Primjena tehnika računalnih društvenih znanosti

Alen, Ugarković

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Alen Ugarković

PRIMJENA TEHNIKA RAČUNALNIH DRUŠTVENIH ZNANOSTI

DIPLOMSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Alen Ugarković

Matični broj: 44104/15–R

Studij: Informacijsko i programsko inženjerstvo

PRIMJENA TEHNIKA RAČUNALNIH DRUŠTVENIH ZNANOSTI

DIPLOMSKI RAD

Mentorica:

Izv. prof. dr. sc. Dijana Oreški

Varaždin, ožujak 2022.

Alen Ugarković

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Tema je ovog rada primjena odabranih tehnika računalnih društvenih znanosti na javno dostupnim podacima o odlasku zaposlenika iz kompanije. Teorijski dio opisuje pojam računalnih društvenih znanosti i daje uvid u neke od najpoznatijih tehnika koje se koriste u tom području. Posebnu pažnju daje se tehnikama strojnog učenja jer se u praktičnom dijelu iste i primjenjuju. Podaci o zaposlenicima služe kao temelj izgradnje prediktivnog modela. Isti taj model koristi se u web aplikaciji koja je napravljena za predikciju odlaska budućih zaposlenika. To je i cilj ovoga rada – pokazati kako se velika količina podataka može upotrijebiti za nova saznanja koristeći moderne tehnologije.

Ključne riječi: računalne društvene znanosti, društvo, društvene znanosti, veliki podaci, digitalni otisak, podatkovna znanost, strojno učenje, python, predikcija

Sadržaj

1. Uvod

Ubrzanim rastom količine digitalnih podataka u svim sferama našeg života došlo je i do potrebe za iskorištavanjem tih istih podataka. Fenomen kojeg nazivamo *big data* već je odavno popularna tema u računalnim i društvenim znanostima. Društvo je doživjelo preokret procvatom digitalne tehnologije, a ista ta tehnologija mijenja i način spoznavanja društva. Proučavanje digitalnog otiska ljudske interakcije daje nam novi pogled na društvo i omogućuje njegovo bolje shvaćanje. Računalni alati i tehnike uspjevaju otkriti skrivene obrasce u ljudskom ponašanju koje ljudi sami nisu u mogućnosti otkriti. Najistaknutiji alat svakako je strojno učenje čije se mogućnosti svakodnevno unapređuju.

Ovaj rad bavi se proučavanjem područja računalnih društvenih znanosti i primjenom najznačajnijih tehnika toga područja nad javno dostupnim skupom podataka. U trećem poglavlju objasnit ćemo kako računalna i društvena znanost zajedno funkcioniraju kao zasebna znanost. U četvrtom poglavlju fokus će biti na tehnikama i alatima koji se koriste u ovom području. Sve tehnike koriste modele i algoritme računalnih znanosti. Algoritmi strojnog učenja najpoznatiji su način za otkrivanje znanja u podacima društvenih znanosti. Takve algoritme koristi i istraživanje kojeg ćemo opisati u petom poglavlju. U tom istraživanju korišten je skup podataka telekomunikacijskih operatera u kombinaciji s podacima koje su dobili anketiranje korisnika telekomunikacijskih usluga. U šestom poglavlju ovog rada primijenit ćemo naučene tehnike i napraviti model za predviđanje odlaska zaposlenika. Rad modela za predikciju prikazat ćemo radom web aplikacije. U posljednjem poglavlju iznijet ćemo temeljne zaključke i spoznaje do kojih smo došli.

2. Metode i tehnike rada

Teorijski dio rada sadrži detaljan opis svih ključnih pojmova koji su vezani uz računalne društvene znanosti. Opisat ćemo domenu kojom se bavi računalna društvena znanost i objasniti kako fenomen velikih podataka utječe na razvoj ove znanosti. Posebnu pažnju posvetit ćemo strojnom učenju kao glavnoj uzdanici ovog područja. Detaljnije ćemo opisati metode i tehnike strojnog učenja da bismo ih mogli primijeniti u praktičnom dijelu.

Praktični dio kreće od opisa javno dostupnog skupa podataka o zaposlenicima neke kompanije. Skup podataka detaljno ćemo opisati i vizualizirati pojedine dijelove kako bismo shvatili problem odlaska zaposlenika. Skup podataka zatim ćemo pripremiti kako bismo mogli izgraditi model za predikciju. Model ćemo kreirati koristeći odabranu tehniku strojnog učenja, a na kraju ćemo ga primijeniti u jednoj jednostavnoj web aplikaciji.

Za rad s podacima koristit ćemo Python programski jezik i Juypter Notebook *online* okruženje. Ono će nam omogućiti brzu i jednostavnu analizu i manipulaciju podacima. Od učitavanja skupa podataka do izgradnje modela za predikciju, Jupyter Notebook bit će nam glavna radna okolina. Model kojeg kreiramo zatim ćemo ubaciti u web aplikaciju napravljenu u Flasku. Aplikaciju ćemo posluživati na lokalnom serveru kako bismo mogli obavljati predikciju na interaktivan način u web pregledniku.

Sadržaj web aplikacije napravit ćemo u HTML jeziku za izradu web stranica. Napravit ćemo formu za unos podataka koja će podatke slati lokalnom serveru na obradu. S podacima u formi baratat ćemo koristeći JavaScript programski jezik i jQuery biblioteku. Serveru ćemo podatke slati asinkrono putem AJAX zahtjeva kako bismo spriječili osvježavanje web stranice.

Stiliziranje web stranice obavit ćemo u CSS-u. Radi se o jeziku za opis dokumenta napisanog u HTML-u.

3. Računalne društvene znanosti

Računalna društvena znanost (eng. *Computational Social Science*, u daljnjem tekstu RDZ) relativno je novo područje koje se bavi sistematičnim pristupom stvaranju znanja o ponašanjima u društvu. Martin Hilbert [1], profesor sa Sveučilišta u Kaliforniji, tvrdi da se najveće kompanije u svijetu uvelike bave računalnim društvenim znanostima. On ne tvrdi kako im je to glavna djelatnost, već bitna stavka u poslovanju.

Matthew J. Salganik [7] sa Sveučilišta Princeton smatra da trenutno nema potrebe za formalnom definicijom područja računalnih društvenih znanosti jer je područje relativno novo i jer se brzo mijenja. Definicija kojom bi opisali područje prije deset godina u današnje vrijeme ne bi bila ispravna. Navodi kako bi trebalo izbjegavati postavljanje ograničenja domene kojom se ovo područje bavi. Za njega je računalna društvena znanost primjer društvenog pokreta, otvorenog za doprinos mnogih ljudi kako bi područje moglo rasti. Najbolji način za proučavanje područja mogu nam dati provedene studije koje ilustriraju najvažnije koncepte područja. Profesor Salganik upućuje na problem koji se često pojavljuje u području RDZ, a to je pokušaj procjene nečega što se nikad nije procjenjivalo pa je jako teško znati koliko dobro nova tehnika radi svoj posao. Jedan od načina provjere tehnike jest usporedba sa stvarnim stanjem, a njega najbolje prikazuju ankete. Međutim, njihovo provođenje zahtjeva mnogo novca i vremena. Zbog toga se moramo osloniti na modernu tehnologiju i velike količine podataka kojima možemo lako pristupiti.

Budućnost je RDZ-a u učenju iz velikih podataka. To bi bila kombinacija računalne i društvene znanosti. Računalna obrađuje veliku količinu podataka i konstruira modele koji nam mogu dati dobru sliku o pojavi koju opisuju podaci, a društvena postavlja istraživačka pitanja, objašnjava računalne modele i donosi smislene zaključke. [7]

Tranzicija iz analognog doba u digitalno praćena je trendom digitalne pohrane informacija. Profesor Salganik navodi kako se svake dvije godine količina digitalno pohranjenih informacija udvostručava i zato smatra da društveni znanstvenici moraju iskoristiti ovu pogodnost digitalnog doba. Digitalni se podaci kreiraju svakodnevno u svim područjima ljudske djelatnosti, a to pruža velike mogućnosti. [7]

RDZ pomažu u razumijevanju društvenog svijeta. Nužna je suradnja podatkovnih znanstvenika i društvenih znanstvenika kako bi područje moglo rasti. Te dvije discipline mogu zajedno stvarati novu vrijednost i dati odgovore na dosad neobjašnjive pojave.

3.1. Digitalna revolucija

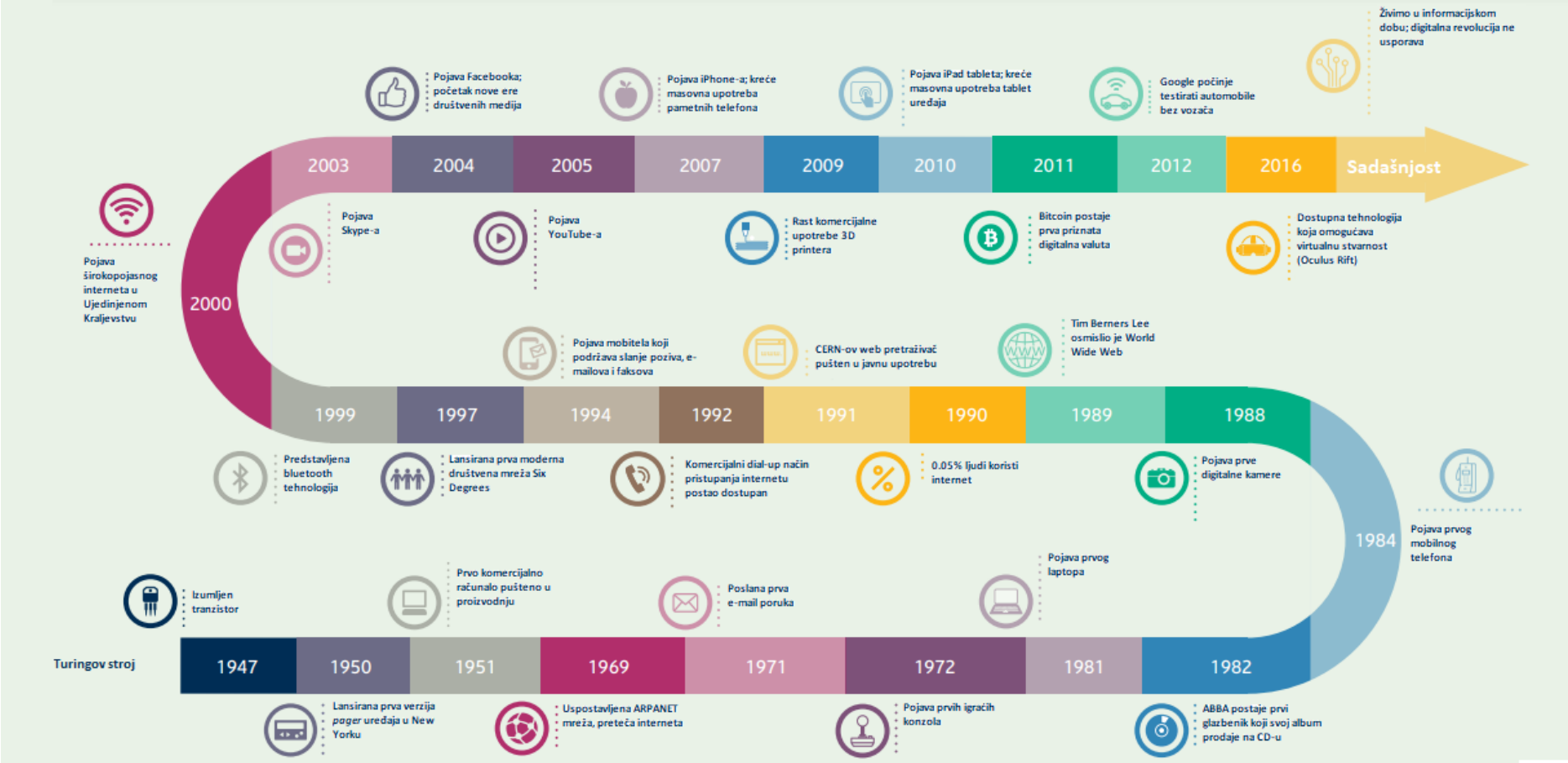
Ideju o razvoju računalnih društvenih znanosti pokrenula je jedna spektakularna pojava koja se zove digitalna revolucija. Dogodila se nevjerojatno brzo i zahvatila je gotovo sve sfere života. Još se krajem 80-ih godina 20-og stoljeća 99% informacija pohranjivalo u analognom obliku, najčešće na papiru. Tijekom devedesetih godina sve se više informacija pretvaralo u digitalni oblik. Početkom 2000-ih godina digitalna je revolucija „eksplodirala“. Sve se veća količina informacija pohranjivala digitalno pa se smatra da je trenutno 99% svih informacija pohranjeno u digitalnom obliku. [1]

Prednosti digitalno pohranjenih informacija su brojne. S obzirom da su informacije u digitalnom formatu, možemo ih i digitalno analizirati, provoditi različite izračune, dovoditi neke zaključke i na taj način stvarati novo znanje. Količina digitalno pohranjenih informacija s kojima trenutno baratamo je nevjerojatna. Svakim danom stvaramo sve više informacija, slikamo, snimamo i digitalno bilježimo društvenu stvarnost. Tim aktivnostima doprinosimo razvoju društvenih znanosti. Dijelimo svoje digitalne informacije, dobivamo povratne informacije od drugih ljudi, analiziramo ih i proučavamo. Na neki smo način svi postali društveni znanstvenici. Svi se, svjesno ili nesvjesno, bavimo računalnim društvenim znanostima jer živimo u digitalnom svijetu. Cijelo je društvo postalo ekstremno ovisno o digitalnoj tehnologiji. [1]

Digitalna nam je revolucija uvelike promijenila način života. Od zdravstva do obrazovanja, svaki se aspekt našeg života transformirao. Živimo u informacijskom dobu, a svijet je već odavno postao globalno selo. Sve je krenulo još za vrijeme Drugog svjetskog rata kada je briljantni britanski matematičar Alan Turing, zajedno s još nekoliko znanstvenika, uspio dešifrirati Enigmu, nacističku spravu za šifriranje poruka. To je uspio uz pomoć stroja kojeg je sam konstruirao. Stroj kojeg je napravio smatra se pretečom današnjeg računala, a njegov cjelokupni rad predstavlja početak računalnog doba. Posao koji su odradili ti znanstvenici dao je smjer kojeg su slijedili ostali briljantni ljudi. Alan Turing smatra se ocem modernog računalstva, no moderno računalo rezultat je rada mnogih ingenioznih ljudi tijekom sljedećih desetljeća. [3]

Na slici 1 prikazana je kratka povijest digitalne revolucije. Turingov stroj započeo je informatičku eru koja se počela ubrzano razvijati nakon Drugog svjetskog rata. Pojavom tranzistora došlo je do tehnološke revolucije. Tranzistor je postao osnova elektroničkih sklopova i računalnih sustava. Već 1980-ih godina računalo je postalo prisutno u mnogim sferama ljudskog života. Devedesete godine bile su prekretnica zbog pojave *World Wide Weba*, a internet je postao važan dio mnogih poslovnih operacija. Prije početka 2000. godine internet je koristilo gotovo pola američkog stanovništva. Početkom novog tisućljeća digitalna se revolucija počela širiti svijetom. Mobilni telefoni postali su svakodnevica, rastao je broj korisnika interneta. U zadnjih deset godina mobilna je komunikacija postala veoma važna. Mogućnost povezivanja na internet preko mobilnih telefona danas je neizostavan dio ljudskih života. Stvaramo sve više informacija i lako ih dijelimo s drugima. Trenutno živimo u informatičkom dobu, a digitalna revolucija i dalje traje. [4]

Računalna znanost posljedica je digitalne revolucije. Razvojem elektroničkih računalnih uređaja postalo je moguće baviti se problemima koji su se smatrali prezahtjevnima. Osim toga, omogućen je razvoj novih metoda analiziranja tih zahtjevnih problema, a posebno se tu ističu metode koje su osmišljene u kontekstu RDZ. [1]



Slika 1: Digitalna revolucija (Prema: [2])

3.2. Društvene znanosti

Društvene znanosti bave se problemima koje nije jednostavno modelirati, problemima gdje postoji mnogo varijabli koje se uzimaju u obzir i gdje nije jednostavno izračunati nekakav prosjek. Razlog tome je raznolikost društva i razne međuovisnosti koje se pojavljuju u društvu. Ljudsko je društvo izrazito kompleksno, postoje sličnosti i različitosti. Na njega možemo gledati kao na mrežu ljudi. Svaka je osoba jedinstvena pa je teško promatrati pojedinca i na temelju toga donositi zaključke. Zato promatramo međusobnu interakciju svih ljudi u mreži koju smo stvorili. Interakcija je dodatno unaprijeđena modernom tehnologijom pa možemo pričati o društveno-tehnološkom sustavu kojeg čine ljudi. [1]

Ono čime se društvene znanosti najviše bave su predviđanja. Promatrajući ponašanja u društvu, počet ćemo primjećivati sličnosti u ponašanju. Sličnosti ili obrasci ponašanja mogu nam dati osnovu za predviđanje i donošenje zanimljivih zaključaka. Na primjer, u modernom društvu odabir bračnog partnera temelji se na slobodnoj volji. Međutim, ako pogledamo širu sliku možemo primjetiti određene pravilnosti. Na temelju toga moguće je napraviti predviđanje o broju ljudi koji će se vjenčati u nekom mjestu. Čovjekova slobodna volja iz perspektive društvenih znanosti postaje nešto što možemo predviđati i donositi zaključke s poprilično velikom sigurnošću. [1]

Profesor Hilbert [1] ističe još jedan fenomen kojeg je još davno primjetio škotski ekonomist Adam Smith. Smithova teorija o nevidljivoj ruci opisuje fenomen kojim pojedinac nenamjerno utječe na opću dobrobit društva. Svaki čovjek vodi se isključivo svojim interesima i fokusiran je na svoj napredak. Međutim, time ujedino promiče i javni interes više nego što bi ga promicao da to uistinu namjerava. To je još jedan primjer proučavanja društvenih struktura na višoj razini, onoj koja u obzir uzima cjelokupno društvo i interakcije unutar njega.

