Razvoj inteligentnog sustava temeljenog na stablu odlučivanja

Lukman, Leon

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leon Lukman

RAZVOJ INTELIGENTNOG SUSTAVA TEMELJENOG NA STABLU ODLUČIVANJA

DIPLOMSKI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leon Lukman

Studij: Organizacija poslovnih sustava

RAZVOJ INTELIGENTNOG SUSTAVA TEMELJENOG NA STABLU ODLUČIVANJA

DIPLOMSKI RAD

Mentor/Mentorica:

Doc. dr. sc. Dijana Oreški

Varaždin, lipanj 2020.

Leon Lukman

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

U teoretskom dijelu rada opisat će se koraci razvoja inteligentnog sustava te napraviti pregled dosadašnjih istraživanja na ovu temu. U praktičnom djelu rada izradit će se stablo odlučivanja na temelju javno dostupnog skupa podataka u odabranoj domeni. Stablo odlučivanja služi kao temelj za izradu inteligentnog sustava.

Ključne riječi: stablo odlučivanja, inteligentni sustavi, strojno učenje, vrste strojnog učenja, podatak, informacija, rudarenje podataka, mjere nereda, weka.

Sadržaj

1. Uvod

Govorit ćemo i obradit sve teorijske aspekte, koji su vezani uz temu ovog diplomskog rada. Za početak ćemo reći nešto o inteligentnim sustavima i što zapravo mislimo pod time. Nakon toga ćemo reći nešto o podatkovnoj znanosti i podatkovnim proizvodima, te o razlici između podataka i informacija, koji se često pogrešno koriste kao sinonimi. Također, pozabaviti ćemo se i s rudarenjem podataka i otkrivanjem znanja u bazama podataka, te ih objasniti i opisati njihov proces. Prije nego što krenemo na strojno učenje, njegove primjene, vrste i razne metode, prvo ćemo definirati što je zapravo učenje, općenito, kako bismo bolje razumjeli razliku između ljudskog i strojnog učenja. Na kraju teorijskog dijela, prije nego što krenemo sa praktičnim, pozabavit ćemo se sa stablima odlučivanja. Reći ćemo nešto o tome, što su zapravo stabla odlučivanja i koji su njihovi dijelovi, kako se provodi grananje i obrezivanje stabla, te nešto o tri najpoznatija algoritma koja se koriste za dobivanje modela stabla odlučivanja i kako evaluirati isti.

Zadnji dio rada je vezan uz praktični dio, u kojem ćemo na primjeru prikazati i dodatno pojasnit teoriju iz prvog dijela. Prvo ćemo reći nešto o samom programu koji ćemo koristiti za strojno učenje, a nakon toga ćemo početi provodit praktični dio, kroz proces otkrivanja znanja u bazama podataka. Upoznat ćemo se sa skupom podataka koje smo odabrali i sa samom domenom, te ćemo pripremiti podatke kako bismo nad njima mogli vršiti obradu. Na kraju ćemo, nakon što provedemo algoritam, prikazati i opisati dobivene rezultate i time završiti praktični dio rada.

2. Inteligentni sustavi

Od samog početka čovječanstva, ljudi su razvijali i koristili razne alate, s namjerom, da im olakšaju život. Unatrag samo nekoliko desetljeća, razvijeno je i računalo, stroj koji je bio u stanju obaviti veliku količinu računalnih operacija, puno brže i točnije nego bilo koji čovjek.

Kako je vrijeme prolazilo i tehnologija se sve više razvijala, rodila se ideja da se napravi računalo koje bi bilo u stanju razmišljati i učiti kao čovjek. Upravo to i bolje razumijevanje inteligencije, cilj je istraživanja koja se provode na području umjetne inteligencije, za koju bi najveći uspjeh bio, da se kreira takav stroj ili sustav. Jasno je da je to nije nimalo lagan zadatak, no iako dosta skromni, već su postignuti prvi uspjesi usmjereni ostvarivanju upravo tog cilja. [1]

„Inteligentnim sustavima se smatraju oni sustavi koji imaju sposobnost prikupljanja i uporabe znanja, kao i sposobnost interakcije, komuniciranja s vanjskim svijetom odnosno korisnicima ili pak s drugim inteligentnim sustavom. To su sustavi koji imaju sposobnost zaključivanja, sposobnost obrade i zamjene znanja, kao i sposobnost objašnjenja svog ponašanja. “ [2]

Umjetna inteligencija zajedno sa drugim granama računalne znanosti, uspjeli su razviti nekoliko računalnih alata; sustave temeljene na znanju, računalnu inteligenciju i hibridne sustave koji zajedno čine inteligentne sustave. Iako ti sustavi nisu bili u stanju riješiti problem kreiranja „prave“ umjetne inteligencije, pomogli su riješiti neke probleme koji su se prije smatrali previše složenima, te pronaći puno efektivniji način za rješavanje drugih. [1]

2.1. Podatkovna znanost

Mnoge znanstvene discipline su izmišljene puno ranije nego što su ih ljudi bili u stanju efektivno i efikasno primijeniti. Glavni razlog zbog kojeg je to bilo tako, je da nisu imali pristup tehnologiji koja im je bila potrebna ili jednostavno u to vrijeme još nije bila dovoljno razvijena. Zahvaljujući napretku tehnologije, neke od tih znanstvenih disciplina su konačno zaživjele, te su počele postajati sve više popularne.

Neke znanosti su se razvile iz jedne ili više starijih, pa se zna desiti da postoje nejasnoće, po čemu se one razlikuju. Podatkovna znanost je jedan takav slučaj, a pošto se radi o relativno novom području, mnogi imaju različita mišljena o tome što ono zapravo predstavlja. Neki vele, da se radi o novoj generaciji statistike, jer se isto koristi za obradu i pronalaženje pravilnosti u podacima. Drugi pa vele, da je spoj više interdisciplinarnih polja ili da se radi o skroz novom tijelu znanja (potpunom skupu koncepata, termina i aktivnosti koje tvore profesionalnu domenu). [3]

No, iako su dosta slične, podatkovna znanost je holistički pristup koji podupire cijeli proces koji uključuje prikupljanje, pohranu, obradu, rudarenje podataka i otkrivanje znanja. „Podatkovna znanost je novo interdisciplinarno polje koje sintetizira i nadograđuje statistiku, informatiku, računarstvo, komunikaciju, upravljanje i sociologiju kako bi proučavalo podatke i njihovo okruženje s namjerom njihove transformacije i dobivanja uvida i odluka“. [3], [4]

„Podatkovna znanost (*eng. Data Science*) je interdisciplinarno područje čiji je glavni zadatak ekstrakcija informacija i znanja iz podataka, bilo da oni dolaze u strukturiranom ili nestrukturiranom obliku. Radi se o području koje obuhvaća velike podatke (*eng*. *Big Data*), strojno učenje(eng. *Machine Learning*), vizualizaciju podataka te sve što je vezano uz izvlačenje znanja iz podataka (statistika, računalne znanosti...). “[5]

2.2. Podatkovni proizvodi

Kao što i sam naziv govori, radi se o digitalnim podatkovnim proizvodima koji su vezani uz neku domenu i koji se mogu kupiti. Ovakvi proizvodi se dobivaju kao rezultat primjene podatkovne znanosti.

