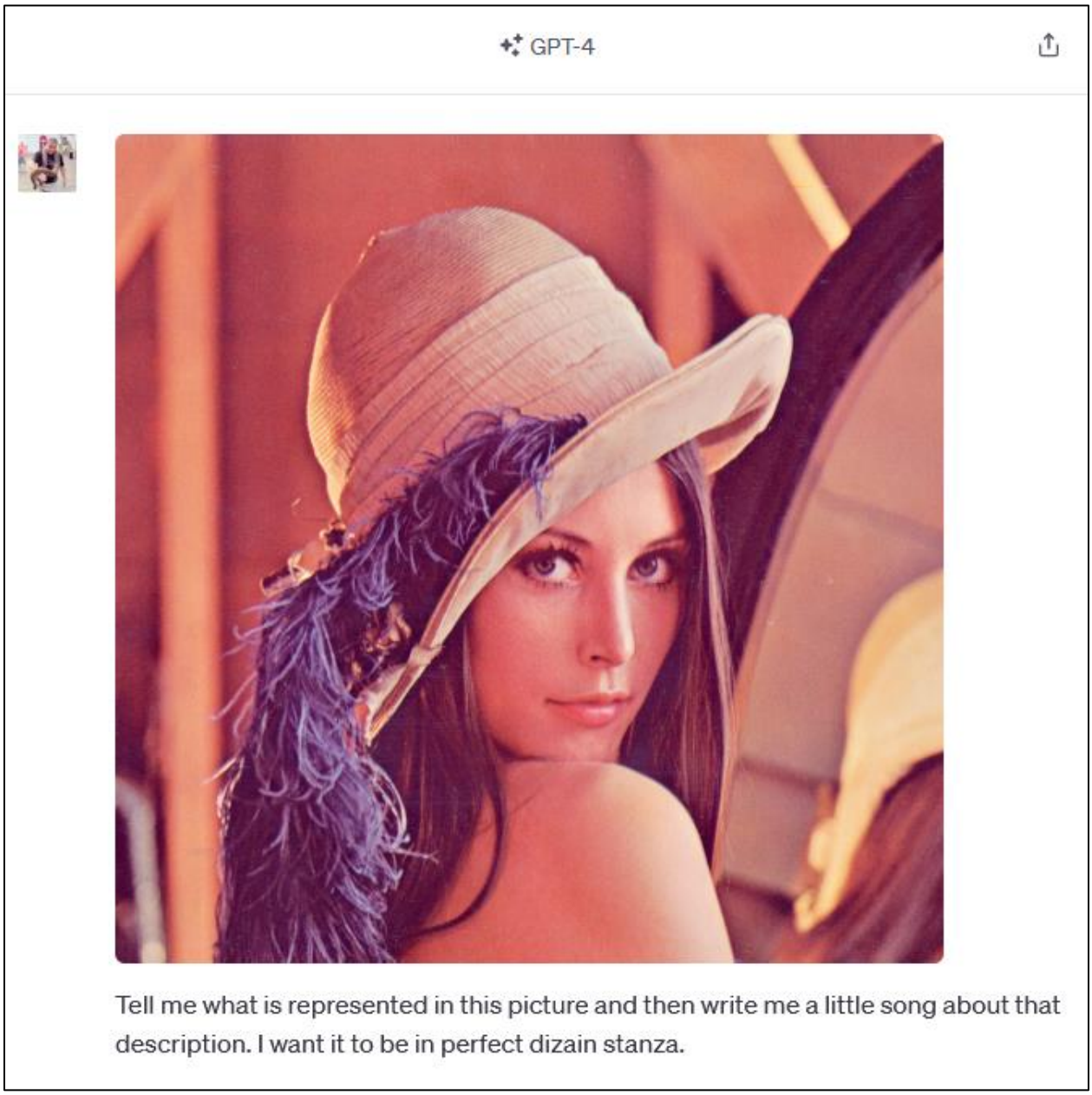
До нагле популаризације вјештачке интелигенције и појачаног интересовања за њене могуће примјене у физичком свијету, довели су пројекти човјеколиког робота (Atlas) и роботског пса (Spot) америчке компаније *Boston Dynamic*.



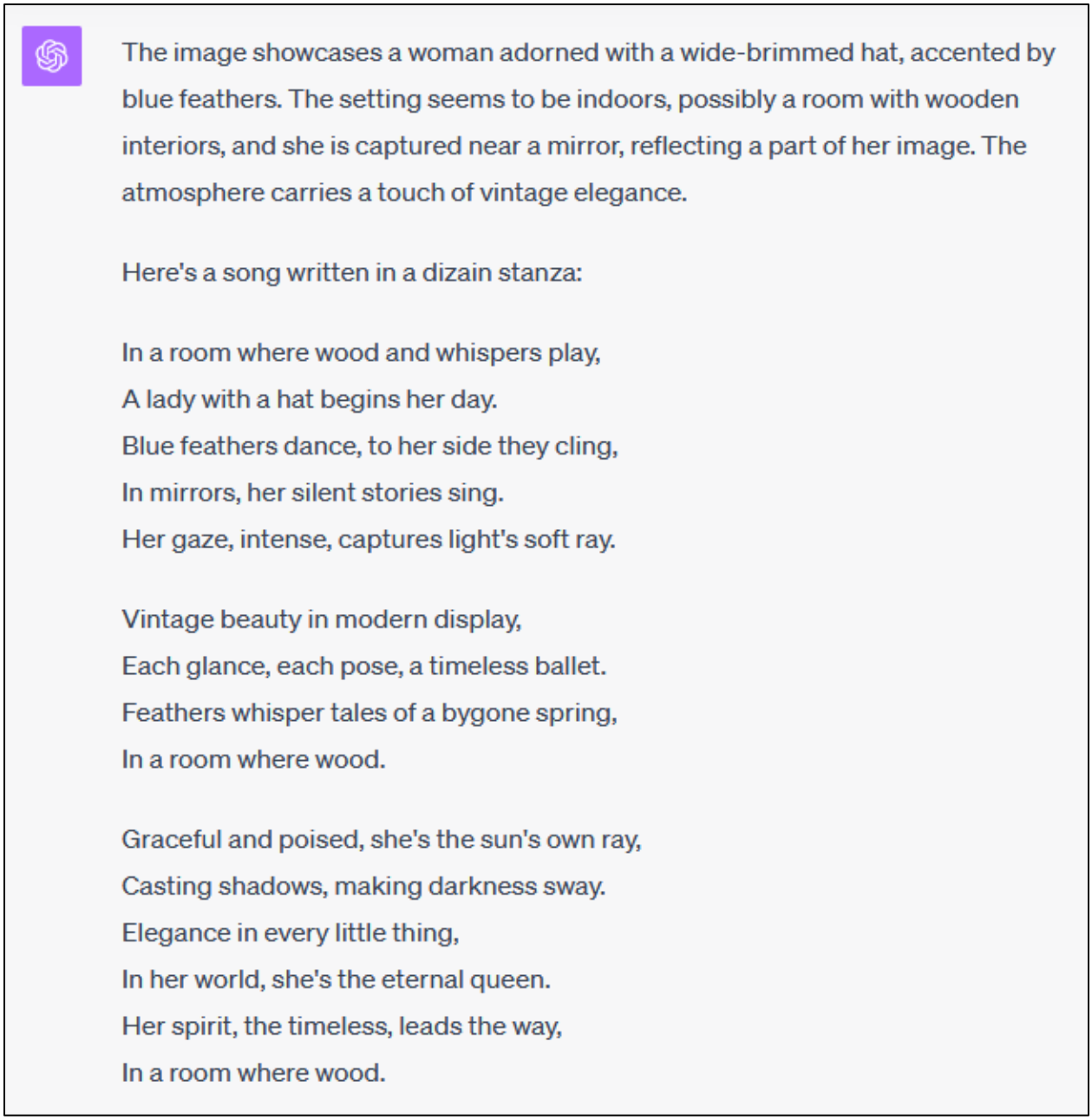
Слика 2.1. Роботски пас и човјеколики робот компаније Boston Dynamics[7]

[7] https://bostondynamics.com/

У посљедње вријеме, *ChatGPT* компаније *Open AI* је довео до наглог повећања интересовања шире јавности за вјештачку интелигенцију. Међутим, ово повећање интересовања истовремено отвара и питање ограничавања развоја вјештачке интелигенције. Људи постају свјесни важности етичких, правних и друштвених аспеката ове технологије и постаје све битније потражити одговоре на питања о томе како осигурати да вјештачка интелигенција буде развијена на начин који користи цјелокупном човјечанству.



Слика 2.2. Захтјев упућен систему ChatGPT компаније Open AI



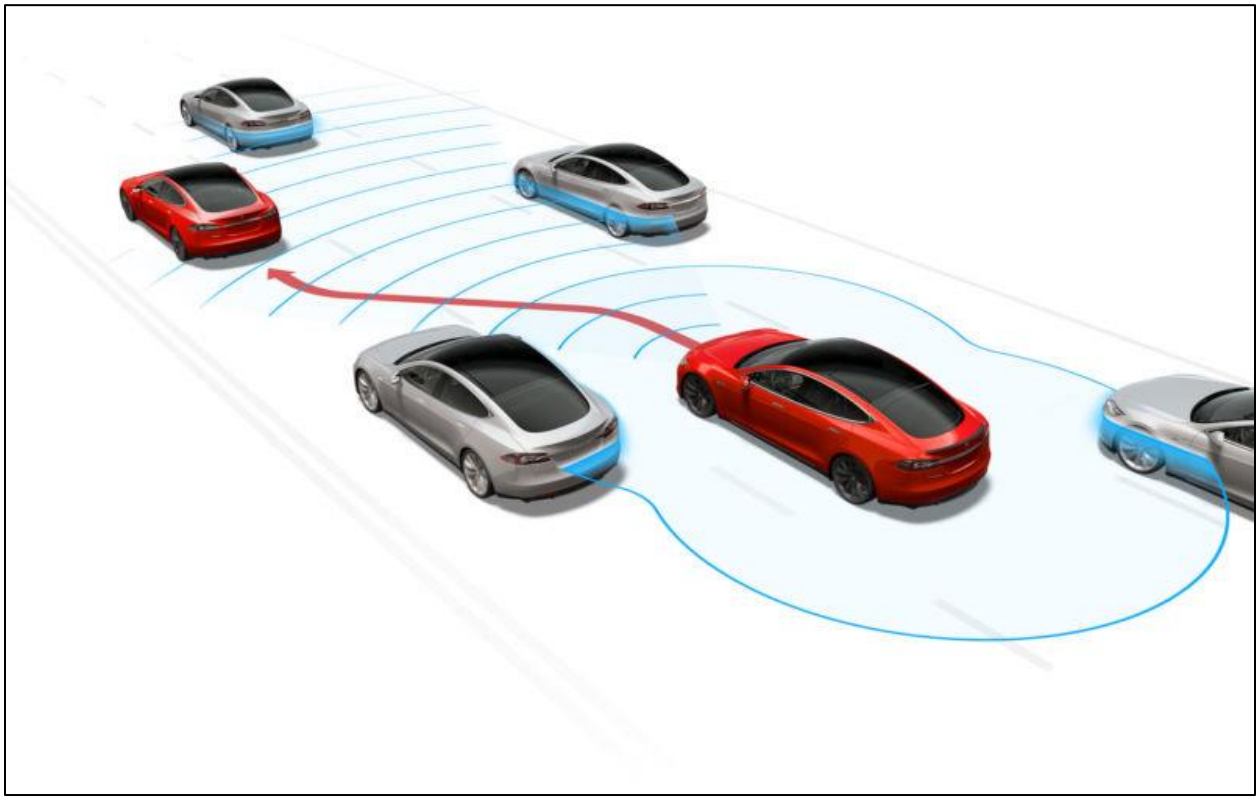
Слика 2.3. ChatGPT препознаје фотографију, описује је и пише пјесму о њој[8]

Као што је већ речено у уводу, машинско учење је облик вјештачке интелигенције који омогућава систему да учи из података, а не путем експлицитног програмирања. Међутим, машинско учење није једноставан процес. Машинско учење користи низ алгоритама који итеративно уче из података да би побољшали, описали податке и предвидјели исходе. Како алгоритми уносе више података за учење, тако је могуће произвести све прецизније моделе засноване на тим подацима. [1] Алгоритми машинског учења као улаз примају огромне скупове података који описују одређене појаве и обрађују

[8] https://chat.openai.com/

их и у њима проналазе правилности, а као излаз стварају моделе који помоћу пронађених правилности могу да предвиђају резултате тих појава над невиђеним улазним подацима.