3.3. Znanstvena metoda u računalnim društvenim znanostima

Znanstvena metoda je procedura kojom pronalazimo odgovore na pitanja koja su nam interesantna. Prema [1], znanstvena metoda u RDZ provodi se sljedećim koracima:

1. Identifikacija predmeta promatranja i definiranje problema
2. Prikupljanje podataka
3. Apstrakcija / Analiza / Model

4. Hipoteza
5. Teorija

Na početku promatramo nekakav fenomen u društvu, nešto što nas zanima i o čemu želimo više saznati. Postavljamo si pitanje na koje želimo dobiti odgovor. Pitanje mora biti smisleno i postavljeno tako da je moguće pronaći odgovor. U procesu znanstvene metode definiranje ispravnog pitanja veoma je važno jer njime pokrećemo proces stvaranja znanja.

Sljedeći nam je korak prikupiti podatke o tom fenomenu, pojavi u društvu. Moramo nekako prikupiti informacije koje su relevantne za zadani problem. Različiti su načini prikupljanja: razgovori i intervjui s ljudima, ankete, opažanja i eksperimenti, postojeći skupovi podataka. Iz tih podataka potrebno je napraviti model koji će dobro opisati naše podatke. Postoje metode i tehnike strojnog učenja koje nam mogu pomoći, pokazati kako podaci izgledaju, postoje li pravilnosti, kakva je korelacija između podataka...

Na temelju modela radimo hipotezu. Hipoteza je objašnjenje ili rješenje problema, odnosno naše očekivanje. Izražava se kao izjavna rečenica. Ona ne mora uvijek biti ispravna, ali važno je da je provjerljiva i da je se može testirati. Ako se hipoteza ne može testirati, onda ne možemo dati odgovor na pitanje o njenoj ispravnosti. Ispravnost hipoteze provodimo eksperimentom. Eksperiment treba ponoviti dovoljan broj puta kako bi se izbjegle slučajne greške. Nakon što testiramo hipotezu, donosimo zaključak. Ili je hipoteza ispravna ili je pogrešna. Ako je ispravna, onda ju prihvaćamo kao temelj naše teorije. [1]

3.4. Veliki podaci i digitalni trag

Podataka kojima se bavi RDZ ima sve više. Živimo u digitalnom okruženju u kojem se prati gotovo svaka naša akcija: objave na društvenim mrežama, povijest našeg pretraživanja, interesi prilikom online kupovine, slike i videozapisi koje pregledavamo... Svaki naš klik ostavlja digitalni trag. Podaci o nama generiraju se i prikupljaju cijelo vrijeme, a kasnije se koriste za razne analize. Svakodnevno generiranje velike količine podataka dovelo je do stvaranja nove sintagme koja ih opisuje, veliki podaci (eng. *big data*). [5]

Prema [6], veliki podaci predstavljaju skupove podataka koji su preveliki da bi se mogli spremiti i procesirati putem tradictionalnih relacijskih baza podataka. Karakteristični su po svojem velikom obujmu, brzini stvaranja i različitosti. Izvori iz kojih nastaju sve su kompleksniji jer se u zadnje vrijeme sve više kreiraju automatski. Kreira ih umjetna inteligencija, mobilni telefoni, društvene mreže, različiti senzori i aplikacije. IBM smatra da je

velika prednost velikih podataka u mogućnostima analiziranja i brzog donošenja odluka, modeliranja i predviđanja.

Prema [1], najbolji termin koji opisuje velike podatke jest digitalni otisak. Profesor Hilbert ističe fenomen kojeg naziva fuzija podataka (eng. *data fusion*). To bi bila integracija nekoliko različitih izvora podataka koja stvara potpunije podatake za analizu. Izvori su mnogi: ljudske aktivnosti na internetu, korištenje mobilnih usluga, korištenje kreditnih kartica… Najznačajnije karakteristike velikih podataka su sljedeće:

- automatsko prikupljanje podataka
- kombinacija različitih izvora podataka
- korištenje strojnog učenja
- laka dostupnost

4. Tehnike računalnih društvenih znanosti

4.1. Računalne simulacije

Prema [1], uz RDZ najviše se vežu pojmovi poput umjetne inteligencije i velikih podataka. Međutim, najkompleksniji aspekt tog područja smatra se razvoj teorije društvenih znanosti pomoću moderne tehnologije. Računalne simulacije jedan su od načina istraživanja teoretskih mogućnosti društvenih znanosti. Omogućavaju nam da proučimo društvo onakvo kakvo jest i daju nam uvid u svijet kakav bismo željeli. Za proučavanje takvih umjetnih društava koriste se modeli temeljeni na agentima. Bez utjecaja na stvarni svijet, simulacijama stvaramo virtualne svjetove. Računalne simulacije kombiniraju hipotetske modele s podacima iz stvarnog svijeta što nam omogućava uvid u kompleksnosti društvenih sustava.

4.2. Analize društvenih mreža

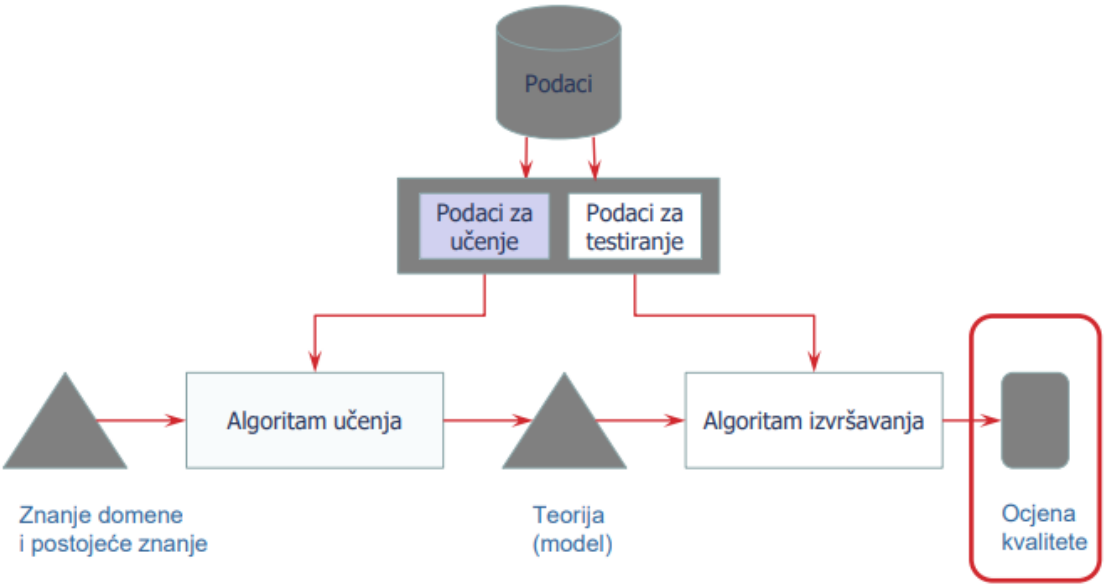
Prema [1], središte modernog društva su društvene mreže. Ljudi se neprestano povezuju i šire svoju mrežu poznanstava (obitelj, prijatelji, radni kolege, razne organizacijske strukture...). Takve su mrežne strukture odvijek postojale, a s obzirom da se sada velikim dijelom nalaze na internetu, RDZ je omogućila njihovo detaljnije proučavanje.

Prema [31], analiza društvenih mreža (eng. *social network analysis*) koristi brojne alate i metode za identifikaciju i analizu veza između aktera društvenih mreža. Akteri mogu biti osobe, grupe ili organizacije. Veze između aktera mogu biti formalne ili neformalne, prijateljske ili poslovne. Proučavaju se komunikacijski tokovi, poslovne transakcije, interakcije na društvenim mrežama. Prezentirati se mogu na različite načine i u različitim formatima (dijagrami, slike, matrice, grafovi...).

4.3. Strojno učenje

Prema [29], strojno učenje je područje koje se bavi identifikacijom pravila i obrazaca u empirijskim podacima pomoću računala. Podaci iz kojih algoritmi uče nazivaju se podaci za treniranje. Rastom količine podataka za treniranje rastu i performanse algoritama, odnosno vjerojatnost da će algoritam prepoznati skrivene obrasce u podacima. Iz toga proizlazi i najvažnija značajka algoritama, a to je njihova sposobnog obavljanja zadataka za koje nisu

eksplicitno programirani. To rade tako što koriste saznanja iz testnih podataka i primjenjuju ih na novim, dosad neviđenim podacima.



Slika 2: Proces strojnog učenja (Izvor: [28])

Na slici iznad prikazan je proces strojnog učenja. Na početku se algoritmu daje skup podataka za treniranje iz kojih on uči. Algoritam koji treniramo nazivamo model. Nakon treniranja, model je potrebno testirati. Podaci za testiranje dolaze iz istog skupa podataka kao i podaci za treniranje. Testiranjem ispitujemo preciznost modela odnosno njegovu sposobnost da naučene koncepte primijeni na dosad neviđenim podacima. Prilikom testiranja možemo otkriti je li naš model pretreniran ili podtreniran. Pretreniranost je slučaj u kojem se model previše prilagodio podacima za treniranje pa daje loše rezultate nad podacima za testiranje. Podtreniranost je slučaj u kojem model ne uspijeva otkriti relacije u podacima. U oba slučaja model nije dobar za predikciju pa je potrebno izmijeniti algoritam učenja.

Prema [28], tri su osnovne skupine u koje se svrstavaju algoritmi strojnog učenja:

1. Nadzirano učenje (eng. *supervized learning*)
2. Nenadzirano učenje (eng. *unsupervized learning*)
3. Učenje podrškom (eng. *reinforcement learning*)

4.3.1. Nadzirano učenje

Prema [28], u nadziranom učenju algoritmu dajemo ulazne vrijednosti i očekivane ciljne vrijednosti, a njegov je zadatak naučiti kako preslikati ulazne vrijednosti u ciljne.

Prema [30], dvije su tehnike nadziranog strojnog učenja: **regresija** i **klasifikacija**. Regresijski algoritmi pokušavaju pronaći korelaciju između ovisnih i neovisnih varijabli i predvidjeti numeričku vrijednost ciljne varijable (cijena, plaća, godine...). Klasifikacijski algoritmi pokušavaju podijeliti skup podataka na određene klase. Rezultat predikcije klasifikacijskih algoritama uvijek je diskretna varijabla (muško/žensko, da/ne...).

Prema [30], najpoznatiji algoritmi nadziranog strojnog učenja su: klasifikacijsko stablo odlučivanja i regresijsko stablo odlučivanja. Klasifikacijsko stablo odlučivanja opisat ćemo detaljnije s obzirom da ćemo ga koristiti u praktičnom dijelu.

4.3.1.1. Stablo odlučivanja

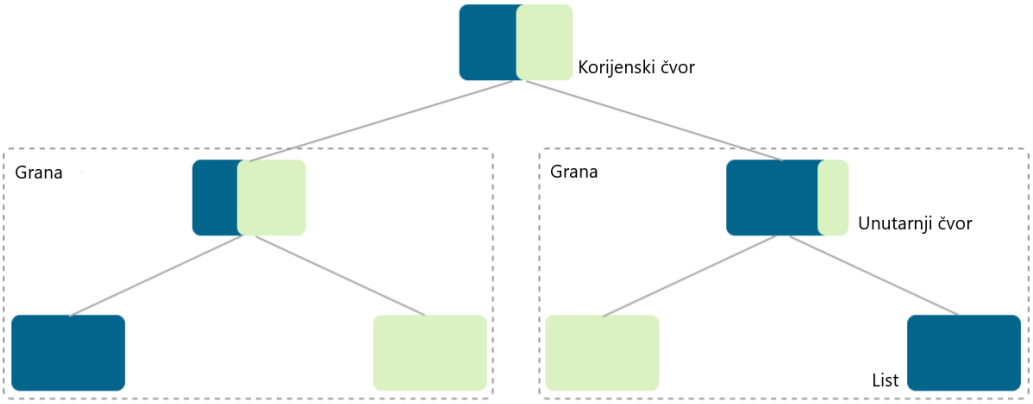
Prema [24], stablo odlučivanja najpopularnija je tehnika u klasifikaciji i predikciji. To je zapravo grafički prikaz koji izgledom podsjeća na stablo. Svaki čvor stabla predstavlja test nad atributom, a svaka grana rezultat tog testa. Na krajevima stabla (listovi) nalazi se ciljna varijabla, ona čiju vrijednost želimo predvidjeti.

Prema [25], pripada skupini algoritama nadziranog strojnog učenja koje koristi skup pravila za donošenje odluka. Algoritam je sličan ljudskom načinu donošenja odluka. Ljudi prilikom odlučivanja sužavaju opcije s obzirom na postavljene uvjete što ih dovodi do konačne odluke, a sam proces može se vizualizirati stablom odlučivanja.

U ovom ćemo se dijelu fokusirati samo na klasifikacijska stabla odlučivanja. Ideja kojom se vodimo prilikom izgradnje stabla je sljedeća: potrebno je iskoristiti atribute određenog skupa podataka kako bi kreirali *da/ne* pitanja i na temelju njih dijelili skup sve dok ne izoliramo sve zapise u određene klase.

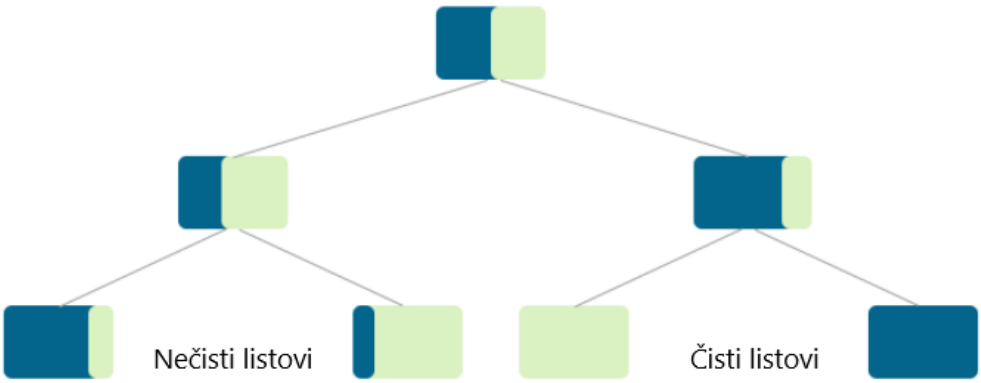
Svaki put kad postavimo pitanje dodajemo novi čvor u stablo. Prvo pitanje predstavlja prvi čvor stabla, njegov korijen. Odgovor na svako pitanje dijeli skup podataka s obzirom na vrijednost atributa nad kojim je pitanje postavljeno. Dijeljenjem skupa podataka nastaju grane stabla.

Jedna grana stabla sadržava sve zapise iz skupa koji pozitivno odgovaraju na pitanje postavljeno u prijašnjem čvoru. Druga grana sadržava sve ostale zapise. Slika 3 prikazuje jednostavno stablo odlučivanja.



Slika 3: Dijelovi stabla odlučivanja (Prema: [25])

Cilj je stabla odlučivanja sužavati početni prostor značajki (sve varijable osim ciljne) postavljajući pritom uvjete sve dok nam ne ponestane uvjeta ili podatkovnih zapisa. Algoritam pokušava u potpunosti razdvojiti skup podataka tako da svaki list (čvor koji više ne dijeli podatke) sadržava podatkovne zapise koji pripadaju jednoj klasi. Ti se listovi smatraju čistima. Međutim, najčešće se u listovima nalaze podatkovni zapisi koji ne pripadaju samo jednoj klasi. Na slici ispod prikazano je stablo odlučivanja s čistim i nečistim listovima. (Prema: [25])



Slika 4: Stablo odlučivanja s čistim i nečistim listovima (Prema: [25])

Algoritam svakom listu može dodijeliti samo jednu ciljnu klasu. Nečistim listovima algoritam dodjeljuje onu klasu koja se najčešće pojavljuje u podatkovnim zapisima. Izgradnja perfektnog stabla koje precizno klasificira svaki podatkovni zapis najčešće nije moguća. Rastom skupa podataka raste i vrijeme potrebno za izgradnju stabla odlučivanja. Kako bi

svaki skup podataka mogli prikazati stablom odlučivanja, koristimo **pohlepni pristup** pri izgradnji svake grane. (Prema: [25])

Prilikom odabira značajke u čvoru odluke, pohlepni pristup pokušava podijeliti skup podataka u najmanje moguće podskupove. Algoritam pokušava smanjiti pogrešku. S obzirom da dijelimo podatkovne zapise koji pripadaju različitim klasama, kvaliteta podjele u čvoru evaluira se s obzirom na čistoću izlaznih čvorova. Evaluira se distribucija klasa podatkovnih zapisa u izlaznim čvorovima. Dva su kriterija po kojima se evaluira kvaliteta podjele: **gini indeks** i **entropija**. (Prema: [25])

Oba kriterija mjere nečistoću podatkovnih zapisa u čvoru. Čvor se smatra najmanje čistim kad 50% podatkovnih zapisa pripada jednoj klasi, a 50% drugoj klasi. Za oba kriterija vrijedi sljedeća tvrdnja: podjela u čvoru dogodit će se samo ako je gini indeks/entropija u izlaznim čvorovima manja od gini indeksa/entropije u roditeljskom čvoru. (Prema: [25])

Uz pohlepni pristup, važno je spomenuti još jednu tehniku koja se koristi kad je riječ o stablima odlučivanja, a to je **obrezivanje** (eng. *pruning*). Prema [26], obrezivanje smanjuje veličinu stabla odlučivanja tako što uklanja dijelove stabla koji nemaju veliku važnost u klasifikaciji. S obzirom da klasifikacijski algoritam nastoji razdijeliti skup podataka na najmanje moguće skupove u kojima se nalaze podatkovni zapisi sa samo jednom klasom, dogodit će se slučaj u kojem listovi stabla sadrže samo nekoliko zapisa. To znači da je stablo previše prilagođeno skupu podataka nad kojim radi (pretreniranost).

4.3.2. Nenadzirano učenje

Prema [28], u nenadziranom strojnom učenju algoritmu dajemo podatke bez označenih vrijednosti, a njegov je zadatak naći pravilnosti u podacima.

Prema [30], dvije su vrste algoritama nenadziranog učenja: **klasteriranje** i **asocijacija**. Najpoznatiji algoritmi nenadziranog učenja su: algoritam k-sredina i neuronske mreže. Detaljnije ćemo opisati algoritam k-sredina s obzirom da ćemo ga koristiti u praktičnom dijelu.