„Podatkovni proizvod se može dobiti iz podataka ili je omogućen ili ga pokreću podaci, a može biti otkriće, predviđanje, usluga, preporuka, uvid u donošenje odluka, razmišljanje, model, način rada, paradigma, alat ili sustav. Krajnji proizvodi podataka od vrijednosti su znanje, inteligencija, mudrost i odluka“. [3]

Kako će se podatkovne znanost, inženjerska teorija i tehnologija dalje razvijat, tako će se također stvarat novi podatkovni proizvodi. To sve će se vjerojatno odvijati brzinom i mjerom koju ne možemo ni zamisliti, baš kao što su to pokazali evolucija internetskih proizvoda i sustavi umjetne inteligencije. [3]

2.3. Podaci i informacije

Pojmovi podatak i informacija, koriste se svakodnevno u raznim situacijama, pogotovo kada se radi o nečemu što je vezano uz računala. Ta dva pojma su usko povezana, pa ih ljudi često pogrešno koriste kao sinonime. Obzirom na njihovo često korištenje, bilo bi korisno da ih razumijemo i znamo razlikovati.

Kada promatramo neku pojavu u stvarnom svijetu, često bilježimo i neka njezina obilježja i karakteristike. Zapis koji smo stvorili na temelju zapažanja i koji se koristi za predstavljanje obilježja promatrane pojave, je činjenica, a može biti kvantitativnog ili kvalitativnog oblika. Upravo to, je ono, što nam podatak nudi, odnosno, što on je. Podatak je samo činjenica bez interpretaciju ili konteksta u kojem se javlja. [6]

„Podatak je nositelj informacije i služi za njeno tehničko uobličavanje kako bi se mogla sačuvati ili prenijeti, a informacija je protumačen podatak o pojavi koju podatak prikazuje.“ [6]

Dakle, nakon što se skup podataka obradi, rezultat obrade koji smo dobili je informacija, koja je širi pojam, te je razumljiva, ima neku vrijednost i može se iskoristiti. Često osim pojma podaci, možemo čuti i sirovi podaci. Pod time se samo misli na izvorne, primarne podatke, koji se obrađuju i koji su prikupljeni s određenim ciljem. Osim primarnih, također imamo i sekundarne podatke, odnosno, već postojeći i prethodno prikupljeni podaci, koji se onda obrađuju. Iako je sakupljanje podataka postalo lakše i jeftinije zahvaljujući tehnološkom napretku, svi ti podaci ne vrijede apsolutno ništa, ukoliko se ne analiziraju, obrade i iskoriste na neki način. [6]

Sama vrijednost i kvaliteta podataka i informacija koji se koriste, ovisi o njihovoj točnosti, razumljivosti, te o tome da li ih možemo dobiti i pristupiti im u pravo vrijeme, te kad su nam potrebni. Iako se za donošenje odluka teži korištenju podataka koji su što svježiji, to ne znači da su stariji podaci manje vrijedni. Baš suprotno, takvi podaci su jako korisni i u nekim slučajevima čak vrjedniji od novih podatka. Jedan od razloga je taj, da se stari podaci mogu koristiti, kako bismo ih usporedili sa novim podacima, te ustanovili promjene koje su se dogodile. Pogotovo su važni, kod praćenja vremenskih promjena ili predviđanja demografskih kretanja stanovništva.

2.4. Atributi

Svi podaci koje smo skupili o nekoj domeni, zovemo skup podataka. On je najčešće u obliku tablice, gdje svaki stupac predstavlja neki atribut, varijablu ili obilježje pojave, čije vrijednosti bilježimo, a svaki red jednu instancu, odnosno, jednu pojavu. Atributi mogu biti numerički ili pak mogu biti kategorijski. Kada su atributi kategorijski, imaju određen broj vrijednosti koje mogu poprimiti, a te vrijednosti se još zovu kategorija / klasa / oznaka i mogu biti tekst ili numeričke prirode, pa čak i nestrukturirani kao što su to slike. [4]

Postoje dvije glavne klase, kada se radi o kategorijskim podacima, a to su nominalni i ordinalni. Kod nominalnih podatkovnih atributa, ne postoji način ili koncept po kojim bi ih mogli poredati. Primjer toga bi bile kategorije vremena tipa sunčano, vjetrovito, kišovito i tako dalje. Iako imamo nekoliko klasa ili kategorija, nemamo neki specifičan redoslijed po kojem bi ih mogli poredati. Kod ordnalnih kategorijskih atributa je situacija već malo drugačija, jer kod njih imamo neki koncept, po kojem ih možemo poredati. Primjer toga bi, recimo, bile veličine odjeće koje možemo poredati od manjih prema većima. [7]

2.5. Rudarenje podataka

Primjena metoda strojnog učenja nad velikim bazama podataka se naziva rudarenje podataka (*eng. Data Mining*). Taj proces se lako može usporediti sa pravim rudarenjem rudnika iz kojeg se vade velike količine rude, te se obradom iste, dobiva vrjedniji materijal. Slično tako je i sa rudarenjem podataka, gdje je cilj od velikog skupa podataka u kojem se nalaze brojne instance, gdje svaka služi kao primjer, kreirati vrijedan i koristan model, koji se onda može iskoristiti i primijeniti nad budućim, dosad neviđenim primjerima. [8]

„Rudarenje podacima je istraživanje i analiza velikih količina podataka pomoću automatskih ili poluautomatskih metoda s ciljem otkrivanja smislenih pravilnosti“. [9]

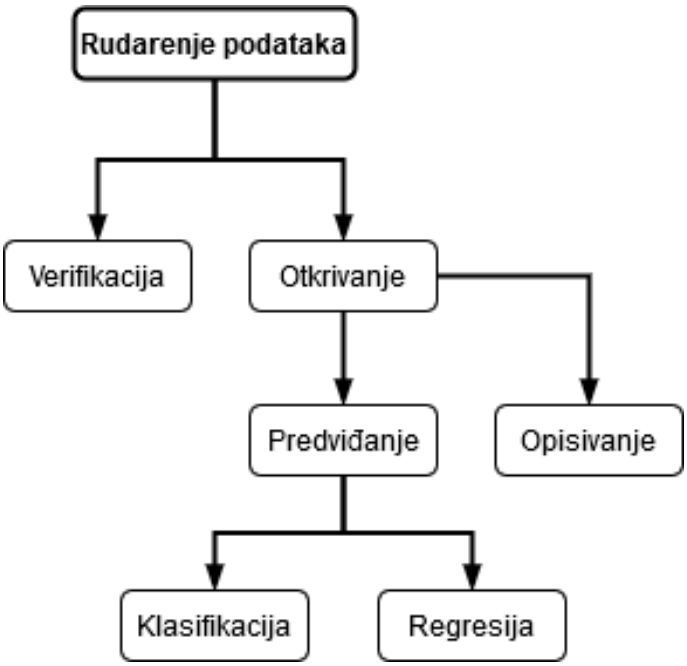
Prilikom rudarenja podataka je korisno rasporediti domenu rudarenja na četiri sloja. Prvi sloj je vezan uz samu primjenu, odnosno za koju svrhu je namijenjen krajnji rezultat. Prema svrsi se onda, u drugom sloju, određuje zadatak koji će se obaviti rudarenjem. Redoslijed trećeg i četvrtog sloja, je malo izmiješan. Obično se prvo primjenjuje algoritam, kojim se onda dobiva, model. Možda je bolje prvo odrediti kakva vrsta modela će biti bolja za korištenje i implementaciju, te na temelju toga odabrati algoritam. Razlog tome je da su neki modeli teže interpretiraju, a neki lakše. [4]

2.5.1. Taksonomija metoda rudarenja podataka

Rudarenje podataka ima puno raznih metoda i modela koji se spominju, a kako bi se što lakše snašli u svemu tome, bilo bi dobro da se veli nešto i o njegovoj taksonomiji.