Неки од типичних примјера у којима се машинско учење показало као одговарајући алат су аутономна возила, препоручена претрага код интернетских претраживача, добијање повратне информације о томе шта купци мисле о компанији, препознавање нежељених порука, откривање превара. Технике машинског учења се обично дијеле на три области у зависности од врсте повратне информације доступне систему учења, а то су: надгледано учење, ненадгледано учење и учење уз подстицај.

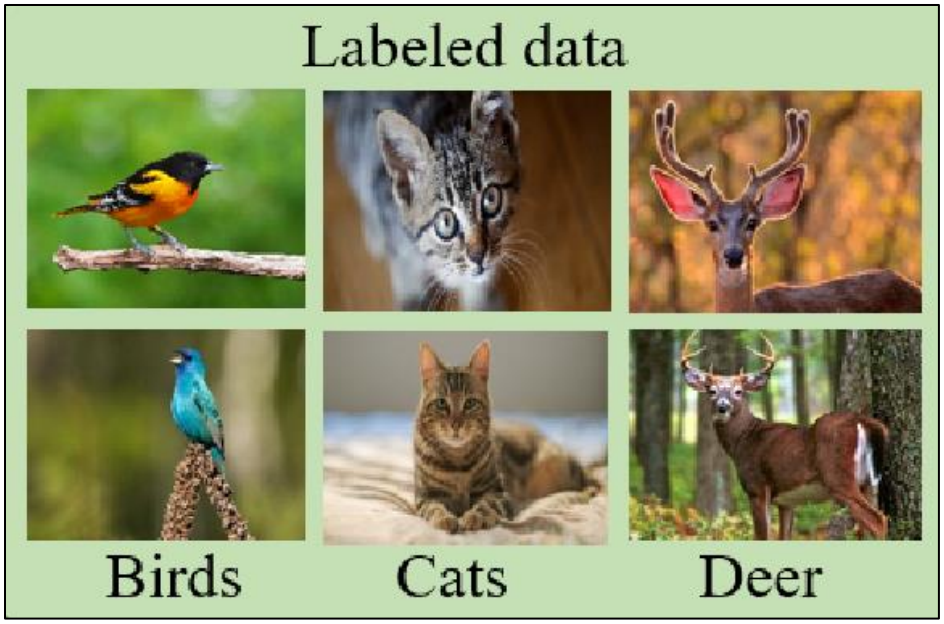


Слика 2.4. Приказ аутономне промјене возне траке возила компаније Tesla Motors [9]

2.1. Надгледано учење

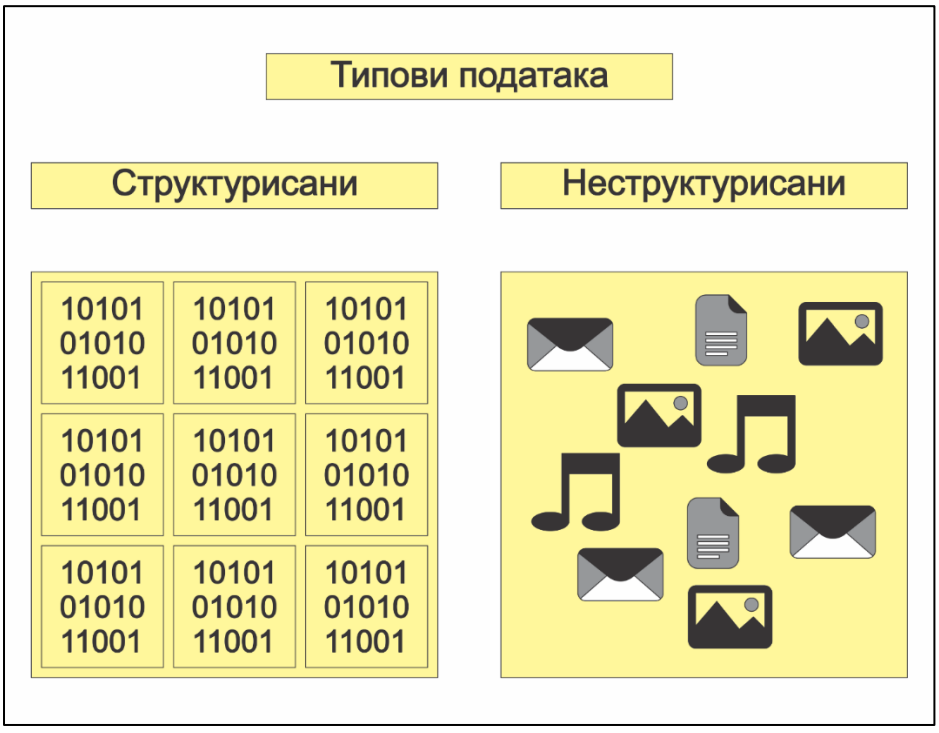
Надгледано учење *(енг. Supervised Learning)* је врста машинског учења у коме алгоритам проучава функцију која пресликава улазне податке на излазне како би научио функцију којом може да предвиди излазе за нове, непознате улазне податке. [3] Та функција се назива моделом. Алгоритам учења добија одређен скуп парова улаз-излаз као скуп података за учење и даје функцију предвиђања за све могуће улазе. Укупан број ставки и природа улазних података може утицати на одабир најбољег алгоритма за специфичан проблем.

[9] https://www.tesla.com/



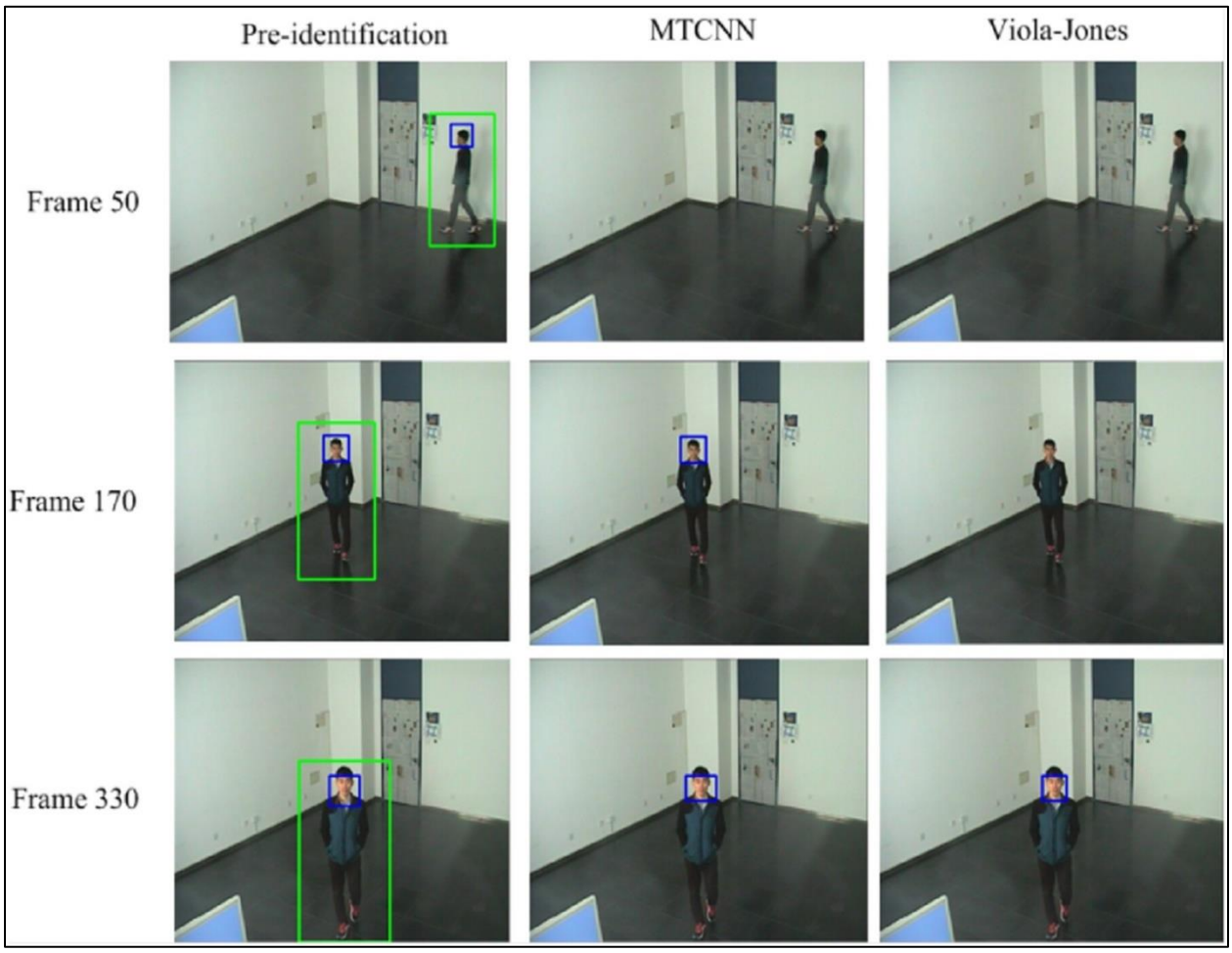
Слика 2.5. Примјер означеног скупа података

Пошто улаз модела чине подаци, они се могу јавити у структурисаном, неструктурисаном и полуструктурисаном облику. Структурисани подаци су сви подаци који се могу интерпретирати у неком фиксном формату (физичка мјерења, човечанству, бројеви телефона, број особа, спол), док су неструктурисани подаци они подаци за које не постоји фиксна структура (разне текстуалне датотеке, фотографије, видео записи, звучни записи, веб странице, резултати претраживања на вебу).



Слика 2.6. Примјер означеног скупа података

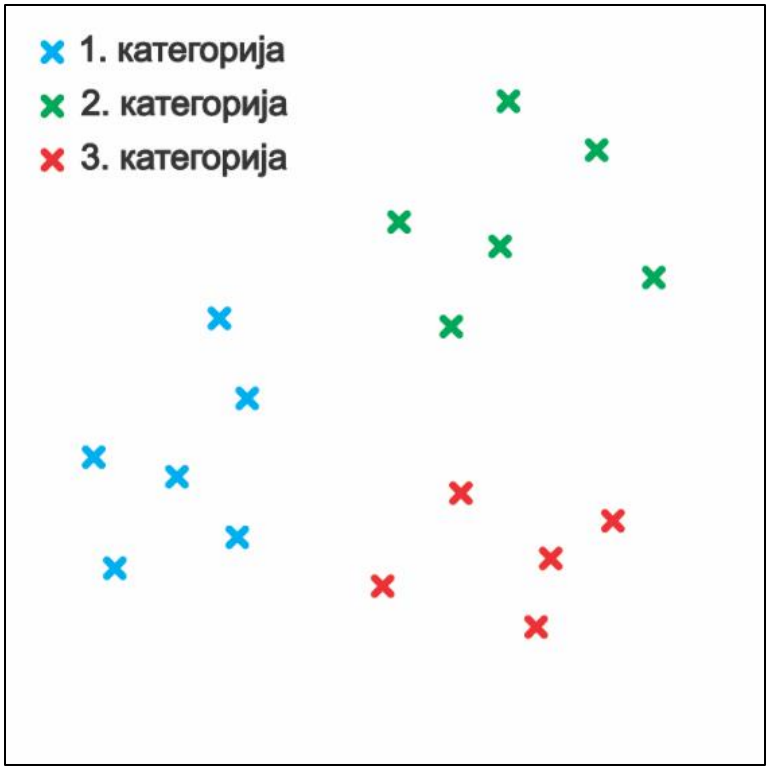
Надгледано учење је најчешћа врста машинског учења која се користи код проблема класификације, регресије и рангирања. Проблем откривања нежељене поште је један од најпознатијих примјера код којих се користи надгледано учење. [4] Надгледано машинско учење се често користи и у апликацијама попут кредитне процјене, медицинске дијагностике, временске прогнозе и персонализације реклама. [5]



Слика 2.7. Препознавање лица у реалном времену [6]

2.1.1. Класификација

Класификација представља технику додјељивања категорије свакој ставци из скупа података. На примјер, класификација докумената састоји се од додјељивања категорија као што су политика, посао, спорт или вријеме сваком документу, док се класификација слика састоји од додјељивања категорије свакој слици као што су аутомобил, воз или авион. Број категорија у таквим задацима често је мањи од неколико стотина, али може бити много већи у неким тешким задацима и чак неограничен као у класификацији текста или препознавању говора. [4]



Слика 2.8. Примјер класификације у три класе

Постоје разне технике класификације, укључујући логистичку регресију, K-nn, стабла одлучивања, и наивни Бајес. Ове технике су различите по начину на који се користе атрибути мјерења и њихови односи са познатим категоријама. Технике се разликују и по начину додјеле нових објекта у категорије.

Класификација може бити надгледана или ненадгледана. У надгледаној класификацији, подаци садрже позитивне примјере за сваку категорију, док се у ненадгледаној класификацији тражи да алгоритам сам пронађе категорије, а можемусе и и спецификовати колико тачно категорија има у улазном скупу података. Да би класификација била квалитетно одрађена подребно је да се припреме подаци за класификацију, укључујући процес нормализације и екстракције ставки.