4.3.2.1. Klasteriranje

Prema [27], klasteriranje je tehnika nenadziranog strojnog učenja koja se odnosi na grupiranje podatakovnih zapisa. Algoritmi klasteriranja nad skupom podataka svrstavaju svaki podatkovni zapis u određenu skupinu. Svaka skupina sadržava zapise koji imaju slične značajke, a ostale skupine se bitno razlikuju po tim istim značajkama. Klaster analizu

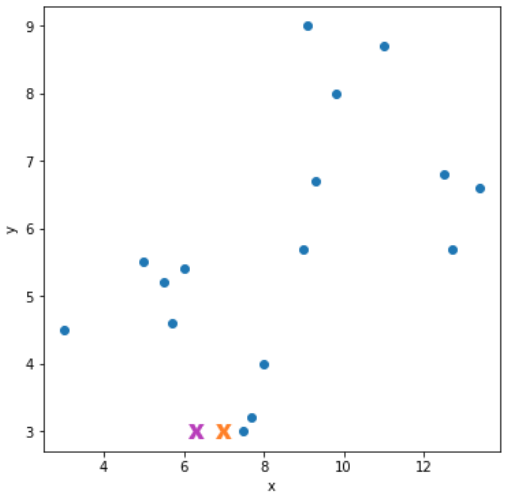
koristimo kako bismo dobili uvid u grupe koje naši podaci mogu formirati. Te grupe nazivamo klasterima.

4.3.2.2. Algoritam k-sredina

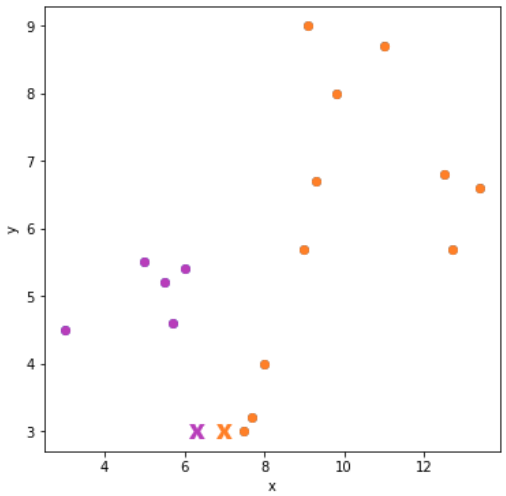
Prema [22], algoritam k-sredina najpopularniji je algoritam za klasteriranje. Koristi iterativnu tehniku za grupiranje neoznačenih podatka u k skupina (klastera). Podaci u svakom klasteru odabrani su tako da im je prosječna udaljenost do centroida (centra svakog klastera) minimalna. Algoritam ima tri koraka:

1. Na graf nasumično postavi k centroida za inicijalne klastere
2. Svaku točku na grafu dodijeli najbližem centroidu
3. Promijeni lokaciju centroida s obzirom na lokacije točaka koje pripadaju toj skupini.
   - koraci 2 i 3 ponavljaju se sve dok centroidi ne stabiliziraju, a točke ne prelaze iz jedne u drugu skupinu

Rad algoritma prikazat ćemo na grafu s nasumičnim podacima. Važno je znati kako algoritmu zadati broj klastera. Tu nam pomaže metoda lakta koju ćemo opisati u sljedećem poglavlju. Za potrebe ovog primjera broj klastera neka bude jednak 2. U prvom koraku nasumično postavljamo 2 centroida.



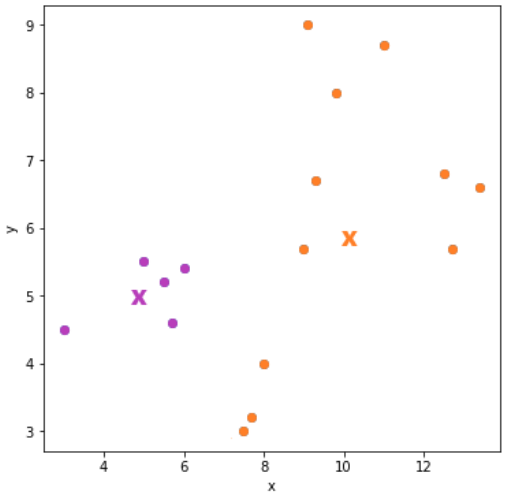
Slika 5: Prvi korak klasteriranja (Izvor: Autor)



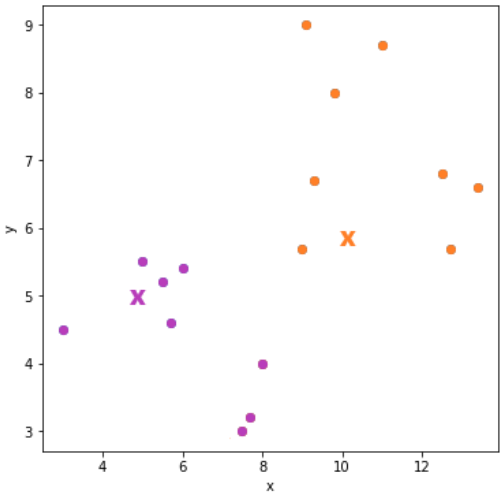
Slika 6: Drugi korak klasteriranja (Izvor: Autor)

Slika 5 prikazuje nasumično postavlljene centroide. Nakon toga, računa se udaljenost između svake točke i oba centroida. Točka se pridružuje onom centroidu do kojeg ima manju

udaljenost. U našem slučaju pridruživanje predstavlja bojanje točaka pripadajućom bojom (prikazano na slici 6).

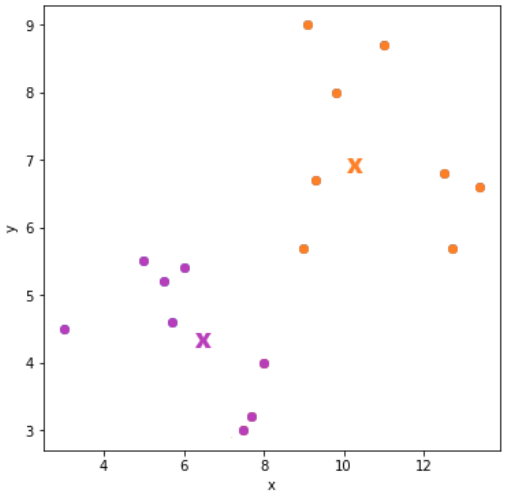


Slika 7: Treći korak klasteriranja (Izvor: Autor)

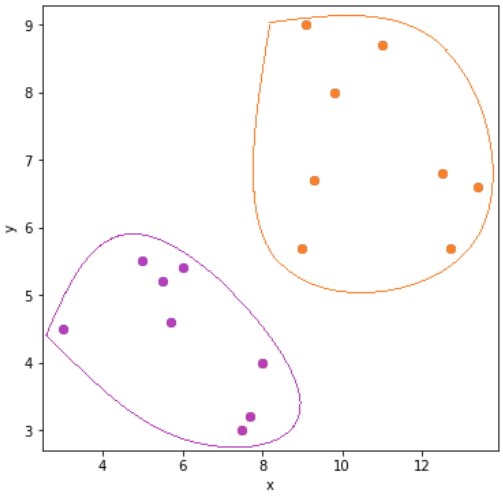


Slika 8: Ponovljeni drugi korak klasteriranja (Izvor: Autor)

Slika 7 predstavlja treći korak u kojem mijenjamo lokaciju centroida. Postavljamo ih tamo gdje je centar gravitacije točaka koje pripadaju tom klasteru (centar gravitacije računa se kao srednja vrijednost svih točaka klastera). Na slici 8 prikazano je ponovno pridruživanje točaka centroidima.



Slika 9: Ponovljeni treći korak klasteriranja (Izvor: Autor)



Slika 10: Konačni klasteri (Izvor: Autor)

Nakon što smo ponovno pridružili točke centroidima, potrebno je ponovno promijeniti lokaciju centroida. Na slici 9 prikazane su nove lokacije centroida. U sljedećem koraku klasterima se ne bi mijenjao broj točaka pa algoritam staje. Na slici 10 prikazano je konačno stanje.

Algoritam k-sredina relativno je jednostavan za primjeniti. Međutim, postoji nekoliko nedostataka. Prema [23], najveći nedostaci su:

- algoritam ima problema u slučaju kada klasteri variraju u veličini i gustoći
- *outlieri*[1] mogu bitno utjecati na položaj centroida; može se dogoditi da *outlieri* dobiju svoj zasebni klaster pa ih je potrebno ukloniti prije klasteriranja
- algoritam jako ovisi o inicijalnom broju klastera koji se ručno odabire

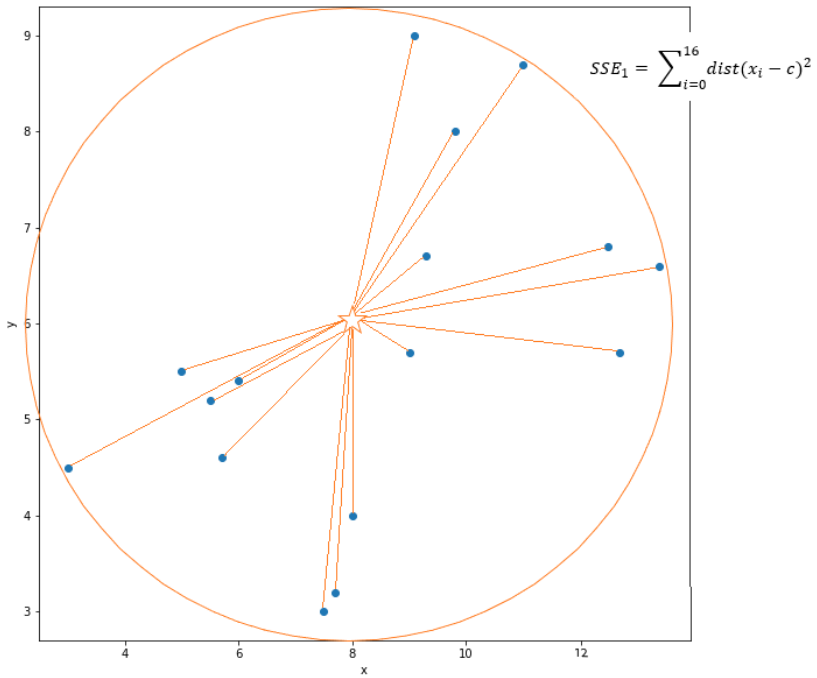
4.3.2.3. Metoda lakta

Metoda lakta služi za određivanje optimalnog broja klastera. Prema [15], određivanje optimalnog broj klastera određuje se na grafu zbroja kvadrata pogrešaka (eng. *sum of squared errors*) prema broju klastera. Zbroj kvadrata pogrešaka suma je koja izgleda ovako:

$$SSE = SSE_1 + SSE_2 + \cdots + SSE_k$$

Da bismo prikazali sumu na grafu, prvo je potrebno odrediti broj iteracija u kojima ćemo računati sumu. Svaka iteracija predstavlja broj klastera na grafu. Uzmemo li u obzir neke nasumične podatke i prikažemo ih na grafu, u prvoj iteraciji imat ćemo samo jedan klaster koji sadržava sve točke. Slika ispod prikazuje početno stanje.

[1] *outlieri* – vrijednosti varijabli koje u velikoj mjeri odstupaju od ostalih (Izvor: [14])

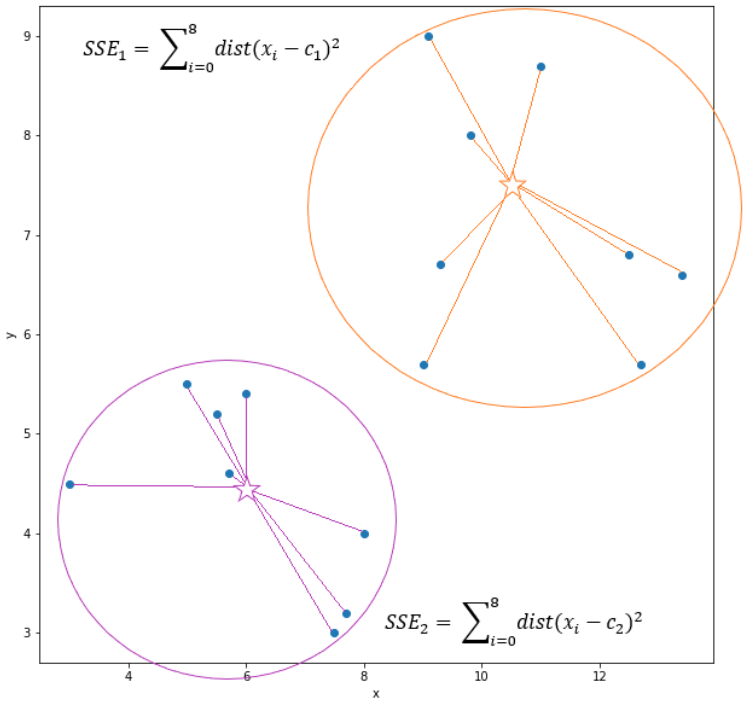


Slika 11: Prva iteracija metode lakta (Izvor: Autor)

Za svaku točku u klasteru računamo udaljenost od centroida (centra klastera). Svaku udaljenost kvadriramo i dodajemo ukupnom zbroju. S obzirom da gornji klaster ima 16 točaka, imat ćemo 16 pribrojnika. To nam daje zbroj kvadrata pogrešaka za prvi klaster, a s obzirom da je jedini, to je ujedno i konačni zbroj:

$$SSE = SSE_1$$

U drugoj iteraciji imamo dva klastera. Svaki klaster ima 8 točaka.



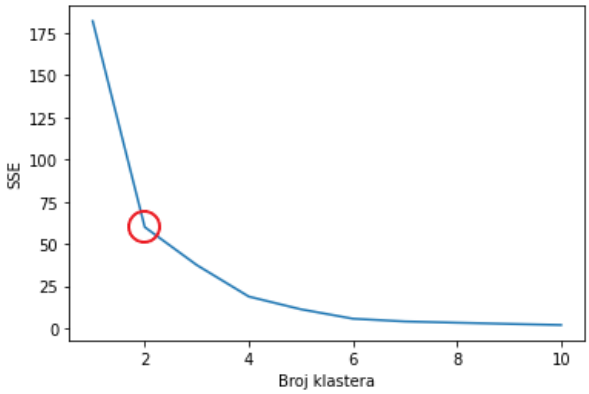
Slika 12: Druga iteracija metode lakta (Izvor: Autor)

To znači da ćemo u svakoj sumi kvadrata pogrešaka imati 8 pribrojnika, a konačni zbroj bit će jednak:

$$SSE = SSE_1 + SSE_2$$

Isti postupak radimo i u svim ostalim iteracijama. Svaka sljedeća iteracija daje manji konačan zbroj jer se broj točaka u klasterima smanjuje pa se smanjuje i zbroj kvadrata pogrešaka. S obzirom da u našem primjeru postoji 16 točaka, maksimalan broj iteracija je 16. U zadnjoj bismo iteraciji imali 16 klastera, u svakom po jednu točku. Konačan zbroj bio bi jednak 0 jer su udaljenosti točaka od centara klastera jednake 0. Zato nema smisla uzimati toliko velik broj iteracija.

Ako uzmemo da je broj iteracija jednak 10, na kraju ćemo imati 10 konačnih zbrojeva. Tih 10 suma prikazat ćemo na grafu zajedno s brojem klastera u svakoj iteraciji. To će izgledati ovako:



Slika 13: Optimalni broj klastera (Izvor: Autor)

Na grafu je crvenom kružnicom označen takozvani lakat. To je mjesto na kojem linija grafa prestaje naglo padati, a to znači da suma kvadrata pogrešaka prestaje s naglim padom. Možemo zaključiti da je optimalan broj klastera u našem primjeru jednak 2.

5. Prethodna slična istraživanja

U ovome poglavlju opisat ćemo istraživanje kojeg je proveo odjel podatkovno intenzivnog međunarodnog razvoja s kalifornijskog sveučilišta u Berkeleyju na čelu s profesorom Joshuom Blumenstockom. Istraživanje je to koje kombinira strojno učenje i ekonomski razvoj, a fokusira se na korištenje novih izvora podataka i metoda kako bi bolje razumjeli uzroke i posljedice siromaštva. Nazvali su ga „Borba protiv siromaštva uz pomoć podataka“ (eng. *Fighting poverty with data*). (Prema: [1])

Borba protiv siromaštva uz pomoć podataka

Prema [1], povod je ovog istraživanja revolucija velikih podataka, odnosno korištenje novih tehnologija i algoritama u rješavanju nekih problema s kojima se i dalje susrećemo u slabo razvijenim zemljama trećeg svijeta. S obzirom da u mnogim siromašnim zemljama ne postoji dovoljno podataka za dobar uvid u stvarno stanje, korištenje novih tehnologija definitivno može dovesti do nekih zaključaka čak i kad su podaci oskudni. U mnogim takvim državama podaci o popisu stanovništva stari su po nekoliko desetljeća pa je vrlo teško donositi odluke i ispravno alocirati resurse jer su podaci zastarjeli.

Istraživanje kojeg su proveli profesor Blumenstock i kolege imalo je cilj izmjeriti siromaštvo u Ruandi, afričkoj državi u središtu kontinenta. Profesor Blumenstock [1] tvrdi kako podaci telekomunikacijskih operatera jedne države mogu dati dobar uvid u ekonomske i socijalne veze koje postoje unutar države. Korištenje mobilnih telefona generira meta podatke u kojima se mogu pronaći razne geografske poveznice.