„Taksonomija je proces imenovanja i razvrstavanja stvari kao što su životinje i biljke u grupe unutar većeg sustava, prema njihovim sličnostima i razlikama.“ [10]

Bitno je razlikovati dvije glavne vrste, odnosno dva cilja rudarenja, koji svaki imaju svoju metodologiju. Prvi je rudarenje usmjereno na verifikaciju, odnosno, na to da se potvrdi neka hipoteza. Drugi pak je, rudarenje orijentirano na otkrivanje, gdje sustav samostalno pronalazi nova pravila i uzorke u podacima. [4]



Slika 1 Taksonomija metoda rudarenja podataka [4]

Pod ovaj tip rudarenja spadaju i metode predviđanja, te opisivanja. Metode opisivanja (*eng. description*) se uglavnom fokusiraju na razumijevanje načina na koji podaci funkcioniraju i na pronalaženje pravilnosti u podacima koje mogu interpretirati ljudi. Metode usmjerene na predviđanje, koriste varijable iz baze podataka, pomoću kojih kreiraju model za predviđanje novih vrijednosti drugih varijabli, za koje smatra da su od interesa. [11]

Glavna razlika između tih dvaju modela je ta, da „ većina tehnika koje su usmjerene na otkrivanje koriste induktivno učenje, a metode verifikacije su više usmjerene na to ta provjere hipotezu nekog eksperta.“ [11]

2.6. Otkrivanje znanja u bazama podataka

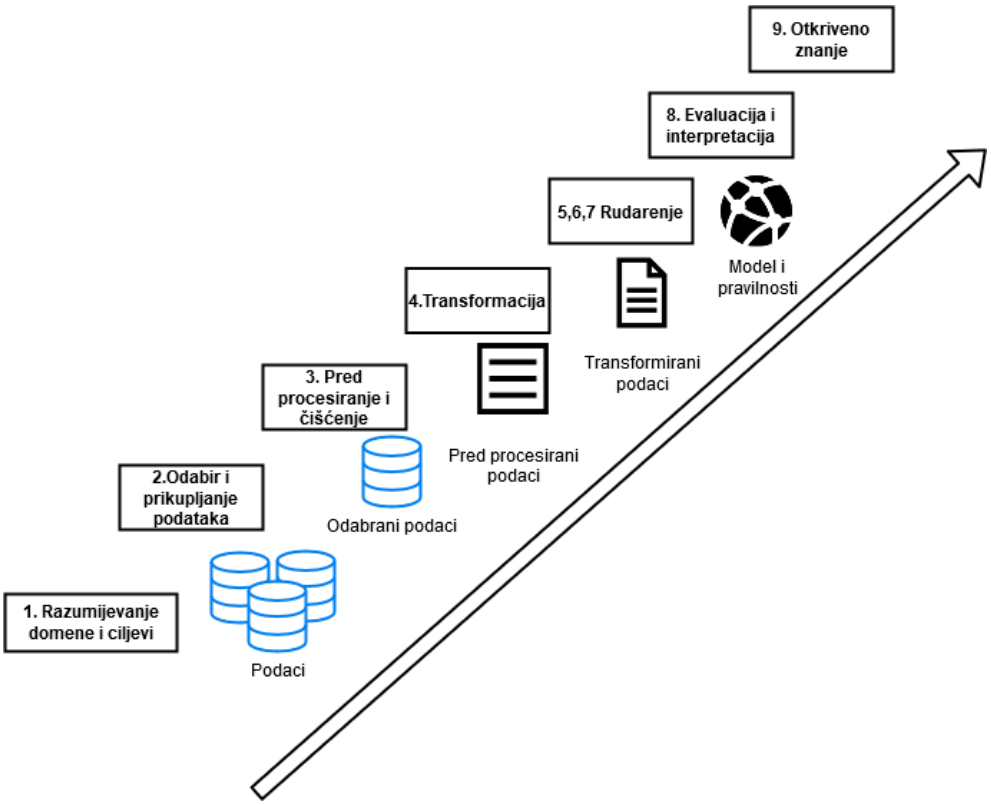
Ovaj pojam KDD (*eng. Knowledge Discovery in Databases*) koji je definiran 1996. od Fayyad-a kao „netrivijalan proces identifikacije validnih, novih, potencijalno korisnih i razumljivih pravilnosti u podacima.“ se koristi naizmjenično sa rudarenjem podataka. A i po njegovoj definiciji koja je jako slična kao i definicija rudarenja podataka, može se vidjeti da je to tako s razlogom. Iako se prije znalo govoriti da je rudarenje podataka samo jedan korak u procesu otkrivanja znanja podataka, to u novije vrijeme više nije tako, pošto su se ta dva pojma ili bolje rečeno, ono što predstavljaju, izjednačila. [4]

Krajnji cilj KDD-a i rudarenja je, naime, otkrivanje, te izvlačenje novog i korisnog znanja iz podatka. To se čini primjenom metoda rudarenja podataka (algoritama) za izvlačenje (identifikaciju) onoga što se smatra znanjem, u skladu s specifikacijama mjera i pragova, koristeći bazu podataka zajedno sa pred procesiranjem podataka, uzimanjem uzoraka i transformacijama same baze. [11]

2.7. Proces otkrivanja znanja i pravilnosti u podacima

Cijeli proces se provodi kroz devet koraka koji su svi iterativni. To znači da je moguće, a ponekad i potrebno, vratiti se, na prethodni korak i dodatno ga prilagoditi. Kako bi se donijele ispravne odluke, važno je razumjeti svaki korak, ali i sam proces otkrivanja znanja kao cjelinu.

Sve započinje tako, da se odabere neki određeni cilj koji se želi ostvariti, a završava tako, da se novootkriveno znanje implementira, jer novostečena saznanja nemaju vrijednost, ukoliko ih ne iskoristimo. [4]



Slika 2 Proces otkrivanja znanja u podacima [4]

U prvom koraku je važno, razviti razumijevanje problemske domene, obzirom na njezinu primjenu. Svi koji sudjeluju u procesu, moraju biti dobro upoznati s domenom, kako bi što bolje mogli razumjeti želje i potrebe krajnjih korisnika, odnosno, njihov cilj.