Процјена и оцјењивање квалитета класификације се ради помоћу метрика као што су тачности, прецизности и F-мјере. Овај процес помаже у одређивању колико добро алгоритам функционише у стварним примјенама и омогућава избор најбољег модела за одређени проблем.

Класификација у машинском учењу користи се за аутоматско разврставање објеката и препознавање значаја података, што помаже у доношењу одлука и побољшању пословања у многим индустријама, као што су здравство, финансије, и маркетинг. [5]

2.1.2. Регресија

Регресија у машинском учењу представља технику коришћену за прогнозирање нумеричке вриједности на основу постојећих података. Регресијско учење користи функцију која повезује улазне вриједности са циљном вриједношћу како би се добиле

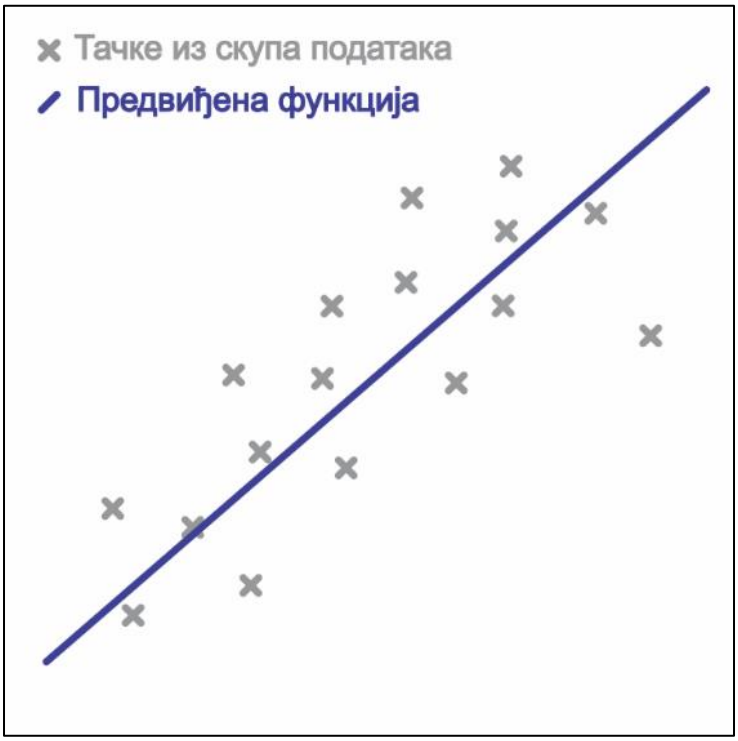
прогнозе за нове примјере. Регресија се користи за рјешавање проблема предвиђања нумеричке вриједности на основу неколико улазних промјењивих.

У регресији, „казна“ за нетачно предвиђање зависи од величине разлике између истинске и предвиђене вриједности, за разлику од проблема класификације, гдје обично не постоји појам блискости између различитих категорија. [4]

Коришћењем регресије, модел може да стекне увид у релације између улазних вриједности и циљне вриједности, као и да научи да процијени циљну вриједност на основу нових улазних вриједности.

Постоји више врста регресионих модела, укључујући линеарну регресију, полиномијалну регресију и регресију са сигмоидном функцијом. Коришћењем различитих модела, може се доћи до различитих резултата, што захтијева одабир модела који најбоље одговара задатом проблему.

Регресија се може користити за предвиђање различитих вриједности, укључујући количину новца коју ће неко потрошити на куповину производа, цијену некретнине, предвиђање вриједности залиха, варијацију економских промјењивих.



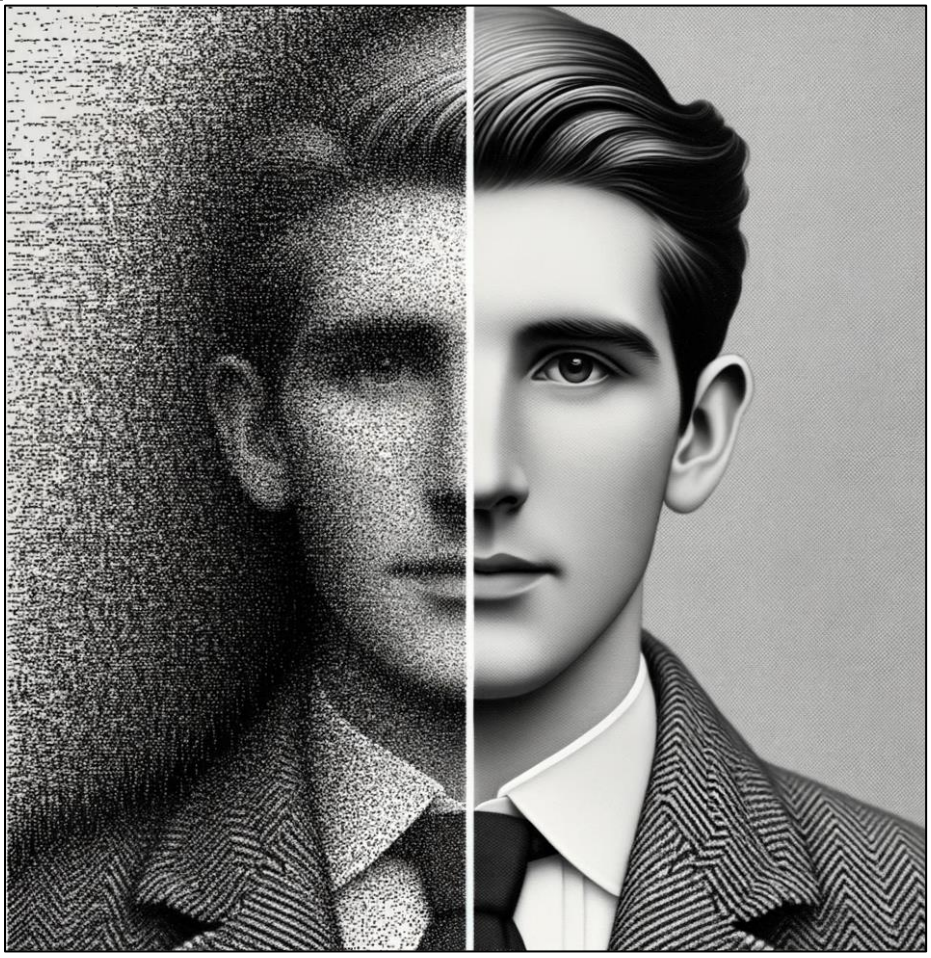
Слика 2.9. Примјер линеарне регресије

2.2. Ненадгледано учење

Ненадгледано учење је врста машинског учења код кога алгоритам учења искључиво прима необиљежене податке за учење и даје предвиђања за све невиђене тачке. Алгоритам не прима унапријед дефинисане ознаке или категорије података, већ умјесто тога, покушава сам пронаћи структуру у подацима.

Будући да уопште нема доступних означених примјера, може бити тешко квантитативно проценити перформансе алгоритма. Груписање и смањење

димензионалности су примјери проблема код којих се ненагледано учење показало као најбоље. [4]



Слика 2.10. Примјер уклањања шума са фотографије помоћу аутоенкодерске неуронске мреже генерисан помоћу DALE-E алата компаније OpenAI[10]

У ненадгледаном учењу, систем користи алгоритме попут кластерисања како би груписао сличне податке и створио категорије. Овакав приступ је користан када имамо велики број података са сложеном структуром, а не знамо које су категорије важне за дати проблем.

Постоје два главна подтипа ненадгледаног учења: груписање и смањење димензионалности. Груписање покушава груписати сличне податке у исте кластере како би се створиле категорије. Смањење димензионалности покушава смањити број димензија података како би се добила слика о главним вриједностима и структури података.

Ненадгледано учење често се користи у апликацијама попут сегментације тржишта, анализе купаца и анализе података. Међутим, његове резултате треба пажљиво интерпретирати и проценити њихову тачност јер недостају унапријед дефинисане ознаке за провјеру резултата.

[10] https://labs.openai.com/

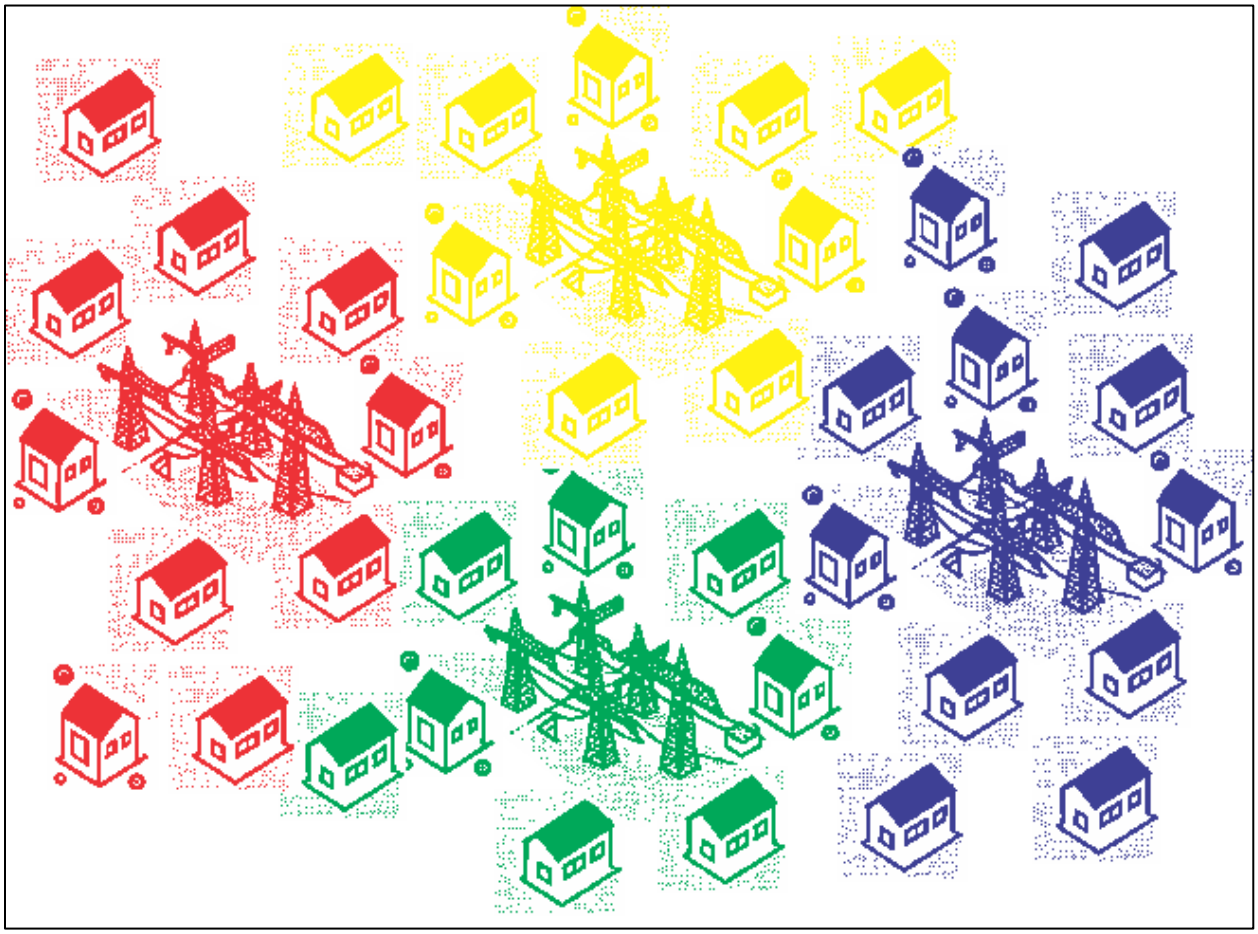
2.2.1. Груписање

Груписање или кластеринг (*енг. Clustering)* у машинском учењу је техника сврставања сличних објеката у исте групе, тзв. кластере. Овај приступ се користи за организовање великих количина необрађених података у корисне групе које имају сличне карактеристике.

Груписање се описује као једна од метода за апстраховање информација из података без да се користе подаци о ознакама објеката. Груписање се може провести коришћењем различитих алгоритама, као што су хијерархијско груписање *(енг. Hierarchical clustering),* груписање методом к-средњих вриједности *(енг. K-means clustering),* итд.

Груписање се описује као проблем ненадгледаног учења, што значи да алгоритам ради без предефинисаног означавања података. Умјесто тога, алгоритам тражи структуру у подацима и сам саставља групе. може се мјерити различитим метрикама, попут еуклидске удаљености или других.