Njihov pristup bio je sljedeći: kombinirali su podatke telekomunikacijskih operatera s podacima koje su prikupili anketiranjem stvarnih 856 korisnika telekomunikacijskih usluga u Ruandi. Na taj su način nadopunili podatke o korištenju mobilnih usluga sa stvarnim izjavama o kvaliteti života korisnika. Takve sjedinjene podatke koristili su u kombinaciji s algoritmima strojnog učenja da bi odredili znakove koji ukazuju na siromaštvo i bogatstvo. (Prema [1])

Koristili su dvije metode kako bi otkrili skrivene veze u podacima:

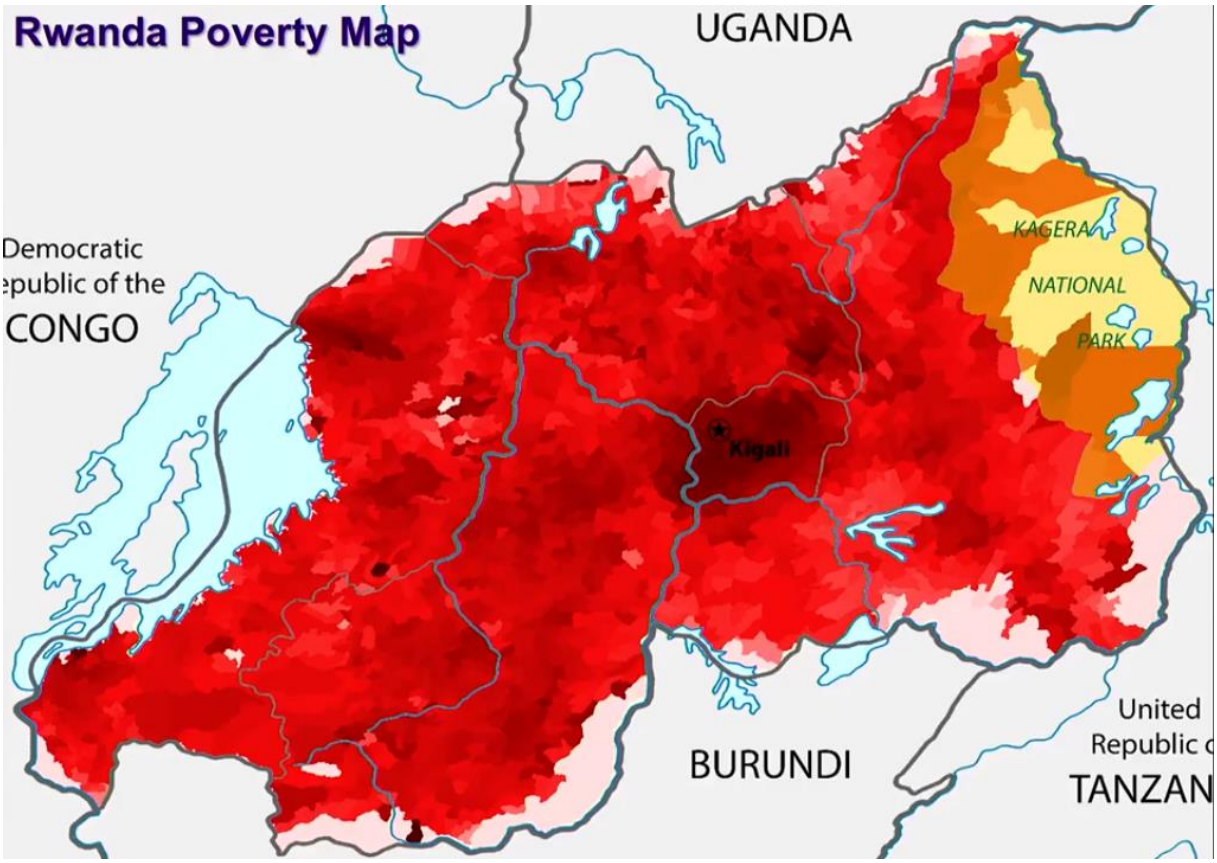
1. Izvlačenje obilježja (eng. *feature engineering*) – pretvaranje sirovih podataka u skup značajki (atributa), kvantitativnih metrika koje služe kao ulazni podaci za model
   - značajke mogu biti sljedeće: broj poziva po danu, broj međunarodnih kontakata, prosječno trajanje poziva vikendom, broj jedinstvenih korištenih odašiljača...

- koristili su algoritam koji automatski generira obilježja, deterministički konačni automat (eng. *deterministic finite automata*)

2. Strojno učenje pod nadzorom (eng. *supervized machine learning*) – skup podataka za treniranje modela s poznatim ulaznim i izlaznim vrijednostima koristi se kako bi naučili funkciju koja će preslikavati ulazne podatke u izlazne
   - ulazni podaci bili su oni koje su kreirali mobiteli, tj. značajke koje su samostalno kreirali tijekom izvlačenja obilježja
   - izlazni podaci bili su oni koje su prikupili anektiranjem; prilikom anketiranja provjeravali su financijsko stanje ispitanika što im je služilo kao temelj za izlazne vrijednosti
   - pomoću algoritma strojnog učenja odredili su koje značajke mogu koristiti u predviđanju bogatstva i siromaštva ljudi

Profesor Blumenstock [1] i ostali postavili su sljedeću hipotezu: „Omjer odlaznih i dolaznih poziva može predvidjeti bogatstvo korisnika mobitela“. Nakon što su kreirali model, dobili su uvid u važnost pojedinih značajki. Prema algoritmu kojeg su koristili u predikciji, značajka koja najviše doprinosi predikciji bogatstva ili siromaštva određenog korisnika u Ruandi bila je zbunjujuća (za čovjeka). Naime, algoritam tvrdi sljedeće: ako želimo saznati koliko je određeni korisnik bogat, moramo gledati entropiju SMS-ova po danima u tjednu za sve korisnikove kontakte i uzeti njihovu srednju vrijednost. Na engleskom, značajka glasi ovako: *Weighted average of all first-degree neighbour's „Day of week (DoW) entropy“ of outgoing SMS volume.* To je značajka koju je algoritam obilježio kao najvažniju. Međutim, postoji još nekoliko tisuća ostalih koje su manje bitne.

Model su zatim iskoristili za predikciju bogatstva svih korisnika za koje postoje podaci telekomunikacijskih operatera, njih 1.5 milijuna. Geografska estimacija bogatstva u Ruandi prema njihovu modelu prikazana je na slici ispod. Tamnija područja na mapi predstavljaju dijelove države gdje žive bogatiji ljudi.



Slika 14: Mapa siromaštva u Ruandi (Izvor: [1])

Prema [1], usporedba predikcijskih podataka s onima koje je objavio državni zavod za statistiku Ruande dokazuje kako je model jako precizan. Kvantitativna usporedba predikcijskih podataka i javno dostupnih podataka pokazuje da je korelacija jednaka 0.92.

6. Praktični dio

Praktični dio ovoga rada bit će primjena jedne on najzastupljenijih tehnika RDZ-a, a to je strojno učenje. U prošlom smo poglavlju napravili pregled sličnog istraživanja kakvog ćemo provesti u ovom dijelu rada. Naglasak je na korištenju javno dostupnih velikih skupova podataka iz kojih je potrebno kreirati model za predikciju. Predikciju će obavljati algoritam strojnog učenja. Koristit ćemo skup podataka koji je javno dostupan na Kaggleovoj stranici: [8]. Odabrani skup sadržava podatke o zaposlenicima neke kompanije, a prikupljao ga je odjel ljudskih resursa. Prikupljeni podaci služe za analizu odljeva zaposlenika. Cilj je saznati zašto zaposlenici napuštaju kompaniju i pokušati predvidjeti tko će kompaniju napustiti u budućnosti.

6.1. Opis problema

Prema [9], odljev zaposlenika ozbiljan je problem u mnogim industrijama. Svaka tvrtka ima svoju politiku odnosa prema zaposlenicima. Međutim, kad je riječ o otkazivanju suradnje između zaposlenika i poslodavca, mnoge tvrtke pokušava razumijeti tko odlazi i zašto. Radnici napuštaju organizacije iz mnogih razloga, naročito u informacijsko komunikacijskoj industriji. Gubitak zaposlenika velik je problem iz više razloga:

1. traženje adekvatne zamjene zahtjeva mnogo truda, naročio ako je riječ o iskusnom zaposleniku posebnih vještina
2. regrutiranje je skup i vremenski zahjevan proces
3. gubitak zaposlenika nepovoljno utječe na provođenje trenutnih projekata i pružanje usluga, a to može uzrokovati nezadovoljstvo klijenata i ostalih zainteresiranih stranki (eng. *stakeholders*)
4. treniranje novih zaposlenika kako bi postigli jednaku razinu stručnosti i produktivnosti može biti vremenski i financijski zahtjevno
5. gubitak radnika utječe na dohotke i ugled tvrtke.

S obzirom da je svaki zaposlenik jedinstven, razlozi otkazivanja suradnje su mnogobrojni. Ako u obzir uzmemo samo one zaposlenike koji svojom voljom otkazuju ugovor o radu, razlozi odlaska mogu biti razni: bolji posao, veća plaća, razne povlastice, bolji uvjeti rada, produktivnije radno okruženje, veće mogućnosti napredovanja u karijeri, pogodnija lokacija... Postoje i negativni razlozi koji uključuju sve već navedene razloge u negativnom

smislu, konflikte sa šefovima i ostalim zaposlenicima kao i ostale subjektivne razloge. (Prema: [9])

S poslovnog stajališta, jedan od kriterija uspješnosti je i zadržavanje postojećih zaposlenika. Korisno bi bilo razumijeti razloge odlaska i pomoću njih izraditi strategije zadržavanja zaposlenika i unaprijediti menadžment. Tu nam mogu pomoći prediktivni modeli koji bi se fokusirali na identifikaciju onih stavki koje najviše utječu na odljev. Prvi korak u izgradnji modela jest prikupljanje podataka. Naš je skup podataka već spreman, a u sljedećem ćemo ga poglavlju detaljno opisati.

6.2. Razumijevanje podataka

Ovaj korak u izgradnji modela još se naziva **eksplorativna analiza podataka**. Prema [13], taj proces daje uvid u karakteristike podataka (uzorke, trendove, *outliere*) i testiranje hipoteza koristeći deskriptivnu statistiku i vizualizaciju podataka.

Skup podataka preuzet je s Kaggleove stranice. Prema [10], Kaggle je internetska zajednica podatkovnih znanstvenika i praktikanata strojnog učenja. Korisnicima omogućuje pronalazak i objavljivanje raznih skupova podataka kao i proučavanje i izgradnju modela u web okruženju.

Skup podataka kojeg smo preuzeli spremljen je u CSV (eng. *comma separated values*) datoteku. Prema [11], to je tip tekstualne datoteke koja sadrži podatkovne zapise s vrijednostima odvojenim zarezom. Svaki redak u CSV datoteci je novi zapis iz skupa zapisa. Svaka vrijednost u pojedinom zapisu odvojena je zarezom.

Radna okolina u kojoj ćemo baratati podacima zove se Jupyter Notebook. To je interaktivna web platforma koja omogućava uvid u razvoj, dokumentiranje, izvršavanje koda i pregled rezultata. [12]

Svaki korak u analizi podataka potkrijepit ćemo slikom iz Jupyter bilježnice. Za početak ćemo uvesti besplatnu Pandas biblioteku koja nam je potrebna za rad s podacima. Ona omogućuje spremanje podataka u *dataframe* objekt, dvodimenzionalnu strukturu podataka za spremanje tabličnih podataka. Koristimo *read_csv* funkciju iz Pandas biblioteke za učitavanje datoteke o zaposlenicima.

```
import pandas
dataframe = pandas.read_csv('employee_data.csv')
```

Nakon što smo učitali podatke iz datoteke u dataframe objekt, pogledat ćemo kako on izgleda.

```
dataframe
```

	avg_monthly_hrs	department	filed_complaint	last_evaluation	n_projects	recently_promoted	salary	satisfaction	status	tenure
0	221	engineering	NaN	0.932868	4	NaN	low	0.829896	Left	5.0
1	232	support	NaN	NaN	3	NaN	low	0.834544	Employed	2.0
2	184	sales	NaN	0.788830	3	NaN	medium	0.834988	Employed	3.0
3	206	sales	NaN	0.575688	4	NaN	low	0.424764	Employed	2.0
4	249	sales	NaN	0.845217	3	NaN	low	0.779043	Employed	3.0
...	...	...	...	...	...	...	...	...	...	...
14244	178	IT	NaN	0.735865	5	NaN	low	0.263282	Employed	5.0
14245	257	sales	NaN	0.638604	3	NaN	low	0.868209	Employed	2.0
14246	232	finance	1.0	0.847623	5	NaN	medium	0.898917	Left	5.0
14247	130	IT	NaN	0.757184	4	NaN	medium	0.641304	Employed	3.0
14248	159	NaN	NaN	0.578742	3	NaN	medium	0.808850	Employed	3.0

14249 rows × 10 columns

Slika 15: Početni skup podataka (Izvor: Autor)

Možemo vidjeti da skup sadrži 14 249 redaka (entiteta) i 10 stupaca (atributa). Svaki redak predstavlja jednog zaposlenika (bivšeg ili sadašnjeg). Devet atributa opisuju zaposlenika, a deseti, u predzadnjem stupcu, njegov trenutni status zaposlenja.

Ako želimo malo detaljniji opis svakog stupca, koristimo funkciju *info*.

```
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14249 entries, 0 to 14248
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   avg_monthly_hrs    14249 non-null  int64
 1   department         13540 non-null  object
 2   filed_complaint    2058 non-null   float64
 3   last_evaluation    12717 non-null  float64
 4   n_projects         14249 non-null  int64
 5   recently_promoted  300 non-null    float64
 6   salary             14249 non-null  object
 7   satisfaction       14068 non-null  float64
 8   status             14249 non-null  object
 9   tenure             14068 non-null  float64
dtypes: float64(5), int64(2), object(3)
memory usage: 1.1+ MB
```

Slika 16: Detaljni opis stupaca (Izvor: Autor)

Postoji šest atributa kojima nedostaje vrijednost u nekim zapisima. Na tim je mjestima tip NaN koji označava da vrijednosti nema. Svaki atribut u kojem nedostaje zapis morat ćemo

posebno uzeti u obzir kad budemo pripremali podatke za modeliranje. Zasad ćemo napraviti opis svih postojećih atributa.

Tablica 1: Opis postojećih značajki

Naziv	Opis	Tip	Vrijednosti koje poprima
avg_monthly_hrs	prosječan broj radnih sati mjesečno	cjelobrojni	0 – n
department	odjel kojem zaposlenik pripada (ili je pripadao)	objekt	engineering, support, sales, IT, product, marketing, temp, procurement, finance, management, information_technology, admin
filed_complaint	je li zaposlenik podnio žalbu u zadnje tri godine	decimalni	0 – 1
last_evaluation	rezultat zadnje evaluacije zaposlenika (veći broj je bolji)	decimalni	0 – 1
n_projects	broj projekata na kojima zaposlenik radi (ili je radio)	cjelobrojni	0 – n
recently_promoted	je li zaposlenik promoviran u zadnje tri godine	decimalni	0, 1
salary	razina plaće s obzirom na ostale zaposlenike u tom odjelu	objekt	low, medium, high
satisfaction	zadovoljstvo zaposlenika kompanijom (veći broj je bolji)	decimalni	0 – 1
tenure	broj godina koje je zaposlenik proveo u kompaniji	decimalni	0 – n
status	trenutni status zaposlenja	objekt	left, employed

(Izvor: Autor)

Kao što smo već rekli, postoje dvije vrste zaposlenika: oni koji su u tvrtki i dalje zaposleni i oni koji su dali otkaz. Funkcijom *value_counts* možemo saznati broj pojavljivanja svih jedinstvenih vrijednosti u nekom stupcu.

```
dataframe['status'].value_counts()
```

```
Employed    10857
Left         3392
Name: status, dtype: int64
```

Slika 17: Broj jedinstvenih vrijednosti po stupcu (Izvor: Autor)

Od 14 249 zaposlenika, njih je 3 392 napustilo kompaniju (23,81%). Možemo reći da tvrtka ima problem jer skoro svaki četvrti čovjek odluči otići iz nje.

Sada ćemo skup podataka podijeliti na temelju statusa zaposlenja kako bismo usporedili karakteristike te dvije skupine. Koristit ćemo funkciju *groupby* kako bismo podatke iz *dataframea* grupirali po stupcu 'status'. Zatim koristimo funkciju *mean* za izračun aritmetičke sredine svih brojčanih atributa.

```
status = dataframe.groupby('status')
status.mean()
```

status	avg_monthly_hrs	filed_complaint	last_evaluation	n_projects	recently_promoted	satisfaction	tenure
Employed	197.700286	1.0	0.714479	3.755273	1.0	0.675979	3.380245
Left	206.502948	1.0	0.730706	3.833137	1.0	0.447500	3.869023

Slika 18: Aritmetička sredina svih atributa (Izvor: Autor)

Zaposlenici koji su dali otkaz imaju veći prosječni broj radnih sati, nešto bolje rezultate zadnje evaluacije, puno manju razinu zadovoljstva u odnosu na zaposlenike koji su ostali raditi.

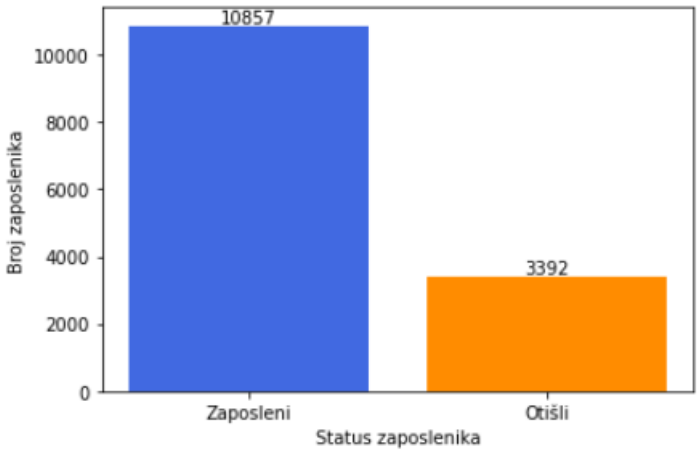
6.2.1. Vizualizacija podataka

Jupyter Notebook omogućava i vizualizaciju podataka pomoću određenih biblioteka. Mi ćemo koristiti Matplotlib biblioteku da bismo podatke prikazali u grafičkom obliku i Numpy biblioteku za rad s poljima.

```
import matplotlib.pyplot as plot
import numpy as np
```

Za početak ćemo prikazati broj zaposlenika po vrstama iz stupca 'status'. Koristimo funkciju *bar* iz biblioteke Matplotlib. Funkciji *bar* proslijedit ćemo tri parametra: jedinstvene vrijednosti iz stupca, broj pojavljivanja svake vrijednosti i listu s nazivima boja za stupce u grfu. Dodat ćemo i nazive svakoj osi grafa. Na vrhu svakog stupca u grafu dodat ćemo i tekst koji je zapravo broj ponavljanja. Na slici ispod možemo vidjeti kako graf izgleda u Jupyter bilježnici.