Nakon što se definiraju ciljevi i kako bi uopće mogli početi s obradom, potrebno je imati dovoljnu količinu podataka. Drugi i jedan od moguće najvažnijih koraka, je njihovo prikupljanje. Potrebno je odrediti podatke koji će se koristiti, vidjeti koji su dostupni, te pronaći dodatne podatke za koje se smatra da nedostaju i sve skupa ih spojiti u jednu cjelinu. Ovaj korak traje najduže, te je ključan, jer o njemu ovisi ostatak procesa i kvaliteta samog rezultata koji će se proizvesti. [4]

Treći korak se odnosi na pred procesiranje i čišćenje podatka, a se provodi, kada su prikupljeni svi potrebni podaci. Ovaj korak uključuje stvari, kao što je upravljanje nedostajućim vrijednostima, te moguće brisanje ili predviđanje vrijednosti nekog atributa, ukoliko on sadrži previše nedostajućih vrijednosti. Prikupljeni podaci su rijetko kad dovoljno dobri, za direktno korištenje. Cilj je da se ukloni „šum“ iz sirovih podataka i da se pretvore u oblik koji je pogodan za korištenje, kako bi greške obrade bile minimalne. Kada se veli šum, misli se na greške u podacima koji su uzrokovane lošim mjerenjem, bilo da se radi o ljudskoj greški ili da je uzrok korištenje loše opreme. [4], [12]

U četvrtom koraku se vrši transformacija podataka, točnije generiranje boljih i prikladnijih podataka, za rudarenje. Pronalaze se korisni atributi, te se koriste metode za redukciju dimenzija ili transformaciju atributa. Glavni cilj ovog koraka je da se smanji broj varijabli, kako bi proces bio manje zahtjevan. [4]

U petom koraku, uglavnom se određuje zadatak za rudarenje, koji najbolje odgovara našim potrebama. Najčešći zadaci, koji će kasnije biti detaljnije objašnjeni, su klasifikacija, regresija ili grupiranje. [4]

U šestom koraku se određuje algoritam, odnosno metoda koja će se koristiti za pronalaženje uzoraka i pravilnosti u podacima. Osim metode traženje zakonitosti u podacima, također je potrebno podesiti odgovarajuće parametre. Kako bi rudarenje podataka bilo što uspješnije, potrebno je poznavati uvjete pod kojima će pojedini algoritam dati nabolje rezultate. [4]

Sedmi korak slijedi nakon što se odabere algoritam, jer ga je sad potrebno primijeniti nad skupom podataka koje smo prikupili. Jednostavnije rečeno, provodi se rudarenje i traže se pravilnosti. Algoritam na temelju pravilnosti u podacima, kreira rezultat u obliku modela ili pravila. Provodi se toliko puta, koliko je potrebno da se dobije zadovoljavajući rezultat, a to opet ovisiti o podešavanju parametara. [4]

U osmom koraku se novootkrivena znanja interpretiraju i dokumentiraju, te se procjenjuju s obzirom na ciljeve koji su bili postavljeni u prvom koraku procesa. Uglavnom se gledaju razumljivost i korisnost dobivenog modela. [4]

U zadnjem, devetom koraku, koriste se novootkrivena znanja dobivena iz prethodno opisanog procesa. To se radi tako, da se nova znanja implementiraju ili iskoriste u nekom sustavu, te se mjere i bilježe promjene nad tim sustavom. Sama efektivnost cijelog prethodnog procesa, ovisi o uspješnosti ovog zadnjeg koraka. [4]

3. Učenje

Većina organizama za koje bi se moglo reći da su malo „jednostavniji“ imaju kraći životni vijek. Oni žive u relativno stabilnom okruženje, te se ponašaju u skladu sa svojim instinktima i nagonima koje su razvili kroz tisuće godina evolucije. Iako mnogo vrsta ima sposobnost učenja i prilagodbe, nijedna nema te sposobnosti toliko razvijene, kao što ih imaju ljudi. Način na koji smo se razvili u to što smo danas, te naša sposobnost učenja i prilagodbe na promjene u našoj okolini, razlozi su zašto smo na vrhu hranidbenog lanca i zašto smo toliko uspješni u preživljavanju.

Ljudi od malih nogu, pa sve do svoje smrti, kontinuirano uče nova, te usavršavaju već postojeća znanja. Stalno se moramo prilagođavati zbog okruženja koje se konstantno mijenja. Ima jako puno mišljenja i definicija o tome što je to zapravo učenje, no smatram da slijedeće, najbolje objašnjava taj pojam:

„Učenje je proces koji često nije pod našom kontrolom i koji je povezan sa okolinom u kojoj živimo i veze koje stvaramo. To uključuje dobivati signale osjetila; upravljati njima; tražiti veze i značenja; i protumačiti ih kako bismo mogli djelovali.“ [13]

Kada naučimo neku strategiju za određenu situaciju ili scenarij, nastojimo ju zapamtiti. Kada se ponovo dođe do takve ili slične situacije, u stanju smo ponovno primijeniti naučenu strategiju na isti ili prilagođen način. Upravo to je glavni cilj učenja, da se novostečeno znanje i iskustvo iskoriste kako bi se što bolje izvršio neki zadatak.

„Za računalo kažemo da uči iz iskustva E (*eng. experience*) s obzirom na neku klasu zadataka T (*eng. tasks*) i mjeru performansi P (*eng. performance*), ako se njegove performanse pri obavljanju zadatka T koje se mjere sa P poboljšaju sa iskustvom E “. [14]

Jednostavnije rečeno, potrebno je uzeti u obzir prethodno spomenute elemente, koji su vezani uz učenje. Moramo imati neki cilj ili zadatak koji želimo ostvariti i što uspješnije odraditi uz pomoć učenja. Nakon toga nam je potrebno iskustvo, jer je najbolje za poboljšanje znanja i performansi u provođenju bilo kojeg zadatka. Zadnja stvar koja nam je potrebna, je neka vrsta mjere kojom ćemo mjeriti uspješnost učenja, jer nam govori da li smo se u nečemu poboljšali, pogoršali ili pak je stanje ostalo.

3.1. Induktivno učenje

Kada imamo veliku količinu podataka ili bolje rečeno primjera, želimo iz njih izvući neko korisno znanje. Taj postupak se zove induktivno učenje i na taj način se pokušavaju pronaći pravila generalizacije. Kažemo da pravilo pokriva neki primjer, ako taj zadovoljava sve uvjete nekog pravila, od kojih se ono sastoji. Instance koje su pokrivene nekim pravilom se uklanjaju iz skupa razmatranja jer su za njih već ustanovljene pravilnosti. [8]

Za ovakvo učenje sustav koristi skup podataka, te mu za njih kažemo ispravne zaključke koje treba donijeti. Sustav pomoću tih podataka onda, kontinuirano podešava pravila, kako bi što točnije mogao raditi sa novim, dosad neviđenim primjerima. Takva pravila se generiraju prema danim primjerima, pomoću posebnih algoritama. Jedan takav algoritam koji na temelju skupa podataka kreira model koji generalizira veze i odnose između ulaznih i izlaznih atributa, zovemo algoritam za induciranje. Kažemo da se radi o empirijskom pristupu, jer se ovakva vrsta učenja bazira na provođenju pokusa i učenju na greškama. Nije moguće biti 100% siguran u točnost dobivenih pravila, zato jer je moguće da ima neki primjer kakav još nismo susreli. Upravo zbog toga je glavni cilj da se kreiraju pravila koja su što točnija, te koja se mogu brzo podesiti kada se ustanovi da su manjkava, odnosno kriva. [1], [4]

Postoji temeljna pretpostavka koja se veže uz induktivno učenje, a zove se hipoteza induktivnog učenja. „Bilo koja hipoteza koja dobro aproksimira ciljanu funkciju kada je primijenjena nad dovoljno velik skup podataka, će također dobro raditi sa neviđenim primjerima.“ [14]

Unatoč tome, pravila se često moraju dodatno podesiti, a te promjene ih onda mogu „ojačati“ ili „oslabiti“. Pod time se misli na to, da se uvjeti pod kojima pravilo vrijedi, mogu dodatno i ciljano postrožiti. Isto tako se može podesiti, tako da bolje generalizira, te da na taj način pokriva više primjera. Pravila se moraju dodatno podesiti, ukoliko ne pokrivaju primjer koji bi trebala i obrnuto, ako vrijede za neki primjer za koji ne bi trebala. [1]

3.2. Strojno učenje

Rudarenje podataka i strojno učenje su usko povezani, te zna desiti da se ta dva pojma pomiješaju. Potrebno je naglasiti, da je rudarenje podataka širi pojam, te da primjenjuje metode strojnog učenje, no, osim njih, može koristiti i druge tehnike na povrh njih.