У сваком случају, груписање је важна техника која се често користи у обради података прије даљег машинског учења, помажући у смањењу димензије података и учинковитијем моделирању. [5]



Слика 2.11. Једноставан примјер груписања домаћинстав према удаљености од електричног разводника

2.2.2. Смањење димензионалности

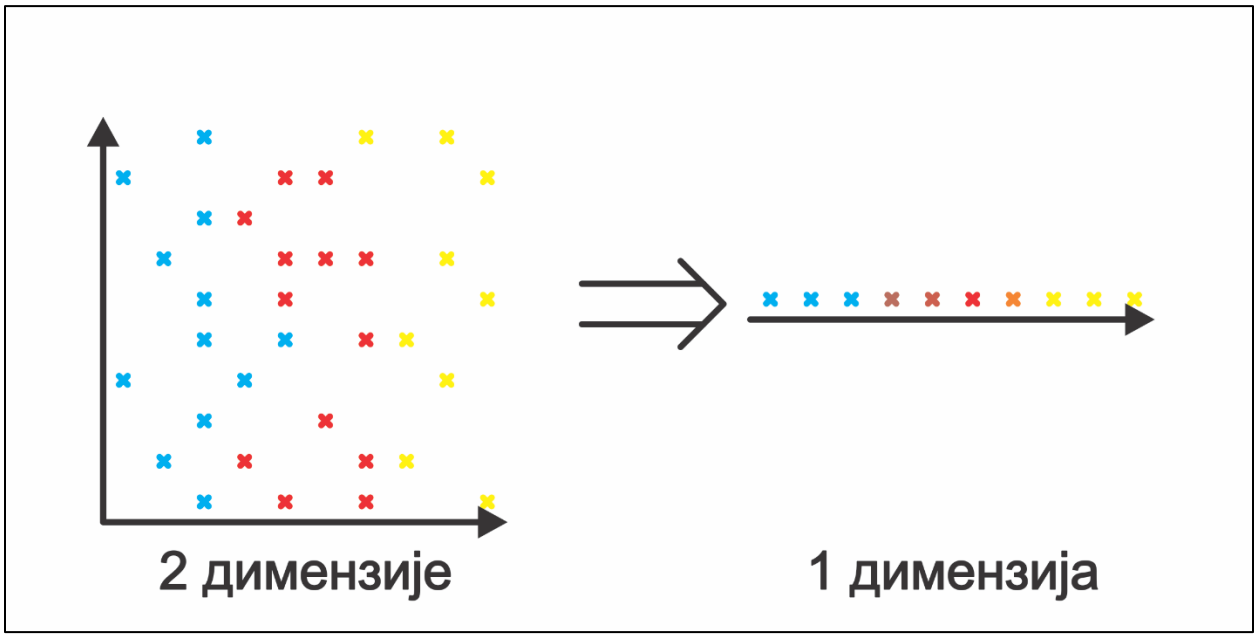
Смањење димензионалности је техника у машинском учењу која се користи за смањивање броја атрибута у подацима. Ова техника се користи како би се смањила сложеност података и убрзао процес машинског учења. Смањење димензионалности се често користи у комбинацији са другим техникама машинског учења, како би се створили бољи модели. Смањење димензионалности може да буде од помоћи и за избјегавање преприлагођавања.

Различите технике смањења димензионалности укључују анализу главних компоненти *(енг. Principal component analysis - PCA*), линеарну дискриминантну анализу *(енг. Linear Discriminant Analysis – LDA)* и мултидимензионално скалирање *(енг. Multidimensional Scaling - MDS)*. Свака од ових техника се користи за различите ситуације и има своје предности и недостатке.

Редукција димензионалности није рјешење за све проблеме у машинском учењу и треба се користити са опрезом како се не би изгубиле важне информације. [5]

Алгоритми који спадају у ову врсту учења су :

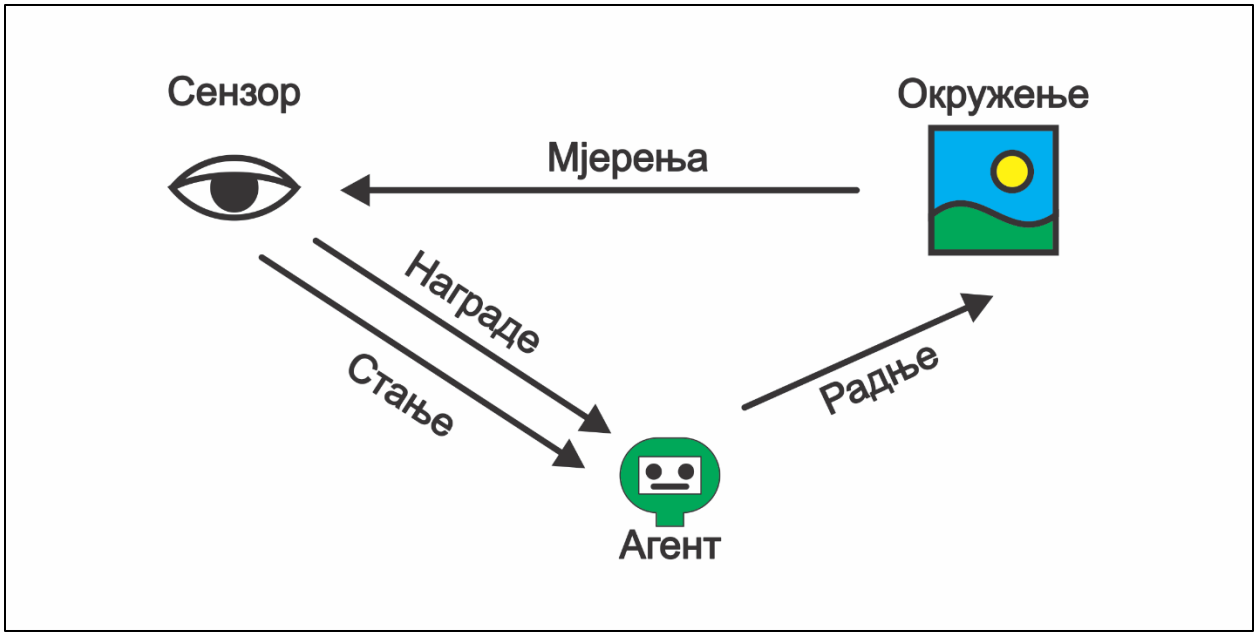
1. K Means
2. PCA (енг. Principal Component Analysis)
3. t-SNE (енг. t-Distributed Stochastic Neighbor Embedding)
4. Правило удруживања (енг. Association rule)



Слика 2.12. Примјер смањења димензионалности ненадгледаним учењем

2.2.3. Учење уз подстицај

Учење уз подстицај *(енг. Reinforcement learning)* је врста машинског учења код кога су фазе обуке и тестирања измијешане у процесу подстицања. Да би прикупио информације, алгоритам учења активно комуницира са окружењем и, у неким случајевима, утиче на окружење и за сваку радњу прима тренутну „награду“ у облику неке нумеричке вриједности, што можемо видјети на *слици 2.13.*

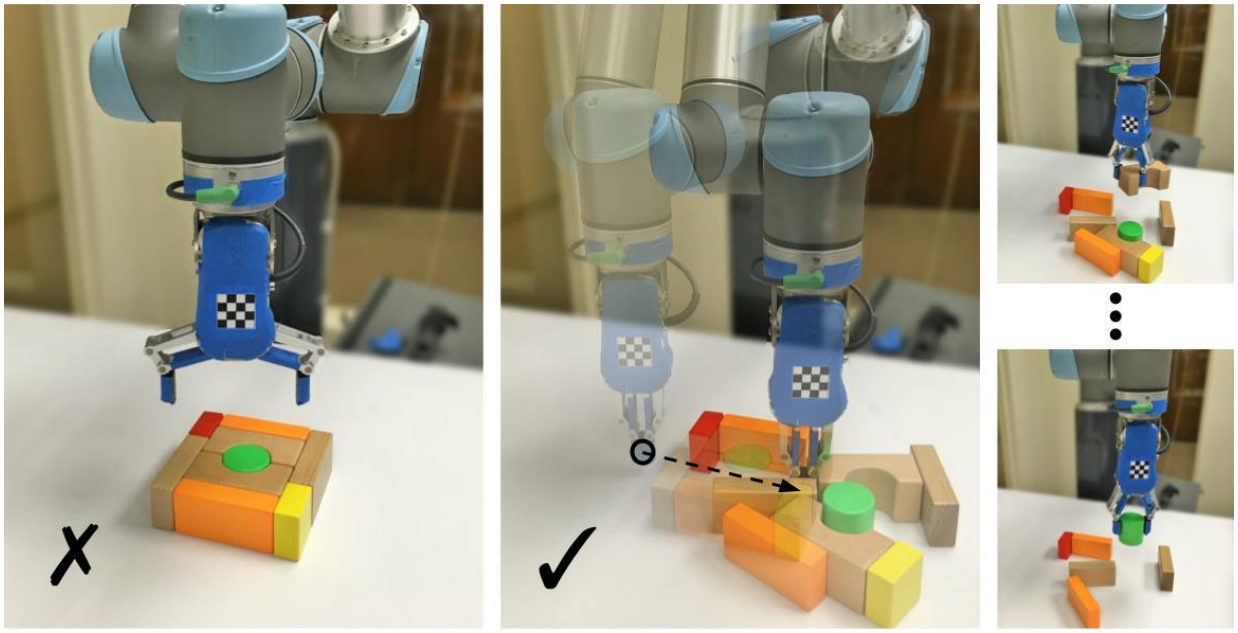


Слика 2.13. Илустрација агента који учи уз подстицаје

Циљ алгоритма учења је максимизирање његове награде током процеса подстицања, међутим, окружење не пружа повратне информације о дугорочним наградама, а алгоритам учења се суочава са дилемом истраживање или експлоатација, јер мора да бира између истраживања непознатих радњи (како би стекао више информација) и искоришћавања већ прикупљених информација. [4]

Другим ријечима, учење уз подстицај је врста учења у којој модел учи кроз искуства. Он понавља неке активности у окружењу и на основу својих поступака добија награду (подстицај) или казну. Циљ овог учења је пронаћи оптималне поступке којима се постиже највећа награда.

Учење уз подстицај се заснива на идеји да модел представља неког агента у окружењу које садржи неке ресурсе и препреке. Агент се креће кроз окружење и на основу свог понашања добија одређени подстицај или казну. Задатак агента је да пронађе оптималан низ корака који му доноси највећи подстицај.



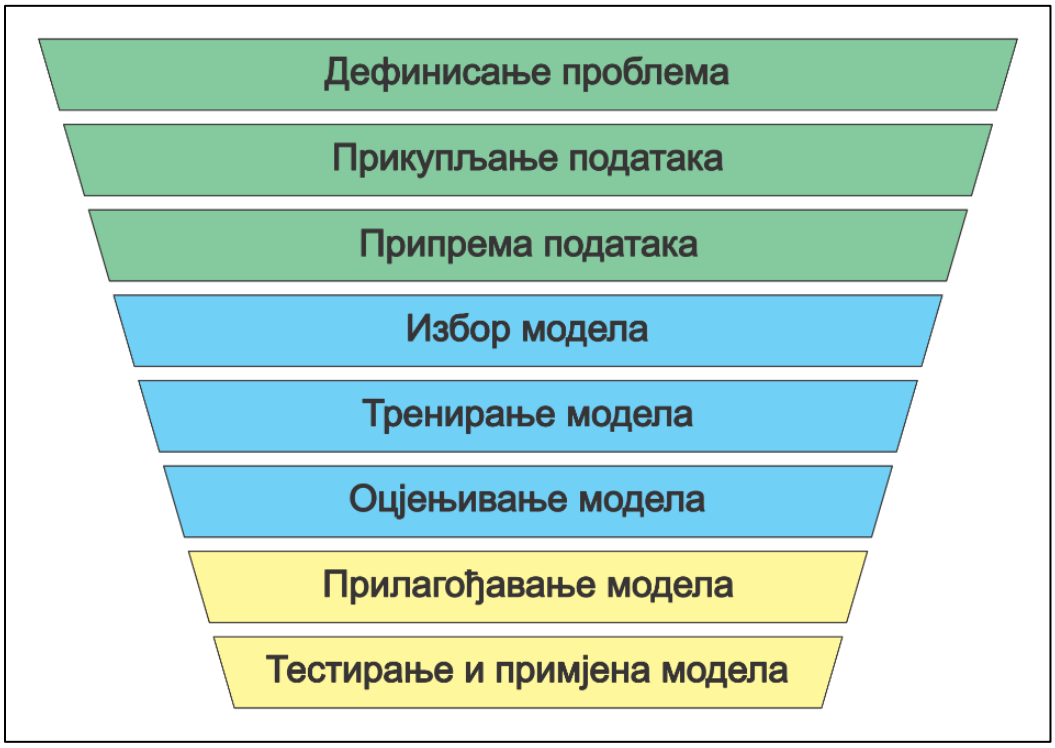
Слика 2.14. Примјер роботске руке која учи уз подстицаје да подигне зелени предмет[11]

Ова врста учења се разликује од других врста машинског учења јер не захтијева да се унапријед дефинишу правила за рјешавање проблема. Умјесто тога, модел учи кроз интеракцију са својим окружењем и на основу својих искустава. Ова врста учења се често користи у проблемима гдје није унапријед познато како се неки проблем може формално дефинисати, као што је случај са играма, управљањем енергијом и управљањем саобраћајем. Учење уз подстицај се имплементира коришћењем алгоритама Q-учења у којима се користе функције награде и Q-вриједности за дефинисање понашања агента у окружењу. [5]

[11] https://vpg.cs.princeton.edu/

3. Процес машинског учења

Процес машинског учења почиње дефинисањем проблема. Послије тога, настављамо са прикупљањем података, који су најчешће велики број мјерења неке појаве (са великим бројем мјерених вриједности). Затим, припремамо податке тако што их чистимо, нормализујемо и претварамо у облик који се може користити у учењу. Наредни корак је одабир модела, што се може урадити уз помоћ литературе, искуства и испробавања различитих модела. Модел се затим тренира на припремљеним подацима коришћењем одређене оптимизационе функције. Након што је модел трениран, важно је да се оцијени квалитет модела, коришћењем метрика попут прецизности и грешке учења. Ако је потребно, може се урадити и подешавање хиперпараметара модела. Када се модел доведе до жељених перформанси, он се користи у стварном свијету како би се предвидјели нови случајеви на којима модел није трениран. [6] На *слици 3.1.* се налазе фазе процеса машинског учења.