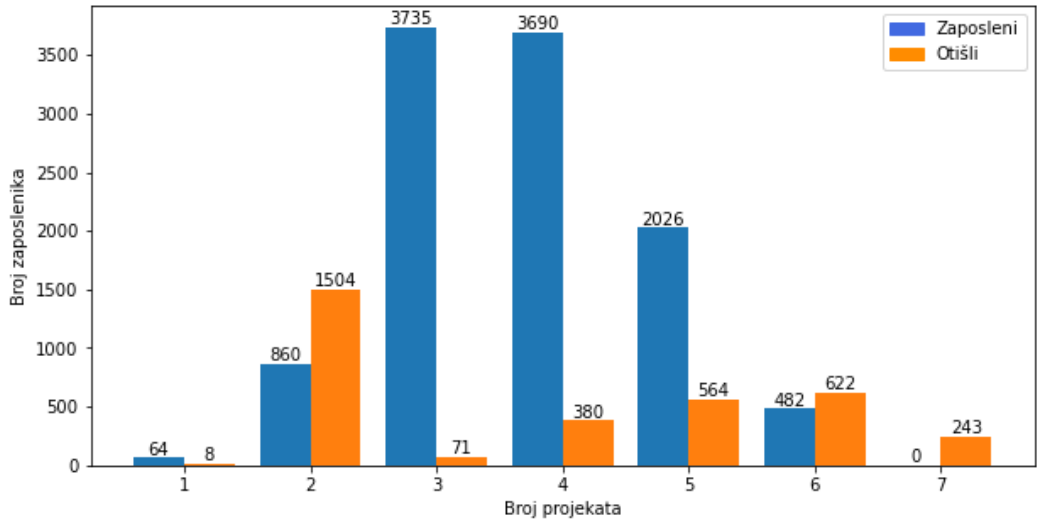
```
xAxisValues = dataframe['status'].unique()  # dohvaćanje svih unikatnih
vrijednosti iz stupca status
xAxisValues = xAxisValues[::-1]  # zamjena vrijednosti u polju
yAxisValues = dataframe['status'].value_counts()  # dohvaćanje brojeva
zaposlenika po statusu
plot.bar(xAxisValues, yAxisValues, color=['royalblue', 'darkorange'])
plot.xlabel('Status zaposlenika')
plot.ylabel('Broj zaposlenika')
plot.xticks(xAxisValues, ['Zaposleni', 'Otišli'])

# dodavanje brojeva iznad stupaca
for i in range(len(xAxisValues)):
    plot.text(i, yAxisValues[i], yAxisValues[i], ha="center", va="bottom")

plot.show()
```



Slika 19: Broj zaposlenika po statusu (Izvor: Autor)

Na sličan način prikazat ćemo i broj zaposlenika po broju projekata.

```
xAxisValues = dataframe['n_projects'].unique()  # dohvaćanje svih unikatnih
vrijednosti iz stupca n_projects
xAxisValues.sort()  # sortiranje vrijednosti uzlazno
barWidth = 0.4  # širina svakog stupca u grafu
xAxisValuesShifted = [i+barWidth for i in xAxisValues]  # svakoj vrijednosti
dodajemo širinu stupca i spremamo u novo polje

# dataframe s trenutnim zaposlenicima
dataframeEmployed = dataframe.groupby('status').get_group("Employed")
# dataframe sa zaposlenicima koji su otišli
dataframeLeft = dataframe.groupby('status').get_group("Left")

yAxisValuesEmployed = []
yAxisValuesLeft = []

# za svaku vrijednost iz polja xAxisValues (n projekata) spremamo broj
ponavljanja u stupcu n_projects, tj. broj zaposlenika s toliko projekata;
# to radimo za oba dataframea
for i in range(len(xAxisValues)):
    try:
      yAxisValuesEmployed.append(dataframeEmployed['n_projects'].value_count
      s()[xAxisValues[i]])
    except:
      yAxisValuesEmployed.append(0)
    try:
      yAxisValuesLeft.append(dataframeLeft['n_projects'].value_counts()[xAxi
      sValues[i]])
    except:
      yAxisValuesLeft.append(0)

plot.bar(xAxisValues, yAxisValuesEmployed, barWidth)  # postavljanje stupaca
za trenutne zaposlenike
plot.bar(xAxisValuesShifted, yAxisValuesLeft, barWidth)  # postavljanje
stupaca za ostale zaposlenike
plot.xlabel('Broj projekata')
plot.ylabel('Broj zaposlenika')

colors = {'Zaposleni':'royalblue', 'Otišli':'darkorange'}
labels = list(colors.keys())
handles = [plot.Rectangle((0,0),1,1, color=colors[label]) for label in
labels]
plot.legend(handles, labels)

plot.xticks(xAxisValues+barWidth/2, xAxisValues) # pomicanje labela na x
osi; drugi parametar su vrijednosti

# dodavanje brojeva nad stupcima
for i in range(len(xAxisValues)):
    plot.text(i+1, yAxisValuesEmployed[i], yAxisValuesEmployed[i],
              ha="center", va="bottom")
    plot.text(i+1+barWidth, yAxisValuesLeft[i], yAxisValuesLeft[i],
              ha="center", va="bottom")

plot.show()
```

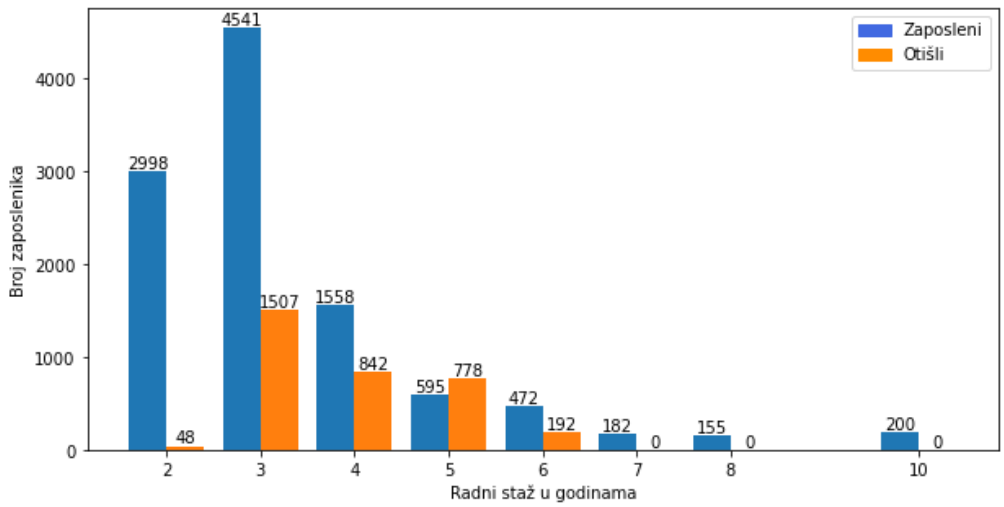


Slika 20: Broj zaposlenika po broju projekata (Izvor: Autor)

Većina zaposlenika radi na 3 ili 4 projekta istovremeno. Možemo reći da istovremeni rad na 3 ili 4 projekta u toj kompaniji radnicima ne predstavlja problem. Međutim, kako se broj projekata povećava, sve je veća vjerojatnost da će radnici dati otkaz. Ako pogledamo stupce gdje je broj projekata jednak 6, vidimo da je broj radnika koji su otišli veći. Brojka 7 predstavlja prekretnicu. Naime, svi zaposlenici koji su istovremeno radili na 7 projekata dali su otkaz. Preopterećenost poslom može biti razlog njihova odlaska.

Nadalje, prikazat ćemo broj zaposlenika po vremenu provedenom u tvrtki.

```
xAxisValues = dataframe['tenure'].dropna().unique()  # dohvaćanje svih
unikatnih vrijednosti iz stupca tenure
xAxisValues.sort()  # sortiranje vrijednosti uzlazno
barWidth = 0.4  # širina svakog stupca u grafu
xAxisValuesShifted = [i+barWidth for i in xAxisValues]  # svakoj vrijednosti
dodajemo širinu stupca i spremamo u novo polje

# dataframe s trenutnim zaposlenicima
dataframeEmployed = dataframe.groupby('status').get_group("Employed")
# dataframe sa zaposlenicima koji su otišli
dataframeLeft = dataframe.groupby('status').get_group("Left")

yAxisValuesEmployed = []
yAxisValuesLeft = []

# za svaki radni staž iz polja xAxisValues spremamo broj ponavljanja u
stupcu tenure, tj. broj zaposlenika s toliko godina radnog staža;
# to radimo za oba dataframea
for i in range(len(xAxisValues)):
    try:
      yAxisValuesEmployed.append(dataframeEmployed['tenure'].value_counts()[
      xAxisValues[i]])
    except:
      yAxisValuesEmployed.append(0)
    try:
      yAxisValuesLeft.append(dataframeLeft['tenure'].value_counts()[xAxisVal
      ues[i]])
    except:
      yAxisValuesLeft.append(0)

plot.figure(figsize=(10,5))
plot.bar(xAxisValues, yAxisValuesEmployed, barWidth)  # postavljanje stupaca
za trenutne zaposlenike
plot.bar(xAxisValuesShifted, yAxisValuesLeft, barWidth)  # postavljanje
stupaca za ostale zaposlenike
plot.xlabel('Radni staž u godinama')
plot.ylabel('Broj zaposlenika')

colors = {'Zaposleni':'royalblue', 'Otišli':'darkorange'}
labels = list(colors.keys())
handles = [plot.Rectangle((0,0),1,1, color=colors[label]) for label in
labels]
plot.legend(handles, labels)

plot.xticks(xAxisValues+barWidth/2, xAxisValues.astype(int)) # pomicanje
labela na x osi; drugi parametar su vrijednosti

# dodavanje brojeva iznad stupaca
for i in range(len(xAxisValues)):
    plot.text(xAxisValues[i], yAxisValuesEmployed[i],
              yAxisValuesEmployed[i], ha="center", va="bottom")
    plot.text(xAxisValues[i]+barWidth, yAxisValuesLeft[i],
              yAxisValuesLeft[i], ha="center", va="bottom")

plot.show()
```

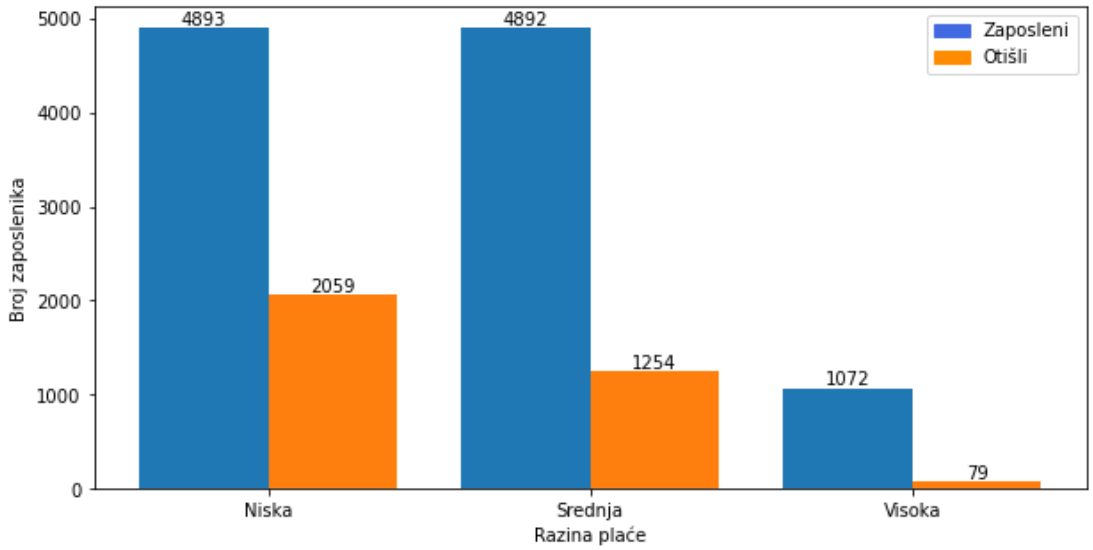


Slika 21: Broj zaposlenika po radnom stažu (Izvor: Autor)

Najviše zaposlenika ima od 2 do 4 godine radnog staža u tvrtki. Zaposlenici koji daju otkaz najčešće imaju 3 godine radnog staža. Jedan od razloga za takvo stanje može biti to što ljudi danas često mijenjaju posao. To su uglavnom mladi ljudi željni promjene, konstantnog napretka ili razvijanja različitih vještina. Na grafu vidimo da s povećanjem radnog staža, broj otkaza pada. Ni jedan zaposlenik koji je u tvrtki 7 i više godina nije dao otkaz. To su već uhodani i iskusni zaposlenici.

Prikazat ćemo i broj zaposlenika po plaći koju primaju (ili su primali).

```
xAxisValuesUnique = dataframe['salary'].dropna().unique()  # dohvaćanje svih
unikatnih vrijednosti iz stupca salary
xAxisValues = np.arange(len(xAxisValuesUnique))  # prebacivanje tekstualnih
u brojčane vrijednosti
xAxisValues = [i+0.5 for i in xAxisValues]  # povećavanje vrijednosti
xAxisValues = np.asarray(xAxisValues)  # pretvaranje u polje
barWidth = 0.4  # širina svakog stupca u grafu
xAxisValuesShifted = [i+barWidth for i in xAxisValues]  # svakoj vrijednosti
dodajemo širinu stupca i spremamo u novo polje

# dataframe s trenutnim zaposlenicima
dataframeEmployed = dataframe.groupby('status').get_group("Employed")
# dataframe sa zaposlenicima koji su otišli
dataframeLeft = dataframe.groupby('status').get_group("Left")

yAxisValuesEmployed = []
yAxisValuesLeft = []

for i in range(len(xAxisValues)):
    try:
      yAxisValuesEmployed.append(dataframeEmployed['salary'].value_counts()[
      xAxisValuesUnique[i]])
    except:
      yAxisValuesEmployed.append(0)
    try:
      yAxisValuesLeft.append(dataframeLeft['salary'].value_counts()[xAxisVal
      uesUnique[i]])
    except:
      yAxisValuesLeft.append(0)

plot.figure(figsize=(10,5))
plot.bar(xAxisValues, yAxisValuesEmployed, barWidth)  # postavljanje stupaca
za trenutne zaposlenike
plot.bar(xAxisValuesShifted, yAxisValuesLeft, barWidth)  # postavljanje
stupaca za ostale zaposlenike
plot.xlabel('Razina plaće')
plot.ylabel('Broj zaposlenika')

colors = {'Zaposleni':'royalblue', 'Otišli':'darkorange'}
labels = list(colors.keys())
handles = [plot.Rectangle((0,0),1,1, color=colors[label]) for label in
labels]
plot.legend(handles, labels)

# dodavanje brojeva iznad stupaca
for i in range(len(xAxisValues)):
    plot.text(xAxisValues[i], yAxisValuesEmployed[i],
              yAxisValuesEmployed[i], ha="center", va="bottom")
    plot.text(xAxisValues[i]+barWidth, yAxisValuesLeft[i],
              yAxisValuesLeft[i], ha="center", va="bottom")

plot.xticks(xAxisValues+barWidth/2, ['Niska', 'Srednja', 'Visoka'])

plot.show()
```



Slika 22: Broj zaposlenika prema razini plaće (Izvor: Autor)

Podjednak broj trenutnih zaposlenika ima nisku i srednju razinu plaće. Među ljudima koji su dali otkaz, najviše njih imali su nisku razinu plaće, a najmanje visoku razinu. Niska plaća definitivno je jedan od razloga zbog kojih se zaposlenici odluče tražiti drugu tvrtku.

6.3. Priprema podataka

Da bismo mogli stvoriti model za predikciju, potrebno je pripremiti podatke iz početnog skupa. Algoritmi strojnog učenja zahtijevaju uređen skup podataka u kojem nema nedostajućih vrijednosti i u kojem su svi podaci numeričkog tipa.

Prvi korak u pripremi podataka bit će izbacivanje duplikata iz skupa jer nam takvi zapisi neće biti korisni. To radimo pomoću funkcije *drop_duplicates* iz biblioteke Pandas.

```
dataframe.drop_duplicates(inplace=True)

dataframe.shape()
```

```
(14221, 10)
```

Početni skup sadržavao je 14 249 zapisa što znači da smo izbacili 28 zapisa. Sljedeći nam je korak prebaciti sve podatke u numerički tip. Tijekom eksplorativne analize podataka saznali smo da naš skup podataka sadrži tri tipa podataka: cjelobrojni, decimalni i objekt koji sadrži nekoliko kategorija. Potonji tip podataka prebacit ćemo u numerički.

Prvi je na redu atribut *department*. Sljedećom komandom saznat ćemo sve jedinstvene vrijednosti tog atributa.

```
dataframe.department.unique()
```

```
array(['engineering', 'support', 'sales', 'IT', 'product', 'marketing',
       'temp', 'procurement', 'finance', nan, 'management',
       'information_technology', 'admin'], dtype=object)
```

Možemo vidjeti kako postoje nedostajuće vrijednosti u stupcu *department*. Potrebno je izbaciti sve redove koji nemaju vrijednost u tom stupcu.

```
dataframe = dataframe[dataframe['department'].notna()]
```

Nakon toga promijenit ćemo svaku vrijednost *information_technology* u *IT* jer je to isti odjel.

```
dataframe['department'].replace({'information_technology':
'IT'},inplace=True)]
```

Sada nam ostaje 11 različitih vrijednosti u stupcu *department.* Svaku ćemo vrijednost zamijeniti jednom brojkom koja će predstavljati određeni odjel. Krenut ćemo od brojke 0 koja će predstavljati odjel *engineering*, a završit ćemo s brojkom 10 koja će predstavljati odjel *admin*.

```
dataframe['department'].replace({'engineering': 0, 'support': 1, 'sales': 2,
                        'IT': 3, 'product': 4, 'marketing': 5, 'temp': 6,
                        'procurement': 7, 'finance': 8, 'management': 9,
                        'admin': 10,},inplace=True)

dataframe.department.unique()
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Na sličan ćemo način modificirati i vrijednosti u stupcu *salary.* Neka nam broj 0 predstavlja vrijednost *low*, broj 1 vrijednost *medium*, a broj 2 vrijednost *high.*

```
dataframe['salary'].replace({'low': 0, 'medium': 1, 'high': 2},
inplace=True)
```

Još nam ostaje jedan ti podataka kojeg moramo prebaciti u numerički, a to je *status*. Neka nam svaki zapis koji u tom stupcu sadrži *Left* poprimi vrijednost 1, a tamo gdje se pojavljuje *Employed* postavit ćemo 0. Ovaj stupac bit će ključan u prediktivnom modelu jer ćemo njega predviđati.