Kada govorimo o strojnom učenju, glavni naglasak je na automatiziranim metodama, te na tome što taj pojam predstavlja i za nas kao ljude. Kroz veliki broj primjera dobiva se iskustvo, koje se onda dalje, primjenjuje, nad drugim zadacima.

Strojno učenje se koristi, jer ima neke određene prednosti nad tradicionalnim načinom rješavanja problema, primjenom direktnog programiranja. Umjesto da „direktno programiramo“ računalo da obavi neki zadatak ili riješi neki problem, tražimo metode, pomoću kojih će računalo samostalno doći do rješenja koje je u obliku modela, koji se onda može poboljšati, nakon što ga implementiramo. Rezultati koji se tako postignu, su u pravilo puno precizniji, a razlog tome je taj, da algoritmi strojnog učenja rade sami, te su u stanju učiti na jako velikoj količini primjera. Zbog toga mogu puno bolje procijeniti veze među podacima. Upravo ta karakteristika je, jedna od mnogih, zbog čega je strojno učenje savršeno za primjenu u okolinama, koje se konstantno mijenjaju. [15]

Želimo dobiti pravilo predviđanja koje je što pouzdanije, no postoje slučajevi, kada imamo neki problem, za koji nam je važno da dobijemo što točniji rezultat, a kod nekih nam je možda bitnije, razumijevanje rezultata i kako doći do njega. [15]

Jedan od glavnih ciljeva strojnog učenja je, da se izrade algoritmi za generalnu upotrebu ili bolje rečeno, algoritmi koji se mogu primijeniti za rješavanje različitih problema. Također se pokušavaju odgonetnuti odgovori na pitanja vezana uz učenje općenito, kao koja obilježja neki problem čine lakim ili teškim. [15]

„Strojno učenje se definira kao proučavanje računalnih programa, koji koriste algoritme i statističke modele, kako bi učili za učenje kroz zaključke i obrasce bez eksplicitnog programiranja“. [16]

Radi se dakle, o programiranju računala, kako bi se optimizirao kriterij izvedbe, koristeći pritom primjere ili prethodno iskustvo. Imamo model koji je definiran nekim parametrima, a učenje je podešavanje tih parametara. [8]

3.2.1. Primjena Strojnog učenja

Strojno učenje se razvija velikom brzinom, te se stalno nalaze nova područja i novi načini primjene. Neka područja primjene uključuju bankarstvo, gdje se koristi za otkrivanje mogućih prijevara i odobravanje kredita. Medicina, gdje se koristi za dijagnosticiranje bolesti ili analizu medicinskih slika. Također se pronalaze primjene i u transportu, gdje se koriti za upravljanje autonomnim vozilima, koja su u stanju samostalno doći od točke A do točke B. Ta autonomna vozila imaju velik broj senzora, pomoću kojih prikupljaju podatke o okolini, u realnom vremenu i koji te iste, onda obrađuje, kako bi donijeli što bolje odluke vezane uz putovanje.

Primjer korištenja strojnog učenja, koji bi mogao biti malo poznatiji, su prijedlozi videa za gledanje na YouTubeu, reklame koje se prikazuju na mobilnim aplikacijama i web stranicama, prijedlozi proizvoda za kupovinu na Amazonu i slično. Sadržaj koji se prikazuje nije slučajno odabran, već na temelju ponašanje korisnika i pretpostavkama o tome, što bi mu moglo biti interesantno.

Koristi se i za predviđanje budućih vrijednosti na burzi, u proizvodnji, gdje omogućuje povećanje efikasnosti i optimizaciju procesa, prepoznavanje lica na snimkama i fotografijama, prepoznavanje glazbe, predlaganje proizvoda za kupnju i reklama, klasifikacija e -mailova i u mnogim drugim područjima.

3.3. Vrste strojnog učenja

3.3.1. Nenadzirano učenje (*Unsupervised*)

Pod ovu vrstu učenja, spadaju tehnike strojnog učenja, kod kojih nema eksperta ili neke druge osobe koja označava podatke, u smislu, da im određuje vrijednost. Na raspolaganju su samo ulazni, podaci kojima nisu pridjeljenje oznake, niti ne znamo kojoj kategoriji pripadaju. Glavni cilj je da se pokušavaju pronaći pravilnosti i da se pokuša grupirati podatke po sličnosti, odnosno, njihovim razlikama.

Jedan od problema je, da je teško evaluirati rezultate. Može se desiti da će izgledati, kao da su podaci dobro grupirani, iako to nije tako. Drugi problem je, da algoritmi dobro rade kada im definiramo broj grupa u koje moraju svrstati podatke, no, može biti, da je taj broj, upravo podatak koji tražimo.

3.3.2. Metode nenadziranog učenja

3.3.2.1. Grupiranje (*Clustering*)

Ljudi se već jako dugo bave grupiranjem pojava, jer smo oduvijek imali potrebu opisivati najistaknutije karakteristike objekata i da im dodijelimo neku oznaku ili odredimo vrstu. Iako se nama to čini dosta jednostavnim i prirodnim, za računala je to malo teže. Koristi se za rješavanje problema formiranja kategorija za promatrane entitete i dodjeljivanje istim odgovarajuće grupe. [4]

Grupiranje (*eng*. *clustering*) objekata, jedna je od najčešćih metoda koja se koristi kod učenja bez nadzora, čiji je cilj, da se dobije ideja o tome što se generalno dešava, a što ne. Među podacima obično postoji neka vrsta strukture, postoje pravilnosti zbog kojih se neke pojave javljaju češće od drugih. Model Grupiranja, grupira instance u grupe, a za to koristi sličnost vrijednosti njihovih atributa.

Drugi naziv za grupiranje koji je možda pogodniji, je particioniranje, jer se zapravo rješava problem, gdje je potrebno podijeliti skup podataka. Podaci su neoznačeni, te ih treba tako grupirati, da slične instance pripadaju istoj particiji. Imamo dvije glavne vrste algoritama koji se koriste za particioniranje podataka, koji se razlikuju po načinu preslikavanja podataka u particije. [12]

Jedna vrsta je, izrazito (*eng*. *Strict*) grupiranje, gdje neka instanca može pripadati jednoj i samo jednoj particiji. Druga vrsta je, neizrazito (*eng*. *Fuzzy*) grupiranje, gdje jedna instanca može pripadati više particija. Prilikom odabira algoritma potrebno je također, razumjeti dvije ključne stvari, a to su, koje mjere sličnosti algoritam koristi i koji su kriteriji razdvajanja među particijama. [12]

Također je moguće da je zadan i broj grupa, odnosno particija, u koje algoritam mora razvrstati podatke ili pak se može dinamično povećavat. Jedan od najpopularnijih algoritama koji nema unaprijed definiran broj grupa, a ni potrebu da bude izrazito ili neizrazito se zove *k-mean* grupiranje. K se odnosi na broj grupa u koje je potrebno razvrstati podatke. [12]

3.3.3. Nadzirano učenje (Supervised)

Ove metode pokušavaju otkriti veze između ulaznih (neovisnih) i izlaznih (zavisnih) varijabli, te se naučiti funkciju koja preslikava vrijednosti s nekog ulaza na odgovarajući izlaz. Te novootkrivene veze se onda prikazuju u strukturiranom obliku, koji nazivamo model. Pomoću njega se, opisuju i objašnjavaju pravilnosti sakrivene u podacima, te se onda mogu koristiti, kako bi se predvidjela vrijednosti nekog izlaznog atributa, kada su nam poznate vrijednosti ulaznih. [4]

Cilj nadziranog učenja je da se nađe funkcija kod koje je rizik pogrešnog predviđanja što manji, odnosno, da je greška predviđene vrijednosti što manja. Često se radi o iterativnom procesu koji se ponavlja, toliko dugo, dok se ne postigne neki rezultat za koji smatramo da je prihvatljiv.