Слика 3.1. Фазе процеса машинског учења

3.1. Дефинисање проблема

Фаза дефинисања проблема у машинском учењу подразумијева јасно одређивање проблема који се жели ријешити и коришћење алата и техника машинског учења за рјешавање тог проблема. Овај процес укључује утврђивање вриједности које се жели предвидјети, као и одређивање да ли се ради о проблему класификације или регресије. У проблемима класификације, излазна вриједност може бити ознака категорије, док је у проблемима регресије најчешће нека бројна вриједност.

Без јасне дефиниције проблема, модел се може тренирати на погрешан начин, што може довести до недостатка жељених резултата. Стога је кључно прецизно одредити проблем и одговарајући приступ у рјешавању проблема, како би се постигао жељени исход. [7]

3.2. Прикупљање података

Фаза прикупљања података у машинском учењу представља процес узимања података из различитих извора, њихову идентификацију и сакупљање у јединствену колекцију података који ће се користити за тренирање и тестирање машинских модела. Подаци морају бити релевантни за проблем који се покушава ријешити и требали би бити квалитетни и служити као представник скупа података из стварног свијета. Велика пажња се посвећује прикупљању квалитетних података, јер лоши подаци доводе до лоших модела и неадекватних резултата.

Подаци могу бити прикупљени из једног извора података или из више извора (преуимањем, екстракцијом из база података, интернет странице). На интернету постоји велики број бесплатних и јавно доступних колекција података. Те колекције података су најчешће унапријед означене и припремљене што олакшава ову фазу машинског учења.

Приликом прикупљања података, потребно је узети у обзир и репрезентативност података и величину података. Потребно је идентификовати и обрадити изворе података, укључујући базе података, текстуалне документе и друге неформализоване изворе. Важно је такође размотрити проблеме као што су неконзистентност и нерепрезентативност података и њихов утицај на перформансе модела.

Потребно је пажљиво одабрати и прикупити податке који ће бити репрезентативни за проблем који се покушава ријешити како би се створио поуздан модел са високим перформансама.[8].

3.3. Припрема података

Фаза припреме података укључује чишћење, трансформацију и припрему података за обраду и тренирање модела.

Једна од најважнијих активности у припреми података је уклањање недостатака и неконзистентности у подацима. То може укључивати исправљање грешака у уносу података, уклањање дуплих записа и неважећих вриједности.

Трансформација података укључује обраду података тако да се припреме за тренирање модела. То може укључивати нормализацију података, кодирање категоризацијских промјенљивих и издвајање нових атрибута из постојећих података.

Затим, подаци се морају припремити за тренирање модела. То укључује раздвајање података у скупове за тренирање и тестирање, како би се модел тренирао на једном скупу, а потом провјерио његову перформансу на другом скупу.

Без добро припремљених података, модел може бити неуспјешан у њиховој предикцији и генерализацији на нове податке.

3.4. Избор модела

Избор модела у машинском учењу подразумијева одабир најприкладнијег алгоритма и имплементацију тог алгоритма у виду модела. Одабир модела се обично темељи на неколико фактора, укључујући тип проблема који се жели ријешити, квалитет и количину прикупљених података и ресурсе који су на располагању.

Избор модела се објашњава кроз неколико фаза. Прва ствар коју треба учинити је дефинисати проблем који се жели ријешити. [7] Ова дефиниција омогућава одабир адекватног типа модела. Након што се дефинише проблем, сљедећи корак је преглед релевантних модела и препорука од стране стручњака у пољу.

Након што се дефинише проблем и препоручи неколико модела, сљедећи корак је евалуација ових модела у складу са дефинисаним критеријумима, као што су прецизност, брзина тренирања и примјене, стабилност, робустност, интерпретабилност и сл. Након што се дефинишу критеријуми, сљедећи корак је стварни избор модела уз помоћ приступа који укључује, одабир најбољег модела уз помоћ одабира на основу испитивања, одабир модела на основу стручног знања или мјешавине тих приступа.

Погрешан избор модела може резултирати неадекватним перформансама. Због тога је важно да се врши евалуација различитих модела како би се одабрао најбољи, коришћењем метода попут крос-валидације и евалуације перформанси на тестном скупу података.

Неки од фактора које треба узети у обзир при одабиру модела су врста проблема, квалитет података и ресурси на располагању.

3.5. Тренирање модела

Тренирање модела у машинском учењу представља процес у коме се моделу дају подаци како би се научио како да предвиди резултате за нове примјере. Тренинг модела се може описати као процес оптимизације функције губитка (енг. *Loss Function*) или функције циља (енг. *Objective function*). [7] Функција губитка одређује колико је добар модел у предвиђању, а функција циља се користи за оптимизацију модела у смислу да се максимизије та функција.

Процес тренирања се обично спроводи коришћењем алгоритама за оптимизацију као што су градијентно спуштање (енг. *Gradient Descent*), стохастичко градијентно спуштање и други. Модел се тренира користећи тренинг податке, а затим се оцјењује користећи валидационе податке.

Након што је модел трениран, вриједност функције губитка се смањује, што значи да модел постаје бољи у предвиђању. Укупан број епоха кроз које се тренира модел може варирати, а одлука о томе када прекинути тренинг зависи од многих фактора, као што су сложеност модела, величина скупа података, брзина оптимизације итд.

Процес тренирања модела није увијек једноставан и може захтјевати велике ресурсе, посебно ако се користе комплексни модели или велики скупови података, па је важно добро искористити га на одговарајући начин, како би се постигла адекватна предикција и генерализација модела на нове податке.

3.6. Оцјењивање модела

Оцјена перформанси модела се користи како би се процијенила његова способност предвиђања, што омогућава праћење и тестирање квалитета модела.

Постоје различите методе за оцјењивање модела, укључујући унакрсну провјеру (енг. *Cross-validation*), провјеру са непознатим скупом података (енг. *Holdout Validation*) и процјену грешке (енг. *Error Estimation*). Ове процедуре оцјењивања користе се за одређивање прецизности модела, капацитета генерализације и сличних мјера квалитете.

Постоје различити критеријуми за оцјењивање модела, укључујући прецизност (енг. *Accuracy*), прецизност учења (енг. *Learning Accuracy*), прецизност генерализације (енг. *Generalization Accuracy*), F1 скор и ROC криву. Ови критеријуми помажу у одређивању на које метрике се потребно фокусирати када се процјењују модели.

Важно је напоменути да коришћење ових критеријума зависи од конкретне ситуације у којој се модел примјењује и циљевима које се желе постићи, па је важно разумјети њихове предности и недостатке при одабиру метрика за оцјењивање модела, јер могу да утичу на перформансе модела.

3.7. Прилагођавање модела

Прилагођавање (адаптација) модела у машинском учењу представља процес подешавања модела који се користи за објашњавање или предвиђање података. Циљ прилагођавања је побољшање перформанси модела тако што ће се оптимизовати његови параметри.

Нпр. ако се модел обучава на скупу података са сликама мачака и паса, а потом се примјењује на скупу података са сликама дивљих животиња, модел ће се вјероватно лоше понашати јер се структура и дистрибуција података разликују. У том случају, прилагођавање модела би се могло провести додавањем нових примјера из новог скупа података у оригинални скуп података за обуку модела или подешавањем постојећих параметара модела како би се побољшала његова способност за препознавање нових примјера.

Постоје два основна начина за прилагођавање модела: надгледано и ненадгледано учење. Надгледано учење користи означене податке с жељеним излазима и користи их за изградњу модела који се потом оптимизује користећи алгоритме који подешавају параметре модела на најбољи начин. Ненадгледано учење користи податке који нису означени с жељеним излазима и користи их за идентификовање структура у подацима. Ова метода користи алгоритме који групишу податке и оптимизују параметре модела како би се добила што боља груписања.

Постоје разне методе оптимизације које се користе за прилагођавање модела, укључујући градијентни спуст, квази-Њутн методе и Марков-Ланов процес. Прилагођавање модела може бити компликовано и захтјевно, па често захтијева интеракцију стручњака за машинско учење с експертима у конкретном домену како би се постигли најбољи резултати.

3.8. Тестирање и примјена модела

Процес тестирања и примјене модела у машинском учењу представља посљедњи корак у изради модела. Циљ је провјерити колико добро модел ради на независном скупу података који није кориштен за обуку. Ако се модел показао прецизним на обучавајућим подацима, то не значи да ће бити прецизан и на новим, независним подацима.

Након што се модел тестира и процијени, потребно је провјерити колико је он генералишући и способан радити на цијелом скупу података. Ако се модел покаже неадекватним, потребно га је прилагодити и поново тестирати док се не постигне жељени ниво тачности.

Након што се постигне жељени ниво тачности, модел се може примијенити на стварним проблемима и користити за доношење одлука или предвиђање резултата. Међутим, ако се модел користи у стварној примјени, неопходно је редовно пратити његову прецизност и прилагођавати га како би се осигурало његово постојано постизање прецизности.

4. Избор алгоритма

Избор алгоритма у машинском учењу одређује квалитет модела и његову ефикасност. Различити фактори играју кључну улогу у избору алгоритма:

- Природа података: карактеристике података, као што су величина и врста, могу утицати на избор алгоритма. На примјер, ако су подаци дискретни, неки алгоритми као што су дрво одлучивања могу бити погоднији од алгоритама заснованих на регресији.
- Циљ проблема: циљ проблема, као што је класификација, регресија или кластерисање, такође може утицати на избор алгоритма.
- Квалитет података: квалитет података, укључујући присуство непотпуних и неодређених података, може утицати на избор алгоритма.
- Перформансе: треба узети у обзир перформансе алгоритма у погледу брзине и прецизности.
- Комплексност модела: такође је важно размотрити комплексност модела и способност тумачења модела.

Избор алгоритма захтијева балансирање између перформанси, комплексности и интерпретабилности, у зависности од специфичности проблема и природе података. Коришћењем тих фактора, може се изабрати најбољи алгоритам за дати проблем.

4.1. Логистичка регресија

Логистичка регресија (енг. *Logistic Regression*) је статистички алат који има за циљ да моделира биномни резултат с једном или више објашњивих промјењивих. У машинском учењу, овај алат често се користи за рјешавање проблема, у којима је циљ да се предвиди припадност објеката некој од двије класе. Основа алгоритма је линеарна регресија, али умјесто да се користи континуална зависна промјењива, логистичка регресија користи логистичку функцију (енг. *Logistic Function*) како би се моделирао однос између и бинарне зависне варијабле. Логистичка функција враћа излаз између 0 и 1 који се може интерпретирати као вјероватноћу припадности објекта једној од двије класе.

У процесу тренирања, алгоритам користи градијентни спуст (енг. *Gradient Descent*) или неку другу оптимизацијску технику како би се пронашли оптимални коефицијенти у логистичкој функцији. Ова оптимизација се врши кроз итеративне кораке у којима се рачунају градијенти функције губитка (енг. *Loss Function*) и користећи их се ажурирају коефицијенти у логистичкој функцији.