```
dataframe[„status“] = pandas.get_dummies(dataframe.status).Left
```

Stupac *filed_complaint* specifičan je po tome što nema vrijednost za zaposlenike koji nisu podnijeli žalbu. Oni koji su podnijeli žalbu, u tom stupcu imaju vrijednost 1.0. U stupcu *recently_promoted* isti slučaj, nedostaju podaci za zaposlenike koji nisu nedavno promovirani.

U oba slučaja nedostajuće vrijednosti zamijenit ćemo s nulom.

```
dataframe.filed_complaint.fillna(0, inplace=True)
dataframe.recently_promoted.fillna(0, inplace=True)
```

Još postoje zapisi koji nemaju vrijednosti u stupcima *last_evaluation*, *satisfaction* i *tenure*. Ti nam zapisi ne mogu biti korisni pri kreiranju modela pa ćemo ih izbaciti jednom komandom.

```
dataframe.dropna(inplace=True)
```

Na kraju nam ostaje skup podataka koji sadrži 12 075 zapisa i 10 atributa. U svim zapisima postoji vrijednost za svaki atribut.

```
print(dataframe.shape)
dataframe.isnull().sum()
```

```
(12075, 10)

avg_monthly_hrs      0
department           0
filed_complaint      0
last_evaluation      0
n_projects           0
recently_promoted    0
salary               0
satisfaction         0
status               0
tenure               0
dtype: int64
```

Ovako pripremljen skup podataka spremit ćemo u novu CSV datoteku koju ćemo koristiti za izgradnju modela.

```
dataframe.to_csv('employee_churn_clean.csv', index=None)
```

6.4. Klaster analiza

Klasteriranje je metoda nenadziranog strojnog učenja u kojem ciljna varijabla ne postoji. S obzirom da naš skup podataka sadrži ciljnu varijablu (status), klasteri već postoje: skup ljudi koji su dali otkaz i skup ljudi koji su zaposleni. Međutim, ako želimo saznati nešto više o ljudima koji su dali otkaz, ciljnu varijablu moramo izbaciti. Skup podataka koji nam ostaje neće biti klasificiran pa nam je ipak potreban algoritam za klasteriranje.

Klaster analizu podataka provest ćemo algoritmom k-sredina. S obzirom da su opće zadovoljstvo radom i broj radnih sati dvije vrlo bitne stavke za svakog zaposlenika, za analizu ćemo koristiti dva atributa koja ih opisuju, a to su *satisfaction* i *avg_monthly_hrs*. Grupirat ćemo zaposlenike s obzirom na zadovoljstvo i prosječan broj radnih sati.

Podaci u dataframeu trenutno sadržavaju sve atribute, a to nam u ovoj analizi nije potrebno. Ostavit ćemo samo podatke iz stupaca *satisfaction* i *avg_monthly_hrs*. U obzir ćemo uzeti samo one zaposlenike koji su otišli.

```
dataframeLeft = dataframe[['satisfaction',
'avg_monthly_hrs']][dataframe.status == 'Left']
dataframeLeft
```

	satisfaction	avg_monthly_hrs
0	0.829896	221
10	0.403552	147
13	0.090343	290
15	0.849667	258
17	0.893365	252
...	...	...
14234	0.493401	157
14236	0.953847	242
14239	0.669866	242
14240	0.341842	131
14246	0.898917	232

3392 rows × 2 columns

Slika 23: Skup zaposlenika koji su dali otkaz (Izvor: Autor)

Nakon restrukturiranja dataframea, ostaju nam dva atributa. Međutim, da bismo podatke mogli analizirati, ne smiju nedostajati vrijednosti. Metodom *dropna* očistit ćemo dataframe od takvih zapisa.

```
dataframeLeft.dropna(inplace=True)
dataframeLeft.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3367 entries, 0 to 14246
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   satisfaction     3367 non-null   float64
 1   avg_monthly_hrs  3367 non-null   int64
dtypes: float64(1), int64(1)
memory usage: 78.9 KB
```

Slika 24: Skup podataka bez nedostajućih vrijednosti (Izvor: Autor)

Nakon ovog koraka, ostaju nam 3367 zapisa bez nedostajućih vrijednosti. Budući da je raspon vrijednosti u stupcu *avg_monthly_hrs* mnogo veći od raspona vrijednosti iz stupca *satisfaction*, potrebno je skalirati varijablu *avg_monthly_hrs*. Prema [17], skaliranje značajki koristi se za normalizaciju značajki podataka. Izvodi se tijekom predobrade podataka kako bi algoritmi strojnog učenja mogli normalno raditi. Širok raspon jedne značajke može bitno utjecati na funkcije strojnog učenja.

S obzirom da vrijednosti u stupcu *satisfaction* imaju raspon od 0 do 1, takav ćemo raspon napraviti i u stupcu *avg_monthly_hrs.* U suprotnom, podaci na grafu izgledat će nepravilno, a algoritam za klasteriranje neće dobro odraditi svoj posao.

Prema [17], najjednostavnija metoda za skaliranje naziva se **min-max normalizacija**. Ona skalira postojeći raspon značajki na raspon [0,1] ili raspon [-1,1]. Mi ćemo koristiti raspon [0,1] . Da bismo skalirali podatke iz stupca *avg_monthly_hrs*, potrebno je saznati maksimalnu i minimalnu vrijednost.

```
minHrs = dataframeLeft['avg_monthly_hrs'].min()
minHrs
```

```
126
```

```
maxHrs = dataframeLeft['avg_monthly_hrs'].max()
maxHrs
```

```
310
```

Sada kad znamo da se broj sati kreće od 126 do 310, broj sati za svakog zaposlenika računat ćemo tako što ćemo od njegova broja sati oduzeti 126 i rezultat podijeliti sa 184 (razlika između maksimalnog i minimalnog broja sati).

```
dataframeScaled = dataframeLeft.copy()
dataframeScaled['avg_monthly_hrs'] = (dataframeScaled['avg_monthly_hrs'] -
minHrs) / (maxHrs - minHrs)

dataframeScaled
```

	satisfaction	avg_monthly_hrs
0	0.829896	0.516304
10	0.403552	0.114130
13	0.090343	0.891304
15	0.849667	0.717391
17	0.893365	0.684783
...	...	...
14234	0.493401	0.168478
14236	0.953847	0.630435
14239	0.669866	0.630435
14240	0.341842	0.027174
14246	0.898917	0.576087

3367 rows × 2 columns

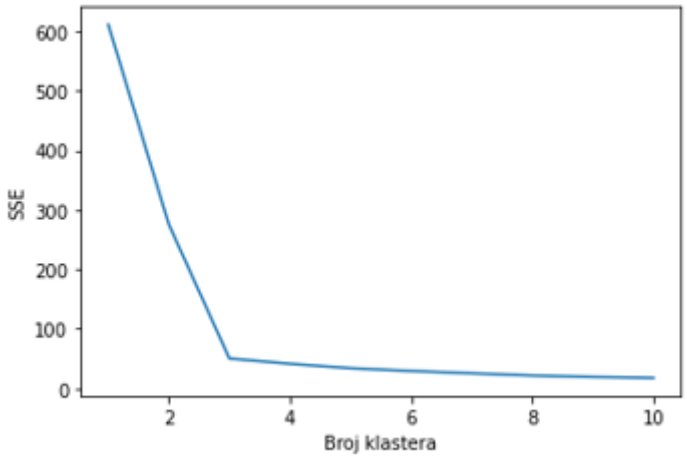
Slika 25: Normalniziran skup podataka (Izvor: Autor)

Podaci su sada spremni za klasteriranje. Kao što smo spomenuli u opisu algoritma, broj klastera je proizvoljan. Optimalan broj odredit ćemo **metodom lakta** koju smo opisali u jedom od prijašnjih poglavlja. Za njeno korištenje potrebna nam je biblioteka Scikit-learn. [16]

```
from sklearn.cluster import KMeans
```

Broj iteracija postavit ćemo na 10. U svakoj iteraciji stvaramo model pomoću *KMeans* metode kojoj prosljeđujemo broj klastera. Nakon toga koristimo funkciju *fit* za klasteriranje. Prema [16], zbroj kvadrata pogrešaka dobit ćemo iz atributa *inertia_* stvorenog modela. Zbroj kvadrata pogrešaka spremamo u novo polje koje će nam služiti za prikaz na grafu.

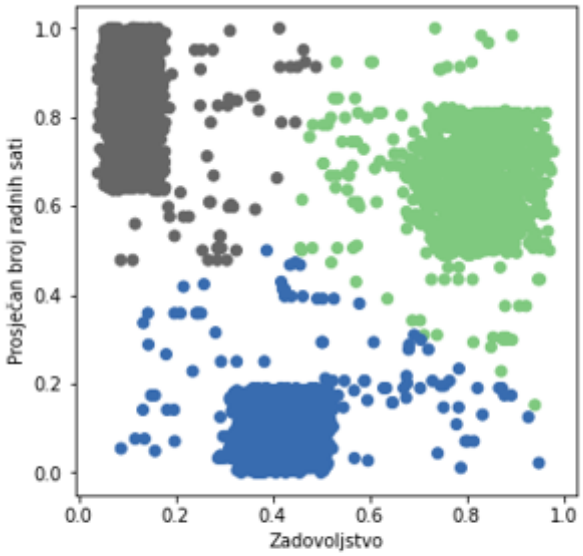
```
kRange = range(1,11)
sse = []
for k in kRange:
    km = KMeans(n_clusters=k)
    km.fit(dataframeScaled)
    sse.append(km.inertia_)

plot.xlabel('Broj klastera')
plot.ylabel('SSE')
plot.plot(kRange,sse)
```



Slika 26: Zbroj kvadrata pogrešaka prema broju klastera (Izvor: Autor)

Na grafu je jasno vidljiv lakat. Zaključujemo da je optimalan broj klastera jednak 3. S tom brojkom možemo nastaviti klasteriranje. Opet koristimo *fit* funkciju, ovaj put s pripremljenim dataframeom. Svaki podatak u dataframeu smješta se u jedan od tri klastera.

```
kmeans = KMeans(n_clusters = 3, random_state = 0).fit(dataframeScaled)
dataframeScaled[‘cluster’] = kmeans.labels_

plot.figure(figsize=(5,5))
plot.scatter(dataframeScaled['satisfaction'],
dataframeScaled['avg_monthly_hrs'], c=dataframeScaled['cluster'],
cmap='Accent')
plot.xlabel('Zadovoljstvo')
plot.ylabel('Prosječan broj radnih sati')
plot.show()

print(f"Broj zaposlenika po klasterima:" +
„\n-----------------------------„ +
„\n1.klaster (plava boja): „ +
str(dataframeScaled['cluster'].value_counts()[1]) +
„\n2.klaster (siva boja): „ +
str(dataframeScaled['cluster'].value_counts()[2]) +
„\n3.klaster (zelena boja): „ +
str(dataframeScaled['cluster'].value_counts()[0]))
```




```
Broj zaposlenika po klasterima:
-----------------------------
1.klaster (plava boja): 1559
2.klaster (siva boja): 898
3.klaster (zelena boja): 910
```

Slika 27: Tri klastera zaposlenika koji su dali otkaz (Izvor: Autor)

Na grafu iznad svaki je klaster obojen drugačijom bojom. Jasno je vidljivo kako se točke gomilaju na tri različita mjesta. Možemo zaključiti kako postoje tri grupe ljudi koji su napustili kompaniju:

1. umjereno zadovoljni zaposlenici s niskim prosjekom broja sati
2. nezadovoljni zaposlenici s visokim prosjekom broja sati
3. jako zadovoljni zaposlenici s poviđenim prosjekom broja sati

6.5. Izgradnja modela za predikciju

Za izgradnju modela koristit ćemo biblioteku Sklearn. Prema [16], *tree* modul iz te biblioteke sadržava modele za klasifikaciju i regresiju bazirane na stablima. Mi ćemo koristiti model *DecisionTreeClassifier*.

```
from sklearn import tree
```

6.5.1. Uvoz podataka

Prvi nam je korak učitati pripremljen skup podataka u dataframe.

```
dataframe = pandas.read_csv('employee_churn_clean.csv')
```

Sada je potrebno podijeliti dataframe na dva dijela. Prvi dio zove se matrica dizajna i sadržava sve značajke koje će nam služiti za predikciju, a drugom dijelu bit će samo značajka za predikciju (stupac *status*). Prema [18], matrica dizajna sadrži podatke o neovisnim varijablama koji služe za objašnjenje podataka o ovisnoj varijabli.

```
design_matrix = dataframe.drop('status', axis=1)
targets = dataframe.status
```

6.5.2. Skupovi za treniranje i testiranje

Sljedeći nam je korak stvoriti skup podataka za treniranje i za testiranje modela. Prema [19], svrha nadziranog strojnog učenja je kreiranje modela koji precizno pretvaraju ulazne varijable (prediktore) u izlazne (predikcije). Način mjerenja preciznosti modela razlikuje se s obzirom na problem koji se pokušava riješiti. Svaki način mjerenja mora biti

nepristran, a to znači da nije moguće evaluirati sposobnost predikcije modela s istim podacima na kojima je model treniran. Model se evaluira sa svježim podacima, onim koje dosad nije vidio. Zato se skup podataka mora podijeliti prije korištenja.

Tu će nam poslužiti metoda *train_test_split* iz biblioteke Sklearn. Prema [19], metoda *train_test_split* dijeli polja ili matrice u nasumične podskupove za treniranje i testiranje. Proslijedit ćemo joj objekte koji sadrže podatke koje želimo podijeliti, a to su: dvodimenzionalno polje s ulaznim vrijednostima i jednodimenzionalno polje s izlaznim vrijednostima. Također ćemo proslijediti parametar *test_size* koji određuje veličinu skupa za testiranje, a ujedno određuje i veličinu skupa za treniranje.

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(design_matrix, targets,
test_size=0.33)
```

Metoda nam vraća 4 polja:

1. x_train: dio za treniranje od prvog polja (*design_matrix*)
2. x_test: dio za testiranje od prvog polja (*design_matrix*)
3. y_train: dio za treniranje od drugog polja (*targets*)
4. y_ test: dio za testiranje od drugog polja (*targets*)

6.5.3. Stablo odlučivanja

S podacima za treniranje i testiranje možemo nastaviti izgradnju modela. To će biti klasifikacijsko stablo odlučivanja (eng. *DecissionTreeClassifier*). Postoje mnogi parametri koje možemo proslijediti klasifikatoru prilikom stvaranja. Oni mogu bitno utjecati na izgled stabla i preciznost modela. S obzirom da ne želimo kompleksno stablo s velikim brojem grana, koristit ćemo parametar *ccp_alpha*. Prema [16], to je parametar kompleksnosti koji se koristi za takozvano obrezivanje stabla odlučivanja. Primarna mu je svrha smanjiti veličinu stabla. Ako ga definiramo, algoritam će prilikom odabira podstabla uspoređivati parametar s kompleksnosti podstabla.

Vrijednost koju ćemo proslijediti nije nasumična jer ona bitno utječe na preciznost modela u predikciji. Prema [16], koristi se funkcija klasifikatora *cost_complexity_pruning_path* koja, između ostalog, vraća sve moguće vrijednosti tog parametra. Odabrali smo onu vrijednost koja slabo utječe na preciznost modela, a uvelike smanjuje kompleksnost stabla. U prilogu ovog rada nalazi se datoteka Jupyter bilježnice u kojoj smo, između ostalog, došli do idealne

vrijednosti parametra. S obzirom da nam parametar *ccp_alpha* služi samo za smanjenje kompleksnosti stabla, postupak odabira nećemo detaljno opisivati.

```
classifier = tree.DecisionTreeClassifier(ccp_alpha=0.01)
```

Funkcija koja trenira model zove se *fit*. Prosljeđujemo joj skupove podataka za treniranje nad kojima se model uči pretvarati ulazne varijable iz skupa *x_train* u izlaznu varijablu iz skupa *y_train.*

```
classifier.fit(x_train, y_train)
```

Istrenirani model potrebno je testirati. Pozivamo funkciju *predict* nad našim modelom i prosljeđujemo joj jedan parametar, a to je skup podataka za testiranje *x_test.* Njegova je struktura ista strukturi skupa *x_train*, ali podaci nisu. Model sada nema odgovore pa radi predikciju nad testnim podacima. Rezultat predikcije je polje koje ima istu strukturu kao skup *y_test.*

```
predictions = classifier.predict(x_test)

predictions
```

```
array([1, 0, 0, ..., 0, 1, 1])
```

Nakon što je model obavio predikciju, potrebno je usporediti stvarne vrijednosti iz skupa *y_test* s vrijednostima iz skupa *predictions.* Provjeru radimo pomoću funkcije *accuracy_score* iz Sklearn biblioteke. Prosljeđujemo joj naše skupove podataka.

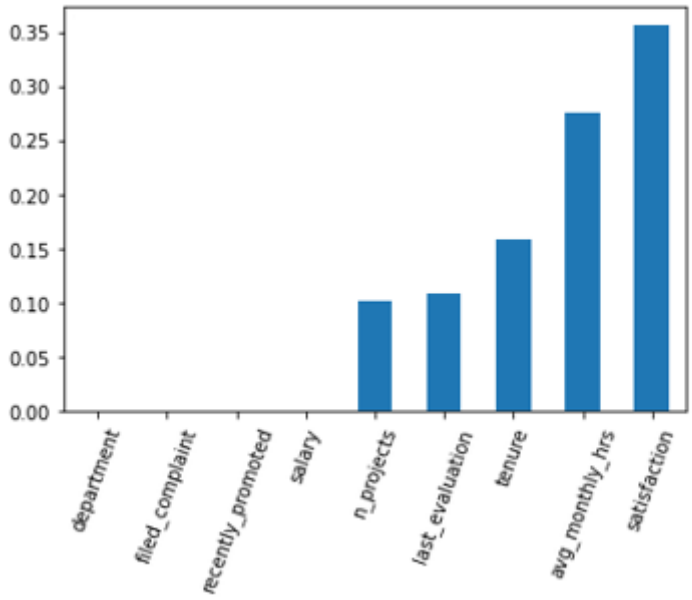
```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, predictions)
```

```
0.9513174404015057
```

Preciznost modela na testnom skupu podataka iznosi 95,13%. Možemo reći da je model jako precizan u predikciji i nema problema s prekomjernim prilagođavanjem podacima za treniranje.