Razlika između učenja, pod i bez nadzora, je da imamo određene podatke koji su već označeni i pomoću kojih zapravo učimo. Dobra stvar je, da zahvaljujući tim označenim podacima, možemo bolje procijeniti kvalitetu rezultata. Osim što je za ovu metodu potrebna dovoljna količina podataka (što ponekad može biti problem), također može doći do nečeg što se zove pretreniranost (*eng. overfitting*), no, o tome ćemo kasnije.

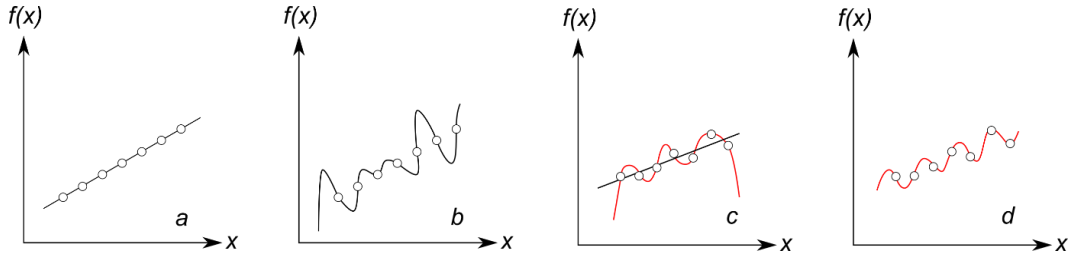
3.3.3.1. Zadaće nadziranog učenja

Kod ovakvih metoda učenja, sama veličina skupa podataka koji se koristi za kreiranje modela i performanse koje se dobe nakon učenja „pozitivno koreliraju“. To znači da, da što više imamo podataka za treniranje, to će biti bolji rezultati predviđanja. No, u praksi su stvari malo kompliciranije, jer je možda potrebno ograničiti skup do neke mjere. Razlog tome su najčešće problemi, kao što su ograničenja zbog premalo resursa, a i veliki skup podataka ujedno i znači da će vrijeme učenja duže trajati. [4]

Glavni zadatak nadzirnog učenja je, da se za dani skup podataka, koji se sastoji od N parova ulaza i izlaza, koji služe kao primjeri za svaki Y_j koji je generiran od nepoznate funkcije $y = f(x)$, otkrije funkcija h, koja je približno jednaka pravoj funkciji f. Ovdje x i y ne moraju biti brojke, već mogu biti bilo koja vrijednost. Funkcija h je hipoteza, a učenje je pretraživanje prostora svih mogućih hipoteza, u potrazi za jednom, koja će funkcionirati dobro sa novim, dosad neviđenim primjerima. Za hipotezu kažemo da dobro generalizira, ako točno predviđa vrijednost od y za nove primjere. Kada je izlaz y konačan skup diskretnih vrijednosti (klasa / oznaka), problem učenja naziva se klasifikacija, a kada se radi o kontinuiranoj vrijednosti (broj) se zove regresija. [17]

Kada prilagođavamo funkciju jedne varijable nekim podatkovnim točkama, te točke se nalaze na (x, y) plohi, gdje je $y = f(x)$, a f je nepoznata funkcija koju ćemo pokušati procijeniti sa funkcijom h. Ta funkcija je odabrana iz prostora hipoteza H, odnosno skupa svih funkcija, koje model može vratiti i koji je za ovaj primjer skup polinoma. Prostor hipoteza ili prostor pretraživanja ili prostor parametara, je onaj u kojem se nalaze potencijala rješenja nekog problema. Broj dimenzija prostora ovisi o broju varijabli za koje tražimo vrijednosti, pa tako možemo imati jedno, dva, tro ili n-dimenzionalni prostor. [1], [17]

Sljedeće slike prikazuju primjere prilagođavanja funkcije jedne varijable nekim podatkovnim točkama. Primjeri su točke koje se nalaze na (x, y) plohi, gdje je $y = f(x)$. Iako ne znamo koja je funkcija f, pokušat ćemo ju približno procijeniti sa funkcijom h, odabranom iz prostora hipoteza H. [17]



Slika 3 Primjeri hipotenuza [17]

U primjeru **a** vidimo podatke koji se u potpunosti podudaraju sa ravnom linijom, ta linija se naziva konzistentna hipoteza (*eng. consistent hypothesis*), jer se podudara sa svim podacima. U primjeru **b** se može vidjeti, da je korišten polinom visokog stupnja, koji se također podudara sa svim podacima, ali i ujedno ukazuje na temeljni problem induktivnog učenja. Problem se odnosi na to, kako da odaberemo jednu, od velikog broja mogućim hipotezama, koje se podudaraju sa podacima. Jedan prijedlog je da se od svih mogućih hipoteza koje su konzistentne s podacima, odabere ona, koja je najjednostavnija. Ovaj princip je poznat kao „*Ockham's razor*“, po engleskom filozofu iz 14. stoljeća i govori kako su jednostavnija objašnjenja puno vjerojatnija i da bi sva nepotrebna složenost trebala odbaciti. [17]

Postoji kompromis između prostora hipoteza i složenosti pronalaženja dobre hipoteze unutar tog prostora. Iako je moguće da će jednostavnija hipoteza, odnosno jednostavniji model, imati više grešaka kada radi sa podacima iz skupa za treniranje, vjerojatno će manje griješiti kasnije sa neviđenim primjerima, pošto neće doći do pretreniranosti (*eng. overfitting-a*). Još jedan razlog zašto je dobro preferirati jednostavnije hipoteze, je činjenica, da ćemo tu hipotezu kasnije htjeti i koristiti nakon što je pronađemo, a izračun linearne funkcije nam garantira brze rezultate, dok za neke složenije možda nećemo ni dobiti odgovor. [8], [17]

U primjeru **c** je prikazan drugi skup podataka za koji ne postoji ravna crta, već je potreban polinom šestog stupnja za točno preklapanje. Općenito, postoji razmjena između kompleksnih hipoteza koje se jako dobro podudaraju s podacima koji se koriste za trening i jednostavnijih hipoteza, koje bolje generaliziraju. Za primjer **d** smo proširili prostor hipoteza kako bi se omogućili polinomi i nad $sin(x)$. Vidimo da se podaci iz primjera **c** mogu točno preklapati, koristeći jednostavniju funkciju oblika $ax + b + c\,sin(x)$. Ovaj primjer pokazuje, kako je zapravo važan odabir prostora mogućih hipoteza. [17]

Uspješnost strojnog učenja stoga ovisi o tome, da li imamo dovoljnu količinu podataka, te o tome koliko neko pravilo ili model griješi. Želimo da pravilo, bude što jednostavnije, no valja napomenuti, da je dosta teško uskladiti te uvjete. Naime, radi se o problemu koji je bolje poznat i koji se može javiti kao pretreniranost (*eng*. *overfitting*) ili podtreniranost (*eng. underfitting*). Ako je pravilo prejednostavno, velika je vjerojatnost da će biti dosta grešaka i onda imamo problem podtreniranosti. Ukoliko pak želimo da pravilo radi čim manje grešaka, vjerojatno će postati previše složeno, te će doći do pretreniranosti. Stoga je potrebno, naći neku srednju mjeru, koja će nam omogućiti da pravilo daje što točnije rezultate, a da pritom nije previše složeno.