Након што се алгоритам тренира, може се користити за предвиђање припадности нових објеката једној од двије класе на основу вриједности њихових значајки. У овом случају, логистичка функција се користи за израчунавање вјероватноће припадности објекта некој класи, а коначна класификација се врши тако што се вјероватноћа пореди са неким прагом (енг. *Threshold*), обично 0.5.

Укупно, логистичка регресија се сматра једноставним и ефикасним алгоритмом у многим класификацијским проблемима. Међутим, његова ефикасност зависи од многих фактора, укључујући квалитет значајки, величину скупа података и врсту проблем

4.2. Gaussian Naive Byes Classifier

Gaussian Naive Byes Classifier је алгоритам класификације базиран на Бајесовој теореми и претпоставци да свака карактеристика у подацима има нормалну расподјелу. Алгоритам се користи за класификацију у којима имамо н-категорија и н-карактеристике за сваку инстанцу у подацима.

Процес функционисања Gaussian Naive Byes Classifier алгоритма почиње са рачунањем Gaussian Probability Desity Function (PDF) за сваку карактеристику у свакој категорији. Овај PDF описује како се карактеристике расподељују у датој категорији. Након тога, Бајесова теорема се користи за рачунање вјеројатноће за сваку категорију за дату инстанцу података. Вјеројатноће се комбинују за све карактеристике у датој инстанци и на крају се израчунава коначна вјеројатност за сваку категорију. Инстанца се класификује у категорију са највећом вјеројатношћу.

Међутим, претпоставка да свака карактеристика има нормалну расподјелу често може бити погрешна у стварним подацима, што може довести до лоших резултата класификације, па је важно провјеравати и примјењивати адекватне претпоставке о расподјели када се користи овај алгоритам.

Gaussian Naive Byes Classifier је једноставан и ефикасан алгоритам за класификацију, а посебно се добро показује у случајевима са великим бројем категорија и малим бројем карактеристика.

4.3. K Nearest Neighbors Classifier

K Nearest Neighbors Classifier (KNN) алгоритам је један од најједноставнијих и најчешће кориштених алгоритама у машинском учењу, коришћен за класификацију. Овај алгоритам користи концепт к-најближих сусједа за класификацију нових објеката.

Алгоритам KNN представља једноставну верзију алгоритма класификације, гдје се узима у обзир к-најближих сусједа објекта који се покушава класификовати.

Одређивање к-најближих сусједа се врши на основу израчунате удаљености између новог објекта и свих објеката из тренинг скупа. Након тога, објекти са најмањом удаљеношћу су дефинисани као к-најближи сусједи.

На крају, класификација новог објекта се врши преко гласања к-најближих сусједа, тако што се узима у обзир најчешћа класа међу њима.

У поређењу са другим алгоритмима, KNN је једноставан за примену, а истовремено ефикасан у класификацији различитих типова података. Међутим, један од недостатака овог алгоритма је што се рачунају удаљености између свих објеката у тренинг скупу и новог објекта, што може бити захтјевно за израчунавање за велике податке.

4.4. Decision Tree Classifier

Decision Tree Classifier алгоритам се користи за разврставање података. Овај алгоритам користи стабло одлучивања као своју главну структуру, гдје се разматрају различити атрибути унутар скупа података и доносе се одлуке о томе како да се подаци разврстају.

Стабло одлучивања користи двије врсте чворова: чвор са члановима и лист. Чвор са члановима разматра неки атрибут и одлучује да ли подаци треба да иду у један од његових

подчворова, који се затим поново разматрају. Овај процес тумачења се понавља све док се не дође до листа, који означава коначно мјесто код разврставања података.

Decision Tree Classifier алгоритам се може користити за разврставање података у више класа, као и за класификацију података. Алгоритам функционише тако што се користе претходни подаци како би се створило стабло одлучивања, које се онда користи да се разврстају нови подаци.

Овај алгоритам је једноставан за разумијевање и имплементирање, што га чини идеалним за многе примјене укључујући финансије, трговину, медицинске и друге индустрије. Међутим, постоје неки недостаци као што склоност преприлагођавању и потреба за одређивањем границе када се разматрају атрибути.

4.5. Random Forest Classifier

Random Forest Classifier је алгоритам који се користи и за класификацију и за регресију. Овај алгоритам користи много одвојених стабала одлучивања (насумичну шуму стабала одлучивања) за добијање закључка. На почетку, користи се метода случајног узорковања да се формира више одвојених стабала одлучивања. Свако од њих даје свој закључак на основу своје процене, а затим се користи гласање да се добије финална процена.

У односу на класично стабло одлучивања, рандом форест користи више стабала одлучивања које су створене користећи случајни избор атрибута и података. То доводи до више робустности и мање вјероватноће оверфиттинг-а. Такође, овај алгоритам има бољу перформансу у случајевима када има много атрибута, што често може довести до проблема у класичном стаблу одлучивања.

Коришћење рандом форест алгоритма може се реализовати за различите типове проблема, укључујући класификацију, регресију и идентификацију најважнијих атрибута. Укупно, рандом форест је један од најчешће коришћених алгоритама у машинском учењу због своје робустности и широке примене у различитим областима.

4.6. Gradient Boosting Classifier

Gradient Boosting Classifier је алгоритам за машинско учење који се користи за класификацију. Овај алгоритам је комбинација неколико класификатора, а сваки од њих је научен да поправља грешке претходних класификатора. Користи технику бустинг (енф. *Boosting)* за побољшање перформанси класификације. Ова техника укључује додавање нових класификатора док се не постигне жељени ниво тачности. Нови класификатори се додају на бази претходно направљених грешака у нади да ће се оне исправити.

Gradient Boosting Classifier такође користи технику Gradient Descent за одређивање правца у ком ће се класификатори обучавати. Gradient Descent користи функцију губитка (енг. *Loss Function*) да би се утврдила удаљеност између тренутног стања и жељеног стања. Ова функција губитка се користи да би се пронашло оптимално рјешење, које ће смањити укупани број грешака.

Овај алгоритам обично даје боље резултате од других, једноставнијих класификатора, попут логистичке регресије или наивног Бајес класификатора. Међутим, Gradient Boosting Classifier је такође склон преприлагођавању, па је потребно пронаћи оптималну комбинацију броја класификатора и хиперпараметара.

4.7. Support Vector Machine Classifier

Support Vector Machine (SVM) Classifier је алгоритам машинског учења који користи математичке моделе како би предвидио категорију новог примјера. SVM користи теорију оптимизације да би формирао једну или више хиперсуперфиција (тј. линија или хипер равни) које најбоље раздвајају примјере у скупу података у одговарајуће категорије.

Приликом формирања хиперсуперфиција, циља на то да пронађе хиперсуперфицију која је најудаљенија од најближих примјера из сваке категорије, што се назива маргином. Овај приступ омогућава SVM -у да створи робустније и генерализоване моделе, што смањује ризик од преувеличавања (тј. претренираности) на скупу података за тренирање.

SVM има два главна мода рада: Линеарни SVM који користи линеарне хиперсуперфиције за раздвајање примјера у категорије, и нелинеарни SVM који користи нелинеарне хиперсуперфиције. Нелинеарни SVM се може користити у ситуацијама када линеарни SVM не може довољно добро да раздвоји примјере у категорије. такође подржава и технике регуларизације, што га чини погодним за рад са великим скуповима података и тешким проблемима класификације. Овај алгоритам се често користи за анализу текста, слика и геномских података.

4.8. LightGBM Classifier

LightGBM је алгоритам за машинско учење који се користи за класификацију и регресију. Развијен је као оптимизована верзија Gradient Boosting Machine (GBM) алгоритма, користећи технике које убрзавају процес тренинга. LightGBM користи технологију усмереног успоравања учења, што значи да се фокусира на тешке примјере у сваком кораку учења, како би се што брже извршио процес учења.

Он се разликује од стандардних GBM алгоритама по томе што користи хистограм за распођелу особина у сваком чвору. Овај приступ значајно смањује потребан број итерација у процесу учења, а такође олакшава и убрзава приступ подацима. Такође, користи бинарно претраживање, што даље смањује вријеме потребно за извршење процеса учења.Овај алгоритам има способност учења на различитим типовима података, укључујући и велике податке и скупове података са спојеним подацима. Такође, он има добре перформансе на проблемима са небалансираним подацима, када има више примјера једне класе него друге.

LightGBM је ефикасан и брз алгоритам за машинско учење, посебно за проблем класификације и регресије, а његова употреба се све више шири у индустрији и истраживачким заједницама.

4.9. K Nearest Neighbors Regressor

K Nearest Neighbors Regressor алгоритам је тип регресијског модела који се заснива на принципу к најближих сусједа. Овај алгоритам се често користи за предвиђање континуалних вриједности, попут цијене некретнине или просјечне мјесечне температуре. Алгоритам функционше тако што се за сваку инстанцу у скупу података за учење проучава н најближих сусједа и израчунава се просјек вриједности промјењиве циља за те сусједе. На крају се добијени просјек користи као предвиђена вриједност за дату инстанцу.

Кључни параметар овог алгоритма је број најближих сусједа, што се може одабрати помоћу валидацијске методе, као што је крос-валидација. Такође, укључивање тежина за сусједе такође може утицати на квалитету предвиђања, гдје су сусједи који су ближе инстанци за предикцију тежи.

Међутим, овај алгоритам има и неколико недостатака, укључујући слабе перформансе у случајевима гдје постоји много димензија и висок ступањ интеракције међу промјењивима, што може резултирати проблемима с пренаучености.

Укратко, К Неарест Неигхборс Регрессор алгоритам је једноставан и брз за тренирање те често даје добре резултате у случајевима гдје су подаци хомогени и дистрибуција варијабле циља није превише сложена. Међутим, потребно је пажљиво одабрати параметре и провјерити перформансу модела на скупу података за тестирање како би се избјегли проблеми са пренаучењем.

4.10. LightGBM Regressor

LightGBM је врста регресијског алгоритма у машинском учењу који користи технику усмјерену према градијентном боостингу. Главна разлика између LightGBM и других алгоритама градијентног боостинга је у томе што користи двоструко бинарно пођелу за брзо рјешавање проблема скалирања.

У раду LightGBM-а, стабла одлучивања се користе као главни модели за регресију. Они се настављају једно за другим у фазама, у свакој фази користећи информације из претходне фазе за побољшање тачности.

Код LightGBM-а, постоје два главна параметра за оптимизацију: број стабала одлучивања и максимална дубина сваког стабла. Број стабала одлучивања одређује колико ће модела бити створено, док максимална дубина сваког стабла одређује колико ће комплексан бити сваки модел.

LightGBM такође користи технику смањења димензионалности како би се избјегло превише сложености и оверфиттинг-а. То се постиже одабиром најзначајнијих значајки које ће се користити за сваки модел.

LightGBM је популаран због своје брзине и ефикасности, посебно када се ради са великим скуповима података. Он је такође способан радити с различитим врстама података, укључујући и податке с различитим скалама и категоричке податке. Укупно, LightGBM је један од најбољих алгоритама регресије у машинском учењу, и користи се у многим реал-њорлд примјенама, укључујући предвиђање цијена, кредитне процјене и персонализацију препорука.

5. Практични рад

Задатак: Анализирати моделе за предвиђање броја особа у просторији креиране коришћењем неколико алгоритама (класификационих и регресионих). Моделе је потребно тренирати подацима који садрже температуру, влажност, ниво угљен-диоксида и др. За реализацију користити Jupyter Notebook алат. Извршити компаративну анализу перформанси добијених модела.

Jupyter Notebook је интерактивно окружење за писање и извршавање кода које је првобитно развијено за подршку језицима као што су Julia, Python и R, али сада подржава многе друге језике. Карактерише га способност комбиновања кода, текста, графике и формула у једном документу.

Документ се састоји од ћелија које могу садржавати различите типове садржаја и најчешће се инсталира кроз дистрибуцију Anaconda, која садржи многе научне библиотеке за Python. Поред класичног Jupyter Notebook-a, постоји и JupyterLab, који је сљедећа генерација Jupyter интерфејса и нуди проширене могућности за рад са више докумената и бољу интеграцију с другим алатима.

Због своје интерактивности и флексибилности, Jupyter Notebook је постао стандардни алат у областима анализе података и машинског учења. Постоји и могућност извоза Notebook докумената у разне формате, што олакшава дијељење и презентацију резултата.

У току развоја рјешења смо користили сљедеће параметре да означимо да ли желимо да се исцртавају или исписују одређени излази. Такође, имамо и праг за прорачун тачности модела.

Parameters:

```
THRESHOLD = 1       #THRESHOLD = 0 gives the same values as model.score()
DRAW = False        #Are we going to draw confusion matrices
PRINT = False       #Are we going to print out parameters
```

Слика 5.1. Дефинисање параметара

5.1. Помоћне функције

Како бисмо што читкије ријешили проблем, дефинисали смо три помоћне функције. Прва фукција је кориштена за рачунање тачности модела. Друга функција је за исцртавање матрице конфузије. Посљедња фунција моји смо морали да креирамо је функција за тренирање и анализу модела.