Osim preciznosti modela, zanima nas i važnost pojedinih značajki u predikciji. Prema [20], algoritmi stabla odlučivanja nude ocjenu važnosti pojedinih značajki s obzirom na njihovu ulogu u samom postupku izgradnje stabla. Nakon što je model obavio treniranje, on nam nudi atribut *feature_importances* u kojem se nalazi relativna ocjena svake pojedine značajke. Važnost značajki prikazat ćemo na stupičastom grafu.

```
feature_names = design_matrix.columns
feature_importance = pandas.DataFrame(classifier.feature_importances_, index
= feature_names).sort_values(0)

feature_importance.plot(kind='bar')
plot.xticks(rotation = 70)
```



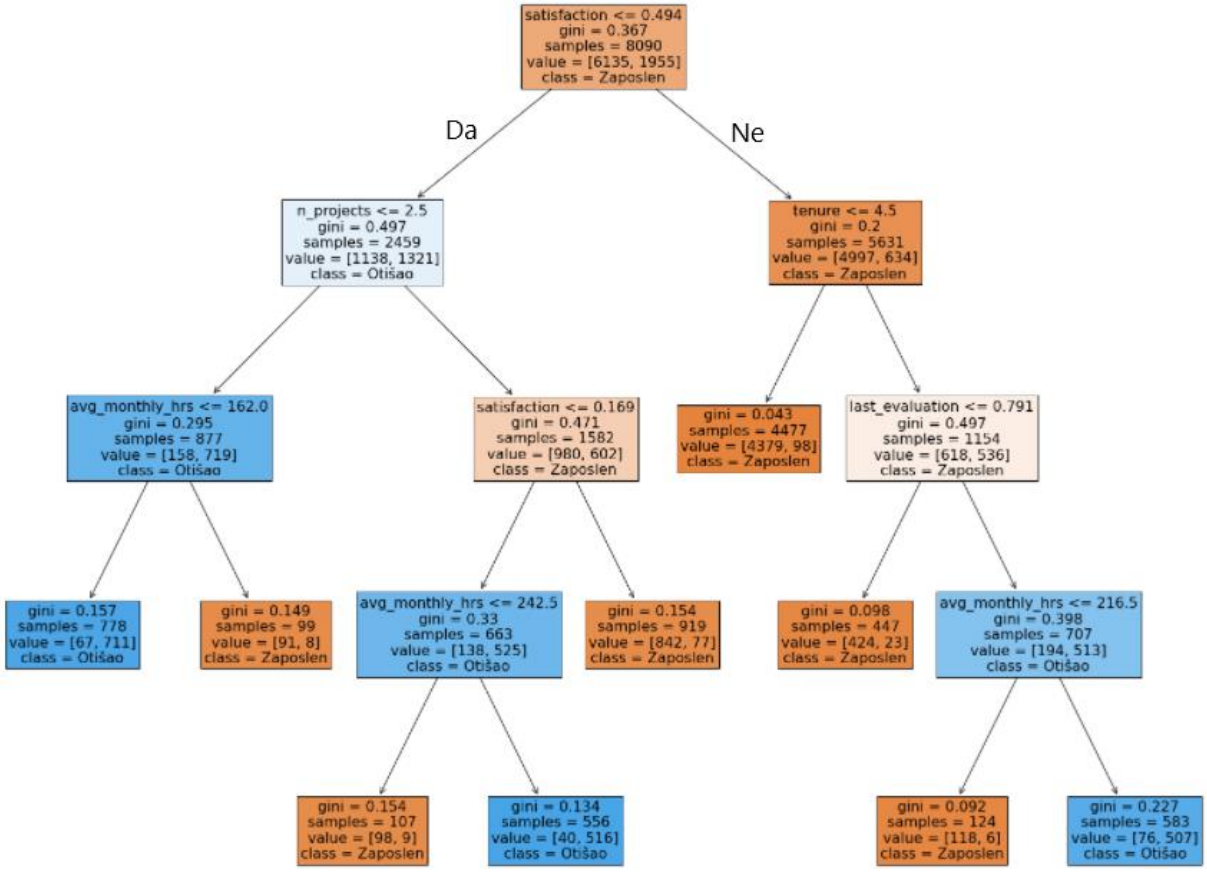
Slika 28: Važnost značajki u izgradnji modela (Izvor: Autor)

Od 9 značajki koje se koriste u predikciji, zadovoljstvo zaposlenika (*satisfaciton*) najviše utječe na predikciju odlaska. Prilikom izgradnje stabla, bitne su i sljedeće značajke: prosječan broj radnih sati (*avg_monthly_hrs*), radni staž (*tenure*), zadnja evaluacija (*last_evaluation*) i broj projekata (*n_projects*). Ostale značajke ne pridonose konačnom rezultatu u ovom modelu. Međutim, to ne znači da te značajke nikad neće biti korištene, nego samo da nisu bile korištene u izgradnji ovog specifičnog modela s ovakvo specifičnim skupovima za treniranje i testiranje.

Biblioteka Sklearn omogućuje nam i vizualni prikaz stabla odlučivanja. Koristimo funkciju *plot_tree* kojoj prosljeđujemo nekoliko parametara: *classifier* - trenirani model (stablo),

feature_names - nazivi svih značajki, *class_names* - nazivi svih ciljnih klasa (moguće vrijednosti predikcije), *filled* – ako je postavljen na *True*, svi čvorovi grafa bit će obojeni, *fontsize* - veličina teksta.

```
fig = plot.figure(figsize=(25,20))
_ = tree.plot_tree(classifier, feature_names = feature_names,
                   class_names = {0: 'Zaposlen', 1: 'Otišao'},
                   filled = True, fontsize = 16)
```



Slika 29: Generirano stablo odlučivanja (Izvor: Autor)

Korijen stabla predstavlja sljedeće pitanje: Je li zadovoljstvo zaposlenika manje ili jednako 0.494? Ako je odgovor da, postavljamo sljedeće pitanje: Je li broj projekata zaposlenika manji ili jednak 2.5. Ako je odgovor da, postavljamo sljedeće pitanje: Je li prosjek broja radnih sati jednak manji ili jednak 162. Ako je odgovor da, zaposlenik će napustiti kompaniju. Od ukupno 8090 zapisa iz skupa za treniranje, 778 instanci pratilo je ovaj put odlučivanja. Ako je odgovor na potonje pitanje ne, zaposlenik neće napustiti kompaniju. Ovaj put odlučivanja pratilo je 99 instanci.

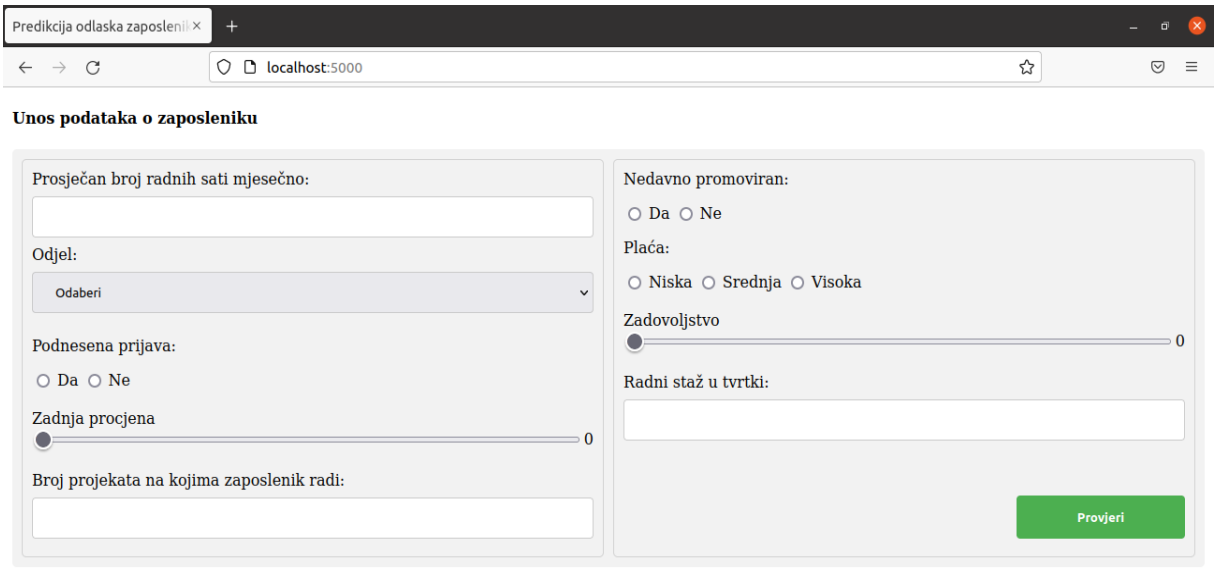
To je bila samo jedna interpretacija stabla. Moguće je stvarati razne predikcije s obzirom na čvorove stabla kojima se krećemo i listove u kojima završimo.

6.6. Primjena modela za predikciju

U ovom poglavlju prikazat ćemo kako se naš model može primijeniti u stvarnom svijetu. Bit će to jednostavna web aplikacija u Flasku koja će koristiti očišćeni skup podataka o zaposlenicima i model za predikciju odlaska novih zaposlenika. Korisnik aplikacije unosit će podatke o nekom novom zaposleniku, a aplikacija će odraditi predikciju i prikazati je na ekranu u obliku informativne poruke.

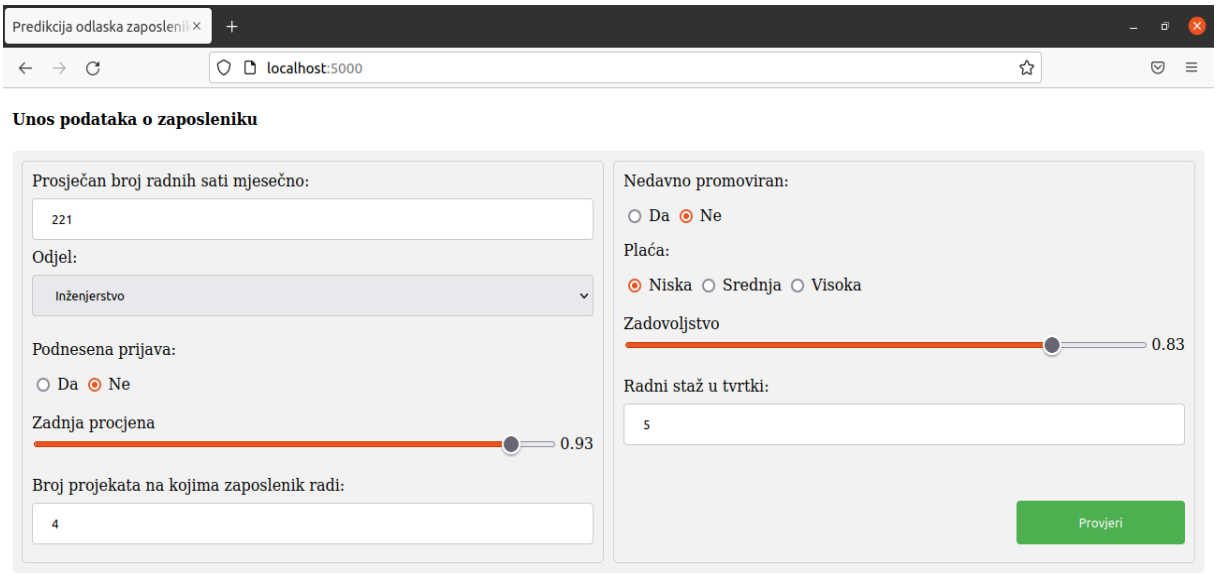
Prema [21], Flask je mala razvojna okolina namijenjena stvaranju web aplikacija u Python programskom jeziku. Sve što je potrebno za razvoj u Flasku je Python okolina. Flask se jednostavno instalira na računalo pomoću *pip* upravitelja paketa.

Za početak, potrebno je stvoriti formu u koju će korisnik unositi podatke. To ćemo napraviti pomoću HTML-a. Prikaz cjelokupnog koda web aplikacije nalazi se u prilozima ovog rada. Ispod se nalazi slika koja prikazuje formu u web pregledniku.



Slika 30: Prazna forma aplikacije u pregledniku (Izvor: Autor)

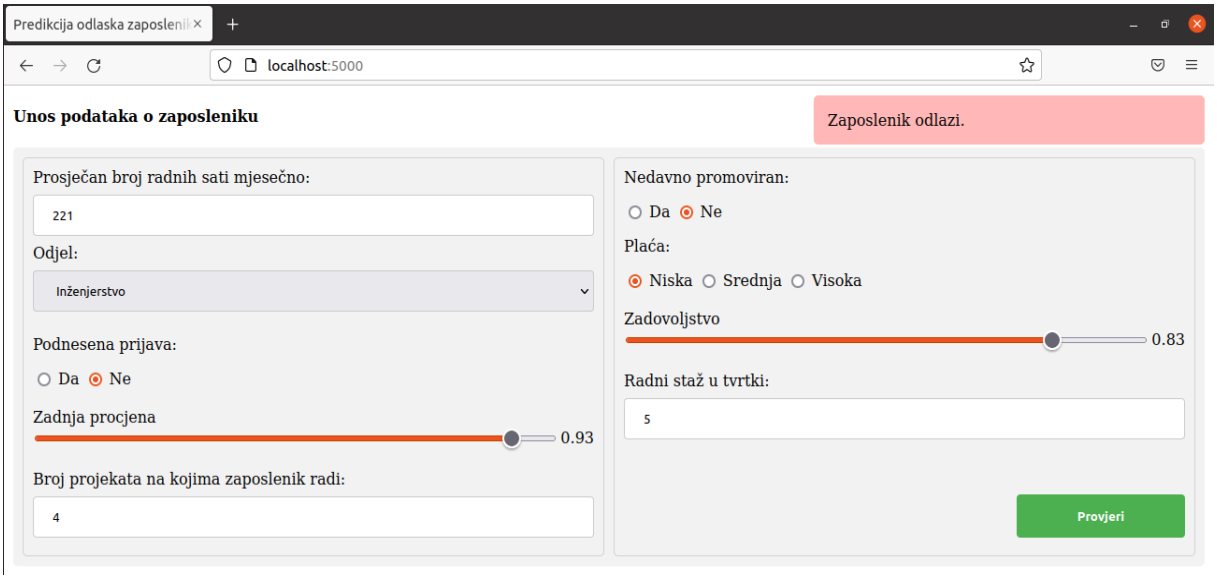
Na formi se nalazi ukupno 9 polja za unos podataka o zaposleniku. Svako polje predstavlja jednu značajku koju smo koristili u izgradnji modela. U donjem desnom uglu forme nalazi se gumb koji šalje podatke iz forme na predikciju. Forma s unesenim podacima prikazana je na slici ispod.



Slika 31: Forma s unesenim podacima (Izvor: Autor)

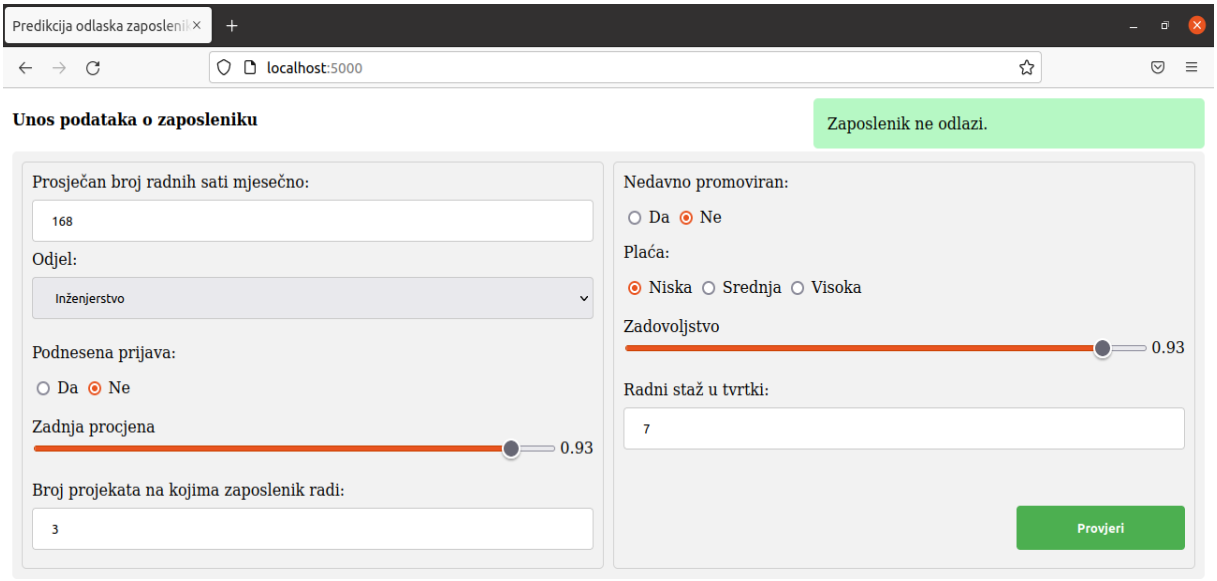
Nakon što korisnik unese podatke i klikne na gumb 'Provjeri', podaci iz forme šalju se pomoću AJAX zahtjeva lokalnom serveru pokrenutom u Flasku. Na serveru se već nalazi istrenirani model za predikciju. To je model kojeg smo izgradili u prošlom poglavlju. Nad modelom se poziva funkcija *predict* kojoj prosljeđujemo parametre iz forme. Ako funkcija vrati 1, korisniku će na ekranu biti ispisana poruka o odlasku zaposlenika. Ako funkcija vrati 0, korisniku će biti ispisana poruka o ostanku.

Na slici ispod prikazan je slučaj u kojem je model predvidio odlazak zaposlenika.



Slika 32: Aplikacija prikazuje poruku o odlasku (Izvor: Autor)

Ako malo izmijenimo podatke u formi i unesemo zaposlenika koji ima duži radni staž, manji prosječni broj sati, manji broj projekata i općenito veće zadovoljstvo, model će napraviti novu predikciju i ispisati drugačiju poruku.



Slika 33: Aplikacija prikazuje poruku o ostanku (Izvor: Autor)

7. Zaključak

Sve tehnike računalnih društvenih znanosti uvelike ovise o računalnim sustavima i njihovim mogućnostima. Metode kao što su analiza društvenih mreža i računalne simulacije raznih društvenih pojava zahtjevaju sustav koji je u mogućnosti obraditi veliku količinu podataka i otkriti skrivene obrasce. Najveći potencijal računalnih društvenih znanosti nude tehnike strojnog učenja. S obzirom da je to područje koje se svakodnevno unapređuje i postaje sve popularnije, razvijat će se i računalne društvene znanosti.