3.3.4. Metode nadzirnog učenja

3.3.4.1. Klasifikacija (*Classification*)

Za početak je potrebno skrenuti pozornost na problem učenja koncepta, kod kojeg se radi o preslikavanju. Imamo skup oznaka, odnosno kategorija, u koje preslikavamo primjere (instance), a to preslikavanje zovemo konceptom. Prema tome, koncept je funkcija oblika $c: X -> \{0,1\}$. Za X velimo da je domena ili skup svih mogućih primjera, a nekoj instanci ili primjeru s obzirom na vrijednosti njegovih atributa, dodjeljujemo oznaku odnosno kategoriju. Takav jedan skup instanci koje su preslikane u istu kategoriju i nad kojim se generalizira, još nazivamo klasom koncepata. [15]

Cilj kod problema klasifikacije je da algoritam strojnog učenja kreira generalno pravilo ili bolje rečeno, da inducira klasifikator (*eng.classifier*), koji će najpreciznije odrediti kategoriju instance, s obzirom na dane ulazne atribute. Svrstavanje u grupe ili klasifikacija dane ulazne instance, se radi na temelju podataka iz skupa za trening, jer se u njemu nalaze parovi ulaznih i izlaznih instanci, koji su kategorizirani po klasama. Najjednostavnije rečeno, provjerava se da li neki objekt pripada nekoj kategoriji

Kada se radi o skupu sa više vrijednosti (na primjer { zelena, crvena, plava, ljubičasta}), tada se problem učenja naziva klasifikacija, no dosta problema su zapravo binarni kod, kojih imamo samo dvije vrijednosti, koje su suprotnosti. Kada je rezultat konačan skup sa samo dvije vrijednosti, tipa dali je nešto istinito ili nije, kažemo da je binarna klasifikacija. U svakom slučaju, traži se funkcija koja preslikava skup svih mogućih primjera, u unaprijed definiran skup oznaka. [12], [17]

3.3.4.2. Regresija (Regression)

Za razliku od klasifikacije gdje nekoj instanci određuje kategorija, kod regresije se pokušava predvidjeti numerička vrijednost, no princip ostaje isti.

Postoji ulaz, odnosno skup instanci i izlaz, te pokušavamo pronaći najbolji način preslikavanja s ulaza na izlaz. Imamo, dakle model, koji je funkcija i parametre pomoću kojih pokušavamo predvidjeti neki broj, kada se radi o regresiji ili klasu, kada se radi o klasifikaciji. Program strojnog učenja optimizira parametre, tako da se greška kod predviđanja minimizira ili drugačije rečeno, pokušava napraviti procjenu, koja je što više bliža točnim vrijednostima iz skupa za treniranje. [8]

3.3.5. Hibridno učenje (Semi-supervised Learning)

Kao što i sam naziv govori, ova metoda je kombinacija prethodnih dviju i na neki način rješava problem otežane evaluacije, koji se javlja kod učenja bez nadzora. Koristi se kada imamo podatke kod kojih su uglavnom svi označeni, ali kod jednog dijela, nam iz nekog razloga, nedostaju oznake. Problem se rješava tako da, nakon što se podaci grupiraju, možemo provjeriti ispravnost grupiranih podataka, pomoću onih koji imaju oznake.

U praksi se često zna desiti da imamo samo nekoliko označenih primjera, a ostatak je neoznačen. Na temelju podataka koji su označeni, onda treba donositi zaključke, kako bi se što bolje odredile klase preostalih. Također treba imati na umu, da možda ni oni označeni podaci nisu u potpunosti točni. Osim nasumičnih grešaka ili nepostojećih vrijednosti, također su mogući i loši izvori podataka. To je često slučaj kada su izvori ljudi koji onda slučajno ili namjerno, daju krive podatke, koji dodatno otežavaju. [17]

4. Stablo odlučivanja

Govorili smo o učenju općenito, o strojnom učenju, nekim metodama kojima se koristi strojno učenje, te o klasifikaciji i regresiji. Stablo odlučivanja je jedan od najjednostavnijih i najpopularnijih oblika strojnog učenja, koji koristi induktivno učenje, za generiranje modela. Primjenjuje se za rješavanje raznih problema, a naziv zahvaljuje svojoj strukturi, koja je slična stablu.

Stablo odlučivanja je „prediktivni model koji se može koristiti za predstavljanje modela klasifikacije i regresije, a u disciplinama koje se bave istraživanjem kako bolje unaprijediti upravljanje, se odnose na hijerarhijske modele odluka i njihove posljedice“. [4]

Radi se o metodi za aproksimaciju, kod koje se naučena funkcija, odnosno model, prikazuje u obliku stabla odlučivanja. Model se generira automatski, tako da se nad skupom podataka primjeni odgovarajući algoritam, te na taj način dobivamo hijerarhijsku strukturu, koja implementira strategiju podijeli i vladaj. [14]

Funkcija kao ulaz, uzima vektor vrijednosti atributa i vraća jednu izlaznu vrijednost, odnosno odluku. Odluka se donosi nizom testova, a ulazne i izlazne vrijednosti koje se koriste za donošenje odluke, mogu biti diskretne ili kontinuirane. Svaka odluka koja se donese, je u obliku disjunktivne normalne forme i može se slijediti, prateći put od korijena stabla. [17]

Kada se koristi parametarska procjena, definira se model cijelog ulaznog prostora i saznajemo njegove parametre iz svih podataka skupa za treniranje. Tada koristimo taj model i parametre za nove testne primjere. Stablo odlučivanja je efikasna ne-parametarska metoda, što znači da ne uključuje nikakve pretpostavke o obliku ili parametrima frekvencijske raspodjele. Kod takve procjene se ulazni prostor dijeli na lokalne regije definirane mjerom udaljenosti, te se za svaku ulaznu vrijednost generira odgovarajući lokalni model od podataka, za tu regiju. Za razliku od parametarskog pristupa, jedina pretpostavka ovdje je, da slične ulazne vrijednosti imaju slične izlazne vrijednosti. Ukratko, algoritam koristi prethodne pojave i odgovarajuću mjeru udaljenosti, kako bi dao izlaznu vrijednost. [8]

Zbog jednostavnosti i transparentnosti, stablo odlučivanja je popularna tehnika rudarenja podataka. Kažemo jednostavna, jer ih nije teško razumjeti, samo se treba slijediti put od korijena pa do lista. Pošto se najčešće prikazuju kao grafičke hijerarhijske strukture, lako ih je interpretirati. No, također je moguće da i stabla postanu komplicirana, a to se događa kada ima previše čvorova. Tada njihova grafička reprezentacija postane beskorisna, te je potrebno pojednostaviti stablo. [4]

4.1. Anatomija stabla odlučivanja

Svima su dobro poznata stabla koja susrećemo svakodnevno u prirodi. Bez obzira na vrstu o kojoj se radi, svako ili barem većina stablo ima korijen, grane i listove. Osim onih u prirodi, imamo stabla u matematici, koja također imaju elemente koji su slični i predstavljaju dijelove pravog stabla.