Helper functions:

```
def calculate_accuracy(y_test, y_pred, threshold):
    count = 0
    correct = 0
    y_test = y_test.tolist()
    y_pred = y_pred.tolist()
    for i, line in enumerate(y_test):
        real = float(y_test[i])
        pred = float(y_pred[i])
        if(abs(real - pred) <= threshold):
            correct = correct + 1
        count = count + 1
    return float(correct)/count
def draw_confusion_matrix(X_test, y_test, y_pred):
    y_test = y_test.tolist()
    y_pred = y_pred.round(0).astype(float).tolist()
    cm = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.imshow(cm)
    ax.grid(False)
    ax.set_xlabel('Predicted outputs', color='black')
    ax.set_ylabel('Actual outputs', color='black')
    ax.xaxis.set(ticks=range(10))
    ax.yaxis.set(ticks=range(10))
    ax.set_ylim(9.5, -0.5)
    for i in range(10):
        for j in range(10):
            ax.text(j, i, cm[i, j], ha='center', va='center', color='white')
    plt.show()
```

Слика 5.2. Помоћне функције за рачунање прецизности и исцртавање матрице

```
def fit_and_analise(model, X, X1, y_train, y_test1, name):
    start_time = time.time()

    model.fit(X, y_train)
    y_pred = model.predict(X)
    y_pred1 = model.predict(X1)

    elapsed_time = time.time() - start_time
    train_accuracy = calculate_accuracy(y_pred, y_train, THRESHOLD)
    test_accuracy = calculate_accuracy(y_test1, y_pred1, THRESHOLD)

    if(PRINT == True):
        print(name + ":")
        print(" TIME: " + str(elapsed_time)[:5] + " seconds")
        print("TRAIN: " + str(train_accuracy)[:5] + "%")
        print("TEST: " + str(test_accuracy)[:5] + "%")
    if(DRAW == True):
        draw_confusion_matrix(X, y_train, y_pred)
        draw_confusion_matrix(X1, y_test1, y_pred1)
    return [name, elapsed_time, train_accuracy, test_accuracy]
```

Слика 5.3. Помоћна функција за прилагођавање и анализирање модела

5.2. Прикупљање података

Подаци су преузети са отвореног BitLab репозиторијума који је креиран за мастер рад "CO2 based room occupancy detection : an IoT and machine learning application" Bockstael, Nicolas ; Jadin, Alexandre.

У првој цјелини увозимо све библиотеке које ће бити кориштене у коду:

- **pandas** за манипулацију и анализу података
- **numpy** за математичке операције над подацима
- **matplotlib.pyplot** за визуализацију података
- **time** за мјерење времена извршавања тренирања модела
- **datetime** за рад са датумима и временима
- **classification_report** и **confusion_matrix** за процјену перформанси модела
- **train_test_split** за подјелу скупа података на скупове за тренирање и тестирање
- **LogisticRegression, GaussianNB, KNeighborsClassifier, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, SVC, LGBMClassifier, LGBMRegressor, KNeighborsRegressor** за израду модела за класификацију и регресију на основу одговарајућих алгоритама и техника машинског учења

Data importing

First, I will do all the neccessary imports:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from datetime import datetime
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from lightgbm import LGBMClassifier
from lightgbm import LGBMRegressor
from sklearn.neighbors import KNeighborsRegressor
```

Слика 5.4. Увоз потребних спољашњих библиотека

Даље, учитавамо csv датотеку која садржи скуп података за тренирање и исписуемо првих пет редова скупа података.

Show raw data:

```
rawData = pd.read_csv("data.txt")
rawData.head()
```

Слика 5.6. Учитавање датотеке са подацима

	name	time	app_id	battery	co2	dev_id	hardware_serial	humidity	light	motion	person_count	temperature	time_device
0	co2_2	1.513170e+18	url_test	3631	559	co2_02	A81758FFFE030F06	37	76	1	7	212	2017-12-13T13:06:24.428788324Z
1	co2_2	1.513170e+18	url_test	3628	569	co2_02	A81758FFFE030F06	37	78	3	7	212	2017-12-13T13:07:24.422294045Z
2	co2_2	1.513170e+18	url_test	3628	588	co2_02	A81758FFFE030F06	37	73	1	7	213	2017-12-13T13:08:24.427909805Z
3	co2_2	1.513170e+18	url_test	3631	579	co2_02	A81758FFFE030F06	37	75	2	7	213	2017-12-13T13:09:24.409998484Z
4	co2_2	1.513170e+18	url_test	3631	565	co2_02	A81758FFFE030F06	37	77	1	7	213	2017-12-13T13:10:24.415049931Z

Слика 5.7. Приказ првих пет редова табеле са подацима

5.3. Припрема података

У овој цјелини прво исписујемо типове података за сваку појединачну колону и бирамо који ће подскуп података бити кориштен за генерисање модела. У овом случају бирамо седам атрибута: co2, humidity, light, motion, person_count, temperature, и time_device.

```
print(rawData.dtypes)
```

```
name                object
time               float64
app_id              object
battery              int64
co2                  int64
dev_id              object
hardware_serial     object
humidity             int64
light                int64
motion               int64
person_count         int64
temperature          int64
time_device         object
dtype: object
```

Слика 5.8. Исписивање типова података за колоне скупа података

```
selectedData = rawData[['co2', 'humidity', 'light', 'motion',
                        'person_count', 'temperature', 'time_device']]
```

Слика 5.9. Избор одговарајућих колона у скупу података

Након тога, рачунамо основне статистичке мјере (број, средње вриједности, стандардну девијацију, минимум, максимум, квартиле) за сваку колону у скупу изабраних података, исписујемо облик (димензије) изабраног скупа података, број дупликата у њему и провјеравамо да ли у скупу података имамо null вриједности и исписујемо њихов број.

	co2	humidity	light	motion	person_count	temperature
count	8610.000000	8610.000000	8610.000000	8610.000000	8610.000000	8610.000000
mean	475.983508	34.059466	82.251336	0.533566	3.485947	221.316841
std	73.325450	5.223338	31.062353	0.833635	2.843026	6.112410
min	336.000000	19.000000	1.000000	0.000000	-1.000000	202.000000
25%	417.000000	31.000000	74.000000	0.000000	1.000000	219.000000
50%	464.000000	34.000000	81.000000	0.000000	3.000000	223.000000
75%	531.000000	37.000000	99.000000	1.000000	5.000000	225.000000
max	709.000000	47.000000	167.000000	3.000000	13.000000	233.000000

Number of rows and columns:

```
print(selectedData.shape)
```

```
(8610, 7)
```

Number of duplicate rows:

```
print(selectedData.duplicated().value_counts())
```

```
False    8610
dtype: int64
```

Checking for null values:

```
if(selectedData.isnull().sum().all() > 0):
    print(selectedData.isnull().sum())
else:
    print("No null values!")
```

```
No null values!
```

Слика 5.10. Исписивање основних информација о скупу података

Даље уклањамо све редове у којима је просторија била празна.

I have identified that there are rows where person_count is lower than 0, so we will remove whose rows:

```
cleanedData = selectedData[selectedData['person_count'] >= 0]
```

Слика 5.11. Уклањање редова за случајеве када је просторија била празна

Пошто смо примијетили да датум и вријеме нису у одговарајућем формату, дефинишемо convert_date функцију како би се претворила колона time_device из низа знакова у формат датума и времена, па извршавамо функцију над скупом података и спремамо нове вриједнсти датума и времена у нову колону time_column.

I need to change format of time_device column so it is more suitable for ML:

```
def convert_date(date_time):
    date = date_time.split("T")[0]
    time = date_time.split("T")[1].split("Z")[0][:8]
    return datetime.strptime(date + " " + time, '%Y-%m-%d %H:%M:%S').timestamp()
time_column = cleanedData['time_device'].apply(convert_date)
timedData = cleanedData.drop(['time_device'], axis = 1)
timedData = timedData.join(time_column)
```

```
data = timedData
data.describe()
```

	co2	humidity	light	motion	person_count	temperature	time_device
count	8558.000000	8558.000000	8558.000000	8558.000000	8558.000000	8558.000000	8.558000e+03
mean	475.910493	34.053751	82.236387	0.531433	3.513204	221.381865	1.514822e+09
std	73.281980	5.236035	31.134211	0.832118	2.829997	6.067152	1.270127e+06
min	336.000000	19.000000	1.000000	0.000000	0.000000	202.000000	1.513167e+09
25%	416.000000	31.000000	74.000000	0.000000	1.000000	219.000000	1.513757e+09
50%	464.000000	34.000000	81.000000	0.000000	3.000000	223.000000	1.514536e+09
75%	531.000000	37.000000	99.000000	1.000000	5.000000	225.000000	1.515690e+09
max	709.000000	47.000000	167.000000	3.000000	13.000000	233.000000	1.518076e+09

Слика 5.12. Дефинисање функције за конверзију датума у одговарајући облик

Затим, поново приказујемо основне статистичке мјере и првих 5 редова скупа података.

Now, to see first few rows of prepared data:

```
data.head()
```

	co2	humidity	light	motion	person_count	temperature	time_device
0	559	37	76	1	7	212	1.513167e+09
1	569	37	78	3	7	212	1.513167e+09
2	588	37	73	1	7	213	1.513167e+09
3	579	37	75	2	7	213	1.513167e+09
4	565	37	77	1	7	213	1.513167e+09

Слика 5.13. Приказ основних статистичких мјера и првих 5 редова скупа података

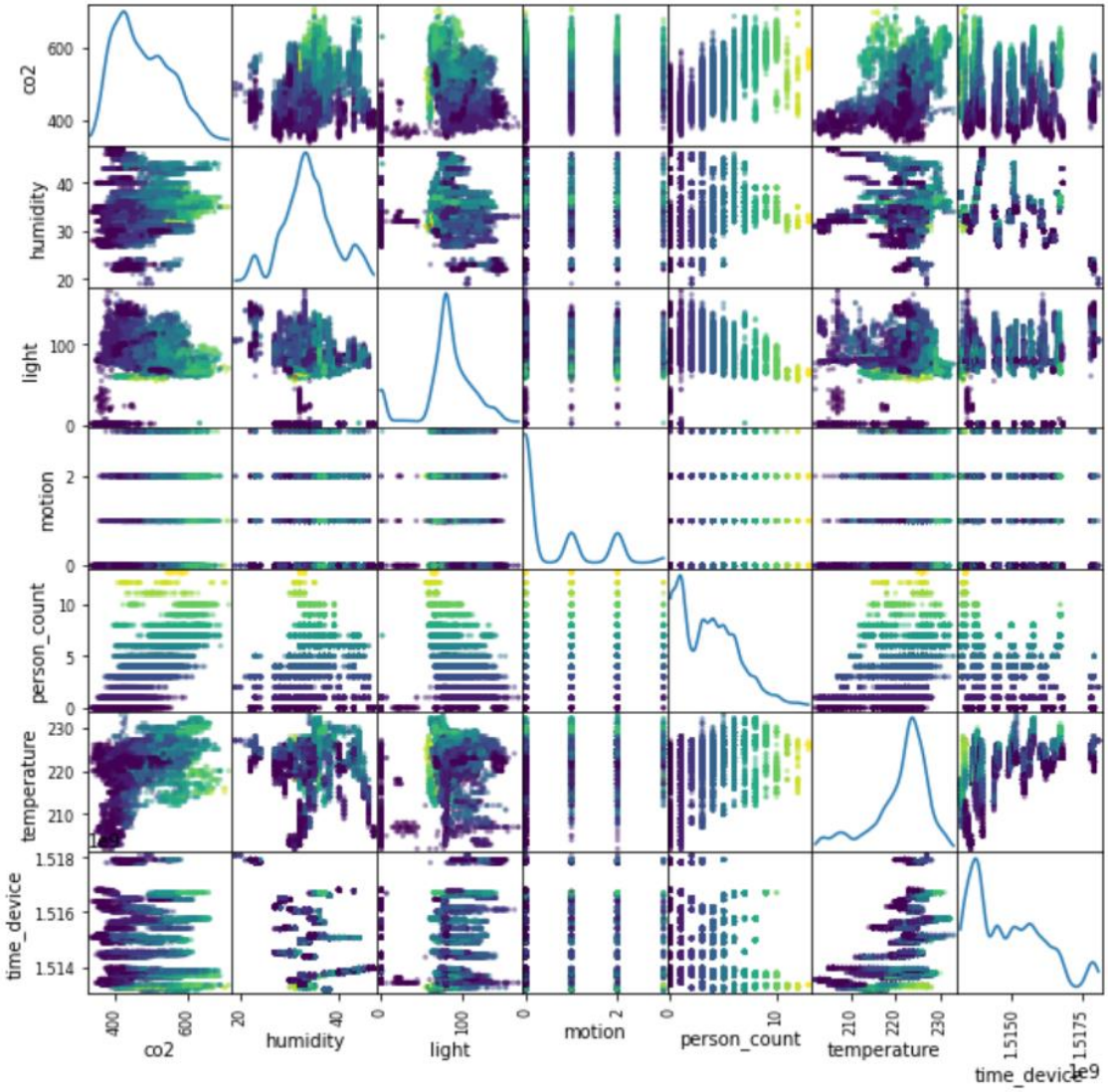
Након овога, можемо да израчунамо и прикажемо и матрицу распршености нашег скупа података. Овај графикон омогућава визуелно поређење дистрибуција између атрибута, као и идентификацију евентуалних зависности или образаца међу атрибутима у скупу података.

Пошто нема смисла да рачунамо корелацију колоне према самој себи, на главној дијагонали можемо да видимо криувуљу дистрибуције података за поједине колоне, чија висина и облик показују густоћу расподјеле података, што може бити корисно за разумијевање карактеристика података.

Scatter matrix

Plot of scatter matrix with respect to person_count column:

```
pd.plotting.scatter_matrix(data, c=data['person_count'], figsize=[10, 10], diagonal = 'kde')
plt.show()
```



Слика 5.14. Приказ матрице распршености

Скуп података смо подијелили на скупове тако што смо прво издвојили колону за број особа из скупа података, а онда смо креирали сљедеће скупове:

- X – скуп улазних података за тренирање модела
- X1 – скуп улазних података за тестирање и оцјењивање модела
- y_train – скуп излазних података за тренирање модела
- y_test – скуп излазних података за тестирање и оцјењивање модела

```
x = data.drop('person_count', axis=1)
y = data['person_count']
X_train, X_test1, y_train, y_test1 = train_test_split(x, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X = scaler.fit_transform(X_train)
X1 = scaler.transform(X_test1)
```

Слика 5.15. Подјела скупа података на одговарајуће подскупове

5.4. Избор и тренирање модела

Наизмјенично смо бирали, креирали и записивали резултате за сљедеће алгоритме:

- `LogisticRegression`
- `GaussianNB`
- `KNeighborsClassifier`
- `DecisionTreeClassifier`
- `RandomForestClassifier`
- `GradientBoostingClassifier`
- `SVC`
- `LGBMClassifier`
- `LGBMRegressor`
- `KNeighborsRegressor`

Models:

```
results = []

model = LogisticRegression(solver='liblinear', C=0.05, multi_class='ovr',random_state=0)
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Logistic Regression"))

model = GaussianNB()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Gaussian Naive Bayes Classification"))

model = KNeighborsClassifier()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "K-Neighbors Classification"))

model = DecisionTreeClassifier()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Decision Tree Classification"))

model = RandomForestClassifier()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Random Forest Classification"))

model = GradientBoostingClassifier()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Gradient Boosting Classification"))

model = SVC()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Support Vector Machine Classification"))

model = LGBMClassifier()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Light Gradient Boosting Machine Classification"))

model = LGBMRegressor()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "Light Gradient Boosting Machine Regression"))

model = KNeighborsRegressor()
results.append(fit_and_analise(model, X, X1, y_train, y_test1, "K-Neighbors Regression"))
```

Слика 5.16. Тренирање модела

5.5. Оцјењивање модела

На крају, потребо је да прикажемо резултате и перформансе свих модела кориштених за тренирање и тестирање.

Приказујемо табеларно помоћу Time plot дијаграма све моделе који су кориштени заједно са временом извршавања и тачношћу алгоритма. Овај дијаграм нам помаже да визуално упоредимо брзину извршавања сваког модела и идентификујемо моделе који су релативно спори у односу на остале.

Accuracy plot дијаграм приказује тачност сваког модела на скуповима за тренирање и тестирање. Ова линија кода пружа визуелни преглед тачности различитих модела и помаже у идентификовању модела који су најпрецизнији.

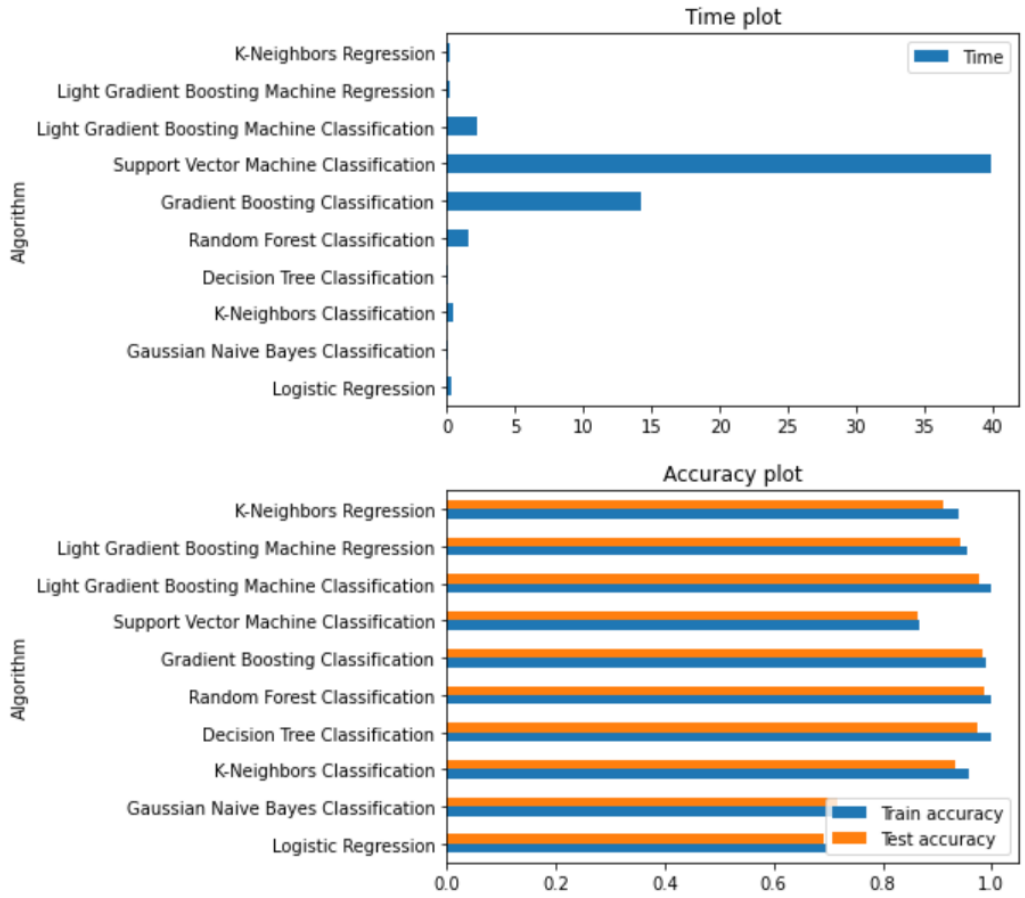
```
resultsFrame=pd.DataFrame(results,columns=['Algorithm','Time', 'Train accuracy', 'Test accuracy'])
print(resultsFrame)
resultsFrame.plot(x = 'Algorithm', y = 'Time', kind = 'barh', title = "Time plot")
resultsFrame.plot(x = 'Algorithm', y = ['Train accuracy', 'Test accuracy'], kind = 'barh', title = "Accuracy plot")
```

Слика 5.17. Приказивање резултата тренирања

```
                                  Algorithm       Time  Train accuracy  \
0                       Logistic Regression   0.306911        0.698072
1            Gaussian Naive Bayes Classification   0.046870        0.736488
2                     K-Neighbors Classification   0.470277        0.957493
3                  Decision Tree Classification   0.046906        1.000000
4                  Random Forest Classification   1.524090        1.000000
5              Gradient Boosting Classification  14.216484        0.989775
6          Support Vector Machine Classification  39.949486        0.866491
7  Light Gradient Boosting Machine Classification   2.260866        1.000000
8     Light Gradient Boosting Machine Regression   0.161067        0.954280
9                       K-Neighbors Regression   0.144357        0.938066

   Test accuracy
0       0.692757
1       0.717874
2       0.931659
3       0.975467
4       0.987150
5       0.983645
6       0.865070
7       0.978388
8       0.943925
9       0.912383
<AxesSubplot:title={'center':'Accuracy plot'}, ylabel='Algorithm'>
```

Слика 5.18. Приказ резултата тренирања



Слика 5.19. Графички приказ резултата тренирања

6. Резултати

У анализи је испитано десет модела за процјену особа у просторији. Сваки модел пружа јединствен скуп предности и недостатака, а њихова ефикасност може се разликовати у зависности од специфичног контекста употребе.

Logistic Regression је брз модел с временомн извршавања од само 0.307 секунди, али са тачношћу тестирања од 69.3%, не пружа довољну прецизност за неке захтјевније примјене. Слично томе, Gaussian Naive Bayes Classification је још бржи, али и даље пружа тачност испод 75%.

Gradient Boosting Classification, Support Vector Machine Classification и Support Vector Machine Classification су спорији модели али пружају врло добре резултате тачности тестирања.

Модел попут K-Neighbors Classification, Decision Tree Classification i Light Gradient Boosting Machine Classification комбинују брзину с високом тачношћу, прелазећи 93% на тестирању. Међутим, савршена тачност тренирања код Decision Tree и Light Gradient Boosting алгоритама може указивати на потенцијално преприлагођавање, што значи да модел може неадекватно генерализовати на невиђеним подацима.

Иако и други алгоритми пружају снажне перформансе, Random Forest Classification се истиче као најпрецизнији са изузетном тачношћу тестирања од 98.7%. Иако није алгоритам који се извршава најбрже, вријеме извршавања од око 1.5 секунди је прихватљиво за потребе мјерења броја особа у просторији, пошто ће се модел само једном тренирати. Његова савршена тачност тренирања такође указује на могућност преприлагођавања.

7. Закључак

Машинско учење ствара нове могућности у свијету технологије, омогућавајући рачунарима да „уче“ из података и доносе одлуке које превазилазе класично програмирање. Ове способности постају кључне у ситуацијама гдје су проблеми комплексни и динамични.

У посљедњих неколико година, машинско учење и вјештачка интелигенција доживјели су праву ренесансу захваљујући брзим технолошким иновацијама. Савремене примјене машинског учења нису само унаприједиле традиционалне области попут препознавања слика или предвиђање текста, већ су и отвориле врата за револуционарне могућности у интеракцији са рачунарима и обради природног језика (OpenAI ChatGPT) и (Google BARD).

У оквиру овог дипломског рада, фокус је на примјени машинског учења у контексту предвиђања броја особа у просторији. Таква предвиђања могу имати широку примјену, од оптимизације клима уређаја до повећања безбједности простора.

Током израде овог рада, нагласак је стављен на цјелокупан процес развоја рјешења. Почетак је обиљежило прикупљање релевантних података (проналазак одговарајућег скупа података), који су основ сваког модела машинског учења. Након тога, подаци су обрађени и анализирани како би се идентификовали кључни обрасци и информације. Овај корак је од суштинског значаја, јер квалитет улазних података директно утиче на тачност и ефикасност коначног модела.

Правилна процјена модела је неопходна како би се осигурало да модел не само да ради добро на познатим подацима, већ и да може генерализовати своје знање на нове, невиђене ситуације. То се постиже пажљивом подјелом података на тренинг, валидациони и тест скуп.

Без обзира на све техничке аспекте и изазове, циљ овог рада је био развити рјешење које може да се користи у стварном свијету, пружајући тачна и брза предвиђања броја особа у просторији. И док сваки модел има своје предности и недостатке, права вриједност долази из његове способности да пружи практичне резултате у стварном окружењу.

8. Литература

[1] D. Kirsch и J. Hurwitz, Machine Learning For Dummies, New Jersey: John Wiley & Sons, Inc., 2018.

[2] F. Williams, Meet the nine billion-dollar companies turning a profit from sustainability, The Guardian, 2016.

[3] S. Russell и P. Norvig, Artificial IntelligenceA Modern Approach, New Jersey: Pearson Education, Inc., 2010.

[4] M. Mohri, A. Rostamizadeh и A. Talwalkar, Foundations of machine learning, Cambridge, Massachusetts: The MIT Press, 2018.

[5] T. Mitchell, Machine Learning, New York: McGraw-Hill, 1997.

[6] E. Alpaydin, Introduction to Machine Learning, Cambridge, Massachusetts: The MIT Press, 2010.

[7] G. James, D. Witten, T. Hastie и R. Tibshirani, An Introduction to Statistical Learning: with Applications in R, New York City, USA: Springer, 2017.

[8] I. H. Witten, E. Frank и M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Burlington, USA: Morgan Kaufmann, 2011.