Veliki skupovi podataka skrivaju mnoge prednosti, ali nije moguće koristiti bilo koji algoritam. Potrebno je razumijeti podatke i način rada algoritama. Cilj ovog rada bio je primijeniti moderne metode rada s podacima i algoritme strojnog učenja nad tim podacima. Pokazali smo kako biranim metodama možemo doći do raznih korisnih spoznaja skrivenih u podacima. Prikazan je i rad aplikacije koja koristi model kreiran nad podacima. Aplikaciju mogu koristiti i ljudi koji nisu upoznati sa skupom podataka i podatkovnom znanosti.

Popis literature

[1] „Computational Social Science Specialization“, [Online kurs]. Dostupno: https://www.coursera.org/specializations/computational-social-science-ucdavis#instructors [pristupano 27.02.2022.]

[2] „A brief history of the digital revolution“, [Na internetu]. Dostupno: https://stfc.ukri.org/files/digital-revolution-infographic/ [pristupano: 10.12.2021.]

[3] R. Ramić, „Stroj koji je započeo informatičku eru“, 2009. [Na internetu]. Dostupno: https://www.poslovni.hr/sci-tech/stroj-koji-je-zapoceo-informaticku-eru-106170 [pristupano: 01.02.2022]

[4] Techopedia, „Digital Revolution“, 2017. [Na internetu]. Dostupno: https://www.techopedia.com/definition/23371/digital-revolution [pristupano: 02.02.2022.]

[5] D. Šebalj, A. Živković, „Promjene u upravljanju podacima koje nosi Big Data“, 2016. [Na internetu]. Dostupno: https://www.bib.irb.hr/852382 [pristupano: 01.02.2022.]

[6] IBM (bez dat.) *Big data analytics* [Na internetu]. Dostupno: https://www.ibm.com/analytics/hadoop/big-data-analytics [pristupano: 07.02.2022.]

[7] M. Salganik (29.05.2020.) „An Introduction to Computational Social Science“, *Youtube* [Video datoteka]. Dostupno: https://www.youtube.com/watch?v=zGG9wPl1C5E&ab_channel=SummerInstituteinComputationalSocialScience [pristupano: 10.11.2021.]

[8] Kaggle, *Employee Churn Prediction* [Na internetu]. Dostupno: https://www.kaggle.com/mzinic/employee-churn-prediction [pristupano: 31.01.2022]

[9] V. Saradhi, „Employee Churn Prediction“, 2011. [Na internetu]. Dostupno: https://www.sciencedirect.com/science/article/pii/S0957417410007621 [pristupano: 31.01.2022.]

[10] „Kaggle“, (bez dat.) u *Wikipedia, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Kaggle [pristupano: 31.01.2022.]

[11] „What is a CSV file?“ (bez dat.) [Na internetu]. Dostupno: https://docs.fileformat.com/spreadsheet/csv/ [pristupano: 31.01.2022.]

[12] „The Jupyter Notebook“ (bez dat.) [Na internetu]. Dostupno: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html [pristupano: 03.02.2022.]

[13] A. Navlani, „Predicting Employee Churn in Python“, 2018. [Na internetu]. Dostupno: https://www.datacamp.com/community/tutorials/predicting-employee-churn-python [pristupano: 04.02.2022.]

[14] „Korelacija“, (bez dat.) u *Wikipedia, the Free Encyclopedia*. Dostupno: https://hr.wikipedia.org/wiki/Korelacija [pristupano: 03.02.2022.]

[15] D. Patel (04.02.2019.) „Machine Learning Tutorial Pyrhon – 13: K Means Clustering Algorithm“, *Youtube* [Video datoteka]. Dostupno: https://www.youtube.com/watch?v=EItlUEPCIzM&ab_channel=codebasics [pristupano: 10.02.2022]

[16] Scikit-learn (bez dat.) [Na internetu]. Dostupno: https://scikit-learn.org/ [pristupano: 20.02.2022.]

[17] „Skaliranje značajki“, (bez dat.) u *Wikipedia, the Free Encyclopedia*. Dostupno: https://hr2.wiki/wiki/Feature_scaling [pristupano: 15.02.2022.]

[18] „Matrica dizajna“, (bez dat.) u *Wikipedia, the Free Encyclopedia*. Dostupno: https://hr2.wiki/wiki/Design_matrix [pristupano: 15.02.2022.]

[19] M. Stojiljković, „Split Your Dataset With scikit-learn's train_test_split()“, (bez dat.) [Na internetu]. Dostupno: https://realpython.com/train-test-split-python-data/ [pristupano: 16.02.2022.]

[20] J. Brownlee, „How to Calculate Feature Importance With Python“, 2020. [Na internetu]. Dostupno: https://machinelearningmastery.com/calculate-feature-importance-with-python/ [pristupano: 16.02.2022.]

[21] Envato, „Kreiranje Web Aplikacije iz Temelja Koristeći Python Flask i MySQL“, 2015. [Na internetu]. Dostupno: https://code.tutsplus.com/hr/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql--cms-22972 [pristupano: 17.02.2022.]

[22] Codecademy (bez dat.) *Clustering: K-Means* [Na internetu]. Dostupno: https://www.codecademy.com/learn/machine-learning/modules/dspath-clustering/cheatsheet [pristupano: 18.02.2022.]

[23] Google Developers (bez dat.) *K-Means Advantages and Disadvantages* [Na internetu]. Dostupno: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages [pristupano: 19.02.2022.]

[24] GeeksForGeeks, „Decision Tree“, 2021. [Na internetu]. Dostupno: https://www.geeksforgeeks.org/decision-tree/ [pristupano: 20.02.2022.]

[25] C. Bento, „Decision Tree Classifier explained in real-life: picking a vacation destination“, 2021. [Na internetu]. Dostupno: https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575 [pristupano: 23.02.2022.]

[26] J. Hoare, „Machine Learning: Pruning Decision Trees“. [Na internetu]. Dostupno: https://www.displayr.com/machine-learning-pruning-decision-trees/ [pristupano: 21.02.2022.]

[27] G. Seif, „The 5 Clustering Algorithms Data Scientists Need to Know“, 2018. [Na internetu]. Dostupno: https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 [pristupano: 24.02.2022.]

[28] D. Oreški, „Algoritmi strojnog učenja“, nastavni materijali na predmetu Inteligentni sustavi [Moodle], Sveučilište u Zagrebu, Fakultet organizacije i informatike, Varaždin, 2020.

[29] „Machine Learning“, (bez dat.) u *Wikipedia, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Machine_learning [pristupano: 25.02.2022.]

[30] Javatpoint (bez dat.) *Regression vs. Classification in Machine Learning* [Na internetu]. Dostupno: https://www.javatpoint.com/regression-vs-classification-in-machine-learning [pristupano: 26.02.2022.]

[31] INTRAC, „Social Network Analysis“, 2017. [Na internetu]. Dostupno: https://www.intrac.org/wpcms/wp-content/uploads/2019/05/Social-network-analysis.pdf [pristupano: 01.03.2022.]

Popis slika

Popis tablica

Prilozi

Prilog 1. data-cleansing.ipynb

- Juypter Notebook bilježnica u kojoj se obavlja priprema podataka

```
import pandas
dataframe = pandas.read_csv('employee_data.csv')
dataframe.shape

dataframe.drop_duplicates(inplace=True)
dataframe.isnull().sum()

dataframe.department.unique()
dataframe = dataframe[dataframe['department'].notna()]
dataframe['department'].replace({'information_technology': 'IT'},inplace=True)
dataframe['department'].replace({'engineering': 0, 'support': 1, 'sales': 2, 'IT':
3, 'product': 4, 'marketing': 5, 'temp': 6, 'procurement': 7, 'finance': 8,
'management': 9, 'admin': 10,},inplace=True)

dataframe.filed_complaint.fillna(0, inplace=True)
dataframe.recently_promoted.fillna(0, inplace=True)

dataframe['salary'].replace({'low': 0, 'medium': 1, 'high': 2}, inplace=True)

dataframe["status"] = pandas.get_dummies(dataframe.status).Left

dataframe.isnull().sum()
dataframe.dropna(inplace=True)
dataframe.isnull().sum()
dataframe.shape
dataframe.info()

dataframe.dropna(inplace=True)
dataframe.to_csv('employee_churn_clean.csv', index=None)
```

Prilog 2. decision-tree-classifier.ipynb

- Juypter Notebook bilježnica u kojoj stvaramo model za predikciju

```
import pandas
dataframe = pandas.read_csv('employee_churn_clean.csv')

from sklearn import tree
from sklearn.model_selection import train_test_split

design_matrix = dataframe.drop('status', axis=1) # sve značajke koje će nam služiti
za predikciju
targets = dataframe.status # varijabla za predikciju


x_train, x_test, y_train, y_test = train_test_split(design_matrix, targets,
test_size=0.33) # podjela u skupove za treniranje i testiranje

classifier = tree.DecisionTreeClassifier().fit(x_train, y_train)

path = classifier.cost_complexity_pruning_path(x_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
ccp_alphas

classifiers = []
for ccp_alpha in ccp_alphas:
    clf = tree.DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(x_train, y_train)
    classifiers.append(clf)
classifiers

from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plot
train_acc = []
test_acc = []
for c in classifiers:
    y_train_prediction = c.predict(x_train)
    y_test_prediction = c.predict(x_test)
    train_acc.append(accuracy_score(y_train_prediction, y_train))
    test_acc.append(accuracy_score(y_test_prediction, y_test))

# graf koji prikazuje preciznost s obzirom na ccp_alpha
plot.scatter(ccp_alphas, train_acc)
plot.scatter(ccp_alphas, test_acc)
plot.plot(ccp_alphas, train_acc, label='train_accuracy', drawstyle='steps-post')
plot.plot(ccp_alphas, test_acc, label='test_accuracy', drawstyle='steps-post')
plot.legend()
plot.title('Preciznost i ccp_alpha')
plot.show()

classifier = tree.DecisionTreeClassifier(ccp_alpha=0.01) # novi model s parametrom
ccp_alpha
classifier.fit(x_train, y_train) # treniranje modela

predictions = classifier.predict(x_test)  #  radimo predikciju nad testnim
podacima, modelu ne dajemo odgovore
predictions

# provjera jel model radi dobru predikciju
accuracy_score(y_test, predictions)

# bitnije značajke
feature_names = design_matrix.columns
feature_importance = pandas.DataFrame(classifier.feature_importances_, index =
feature_names).sort_values(0)
feature_importance.plot(kind='bar')
```

```
fig = plot.figure(figsize=(25,20))
_ = tree.plot_tree(classifier, feature_names = feature_names, class_names = {0:
'Zaposlen', 1: "Otišao"}, filled = True, fontsize = 12)
```

Prilog 3. index.html

- sadržaj web aplikacije

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="../static/css/style.css">
        <title>Predikcija odlaska zaposlenika</title>
    </head>
    <body>
        <h4>Unos podataka o zaposleniku</h4>

        <div class="message background-green">
            <p class="prediction"></p>
        </div>

        <div class="form-container">
            <form id="form" action="" method="post">
                <div class="d-flex g-10">
                    <div class="f-basis-100 grey-border p-10">
                        <label for="avgMonthlyHrs">Prosječan broj radnih sati
mjesečno:</label>
                        <input type="text" id="avgMonthlyHrs" name="avgMonthlyHrs"
value="">

                        <label for="department">Odjel:</label>
                        <select id="department" name="department">
                            <option value="" selected disabled
hidden>Odaberi</option>
                            <option value="0">Inženjerstvo</option>
                            <option value="1">Podrška</option>
                            <option value="2">Prodaja</option>
                            <option value="3">IT</option>
                            <option value="4">Proizvodnja</option>
                            <option value="5">Marketing</option>
                            <option value="6">Privremeni</option>
                            <option value="7">Nabava</option>
                            <option value="8">Financije</option>
                            <option value="9">Menadžment</option>
                            <option value="10">Admin</option>
                        </select><br>

                        <p>Podnesena prijava:</p>
                        <input type="radio" id="complaintFiled" name="complaint"
value="1">
                        <label for="complaintFiled">Da</label>
                        <input type="radio" id="complaintNotFiled" name="complaint"
value="0">
                        <label for="complaintNotFiled">Ne</label>

                        <div class="range">
                            <label for="lastEvaluation">Zadnja procjena</label>
                            <div class="range-data">
                                <input type="range" value="0" min="0" step="0.01"
max="1" id="lastEvaluation" name="lastEvaluation"
oninput="this.nextElementSibling.value = this.value">
                                <output>0</output>

                            </div>
                        </div>
                        <label for="numOfProjects">Broj projekata na kojima
zaposlenik radi:</label>
                        <input type="text" id="numOfProjects" name="numOfProjects"
value="">
```

```
                    </div>
                    <div class="d-flex flex-column justify-content-between f-basis-
100 grey-border p-10">
                        <div>
                            <p class="mt-0">Nedavno promoviran:</p>
                            <input type="radio" id="recentlyPromoted"
name="promotion" value="1">
                            <label for="recentlyPromoted">Da</label>
                            <input type="radio" id="notRecentlyPromoted"
name="promotion" value="0">
                            <label for="notRecentlyPromoted">Ne</label>

                            <p>Plaća:</p>
                            <input type="radio" id="salaryLow" name="salary"
value="0">
                            <label for="salaryLow">Niska</label>
                            <input type="radio" id="salaryMedium" name="salary"
value="1">
                            <label for="salaryMedium">Srednja</label>
                            <input type="radio" id="salaryHigh" name="salary"
value="2">
                            <label for="salaryHigh">Visoka</label>

                            <div class="range">
                                <label for="satisfaction">Zadovoljstvo</label>
                                <div class="range-data">
                                    <input type="range" value="0" min="0"
step="0.01" max="1" id="satisfaction" name="satisfaction"
oninput="this.nextElementSibling.value = this.value">
                                    <output>0</output>
                                </div>
                            </div>

                            <label for="tenure">Radni staž u tvrtki:</label>
                            <input type="text" id="tenure" name="tenure" value="">
                        </div>
                        <input type="submit" value="Provjeri">
                    </div>
                </div>
            </form>
        </div>

        <script src="https://code.jquery.com/jquery-3.5.1.js"
          integrity="sha256-QWo7LDvxbWT2tbbQ97B53yJnYU3WhH/C8ycbRAkjPDc="
          crossorigin="anonymous"></script>

          <script type="text/javascript">
            $(document).on('submit', '#form', function(e) {
              e.preventDefault();

              $.ajax({
                type: 'POST',
                url: '/',
                data: {
                  avgMonthlyHrs: $('#avgMonthlyHrs').val(),
                  department: $('#department').val(),
                  complaint: $('input[name="complaint"]:checked').val(),
                  lastEvaluation: $('#lastEvaluation').val(),
                  numOfProjects: $('#numOfProjects').val(),
                  promotion: $('input[name="promotion"]:checked').val(),
                  salary: $('input[name="salary"]:checked').val(),
                  satisfaction: $('#satisfaction').val(),
                  tenure: $('#tenure').val()
                },
                success: function(data)
                {
                    $(".message").show()
```

```
                    if(data.prediction === '1') {
                        $(".message").removeClass('background-green')
                        $(".message").addClass('background-red')
                        $(".prediction").html("Zaposlenik odlazi.");
                    }
                    else {
                        $(".message").removeClass('background-red')
                        $(".message").addClass('background-green')
                        $(".prediction").html("Zaposlenik ne odlazi.");
                    }
                }
            })
        });
    </script>
</body>
</html>
```

Prilog 4. app.py

- lokalni server web aplikacije

```
import sys
import argparse
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split
from flask import Flask, render_template, request, jsonify

app = Flask(__name__)

@app.route('/', methods=['post', 'get'])

def send():
    if request.method == 'POST':
        avgMonthlyHrs = request.form['avgMonthlyHrs']
        department = request.form['department']
        complaint = request.form['complaint']
        lastEvaluation = request.form['lastEvaluation']
        numOfProjects = request.form['numOfProjects']
        promotion = request.form['promotion']
        salary = request.form['salary']
        satisfaction = request.form['satisfaction']
        tenure = request.form['tenure']

        prediction = classifier.predict([[avgMonthlyHrs, department, complaint,
                                          lastEvaluation, numOfProjects, promotion,
                                          salary, satisfaction, tenure]])[0]

        return jsonify({'prediction': str(prediction)})

    return render_template('index.html')


if __name__ == '__main__':
    parser = argparse.ArgumentParser(formatter_class=argparse.RawTextHelpFormatter)
    parser.add_argument("-csv", type=str, required=True)
    args = parser.parse_args()

    dataset = args.csv

    try:
        dataframe = pd.read_csv(dataset)
    except FileNotFoundError:
        print("Datoteka " + dataset + " ne postoji!")
        sys.exit()

    design_matrix = dataframe.drop('status', axis=1)
    targets = dataframe.status

    x_train, x_test, y_train, y_test = train_test_split(design_matrix, targets,
test_size=0.33)

    classifier = tree.DecisionTreeClassifier(ccp_alpha=0.01)
    classifier.fit(x_train, y_train)

    app.run()
```

Prilog 5. style.css

- datoteka za cjelokupno stiliziranje web aplikacije

```
body {
    margin: 10px;
}

.form-container {
    border-radius: 5px;
    padding: 10px;
    background-color: #f2f2f2;
}

.range {
    margin: 20px 0;
}
.range label {
    margin-top: 10px;
}
.range-data {
    display: flex;
    align-items: center;
}
.range input {
    width: 100%;
    margin-right: 5px;
}

input[type=text], select {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type=submit] {
    width: 30%;
    background-color: #4CAF50;
    color: white;
    font-weight: bold;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-left: auto;
}

.message {
    position: absolute;
    top: 0;
    right: 0;
    height: 50px;
    margin: 10px;
    width: 30%;
    border-radius: 5px;
    padding: 0 15px;
    display: none;
}

input[type=submit]:hover {
    background-color: #45a049;
}
```

```
.d-flex {
    display: flex;
}

.flex-column {
    flex-direction: column;
}

.justify-content-between {
    justify-content: space-between;
}

.f-basis-100 {
    flex-basis: 100%;
}

.grey-border {
    border: 1px solid lightgray;
    border-radius: 5px;
}

.g-10 {
    gap: 10px;
}

.p-10 {
    padding: 10px;
}

.mt-0 {
    margin-top: 0;
}

.background-green {
    background-color: rgb(182, 248, 196);
}

.background-red {
    background-color: rgb(255, 183, 183);
}
```