Kad govorimo o stablima u matematici, mislimo na grafove, koji moraju zadovoljiti neke uvjete, odnosno koji imaju neke karakteristike, zbog kojih ih nazivamo stablima.

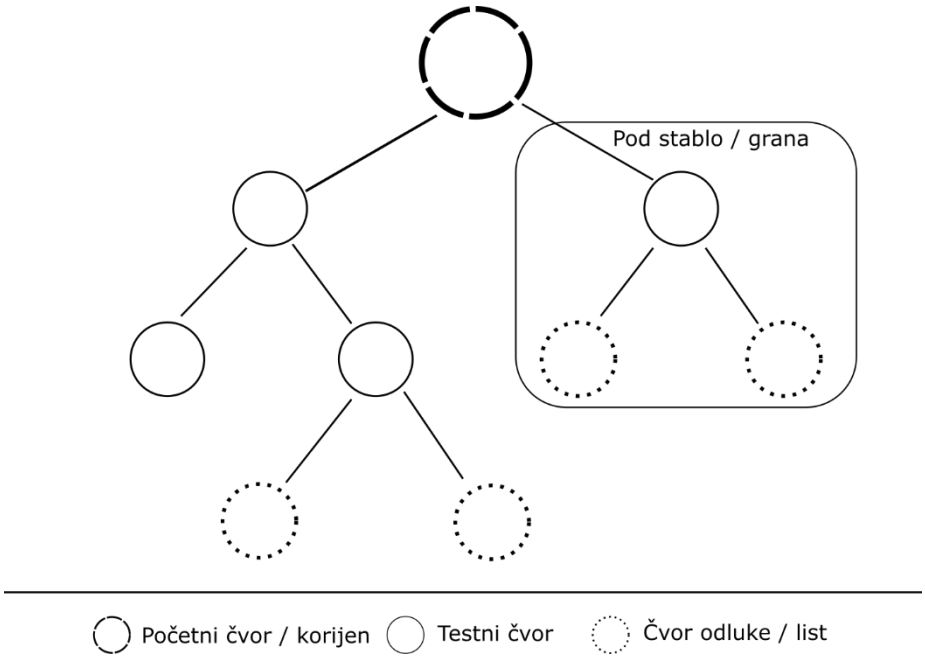
- Odnosno, između svaka dva vrha, postoji jedinstven put
- broj bridova koji je jednak broju vrhova umanjen za jedan
- je povezani aciklički graf
- svaki brid je rezni brid

Vrsta stabla koja nas zanima, u ovom slučaju su korijenska stabla, odnosno uređeni par, koji se sastoji od nekog stabla i jednog njegovog istaknutog vrha, koji nema ulaznih bridova predstavlja korijen. [18]

Kad govorimo o stablima odlučivanja, mislimo na usmjereni korijenski graf, za čije se opisivanje koriste slični, ali malo drugačiji pojmovi, nego za one u matematici. Stabla odlučivanja se sastoje od čvorova, koji su povezani bridovima, koje zovemo grane.

Svako stablo ima korijen, odnosno jedan čvor bez ulaznih bridova, koji predstavlja sve instance, koje se onda dalje dijele. Čvor koji se tako dalje grana, zovemo čvor odluke ili testni čvor, koji najčešće testira jedan atribut pojave, te dijeli na dvije ili više grana. Svaki čvor je testna funkcija za neku određenu karakteristiku, sa diskretnim izlazima kojima se označavaju grane. Za ulazne podatke koji se testiraju na svakom unutarnjem (testnom) čvoru, se odabire jedna od izlaznih grana iz tog čvora, koje odgovaraju rasponu vrijednosti. Ti rasponi moraju biti međusobno isključivi (disjunktni) i cjeloviti jer osiguravaju, da se svaka instanca preslika u jednu klasu. Proces koji se koristi kako bi neki čvor stabla rastavili na dva ili više, se zove grananje (*eng. splitting*). Bilo koji unutarnji čvor stabla koji se dalje grana, nazivamo roditeljem, a čvorove koji izlaze iz njega, djecom. [4]

Svaka testna funkcija nekog unutarnjeg čvora definira jednu diskriminantu, koja dijeli prostor instanci. Taj postupak se ponavlja toliko dugo, dok ne dođemo do lista, odnosno završnog čvora, koji se ne grana dalje i u kojem se nalazi vrijednost koju smo tražili. Takav čvor stabla odlučivanja zovemo čvor odluke, završni čvor ili jednostavno, list. Svaki list ima izlaznu oznaku, koja je u slučaju klasifikacije klasa kojoj je pridružen, odnosno numerička vrijednost, u slučaju regresije. Također, definira jednu lokalnu regiju ulaznog prostora, u kojoj se nalaze sve instance koje imaju isti izlaz. Same granice tih regija su definirane diskriminantama, odnosno, testnim čvorovima stabla. Preslikavanje se odvija navigacijom od korijena pa do lista, s obzirom na testne čvorove. Na taj put možemo gledati kao na jedno složeno pravilo, koje se sastoji od niza manjih, koja se mogu predstaviti u obliku *IF-THEN*. Upravo to je jedan od razloga, zašto su stabla odlučivanja tako popularna i lako shvatljiva. [8], [4]



Slika 4 Dijelovi stabla odlučivanja

4.2. Vrste stabla

Prije smo govorili o metodama nadzirnog učenja regresiji i klasifikaciji, a što se tiče vrsta stabla odlučivanja, imamo stabla klasifikacije i stabla regresije. Obije vrste rade na sličan način, ali i također postoje razlike među njima. Ukoliko se stablo koristi za rješavanje klasifikacijskog problema, onda ga nazivamo stablom klasifikacije, a ukoliko se koristi za zadatke regresije, nazivamo ga, regresijskim stablom. [4]

Iako ima dosta različitih metoda koje se mogu koristiti za klasificiranje, predviđanje kategorije ili točnije, za klasifikaciju nekog objekta ili instance, u skup, unaprijed definiranih klasa. Stabla odlučivanja su obično prva stvar, koja se pokuša, kad se iz podataka želi naučiti metoda za tu svrhu. [17]

Iako se stabla odlučivanja uglavnom koriste za zadatke koji su vezani uz klasifikaciju, također se mogu koristiti i za regresijske zadatke. Rad sa varijablama koje imaju kontinuirane vrijednosti, je iznimno važno, jer puno procesa daju upravo numeričke podatke. Stablo regresije se koristi za predviđanje numeričke izlazne vrijednosti. Svaki list stabla regresije, nema jednu vrijednost, već linearnu funkciju, za neki podskup numeričkog atributa. Algoritam za učenje koji se primjenjuje, mora odlučiti kada prestati s grananjem i početi primjenjivati linearnu regresiju, nad atributima. [17]

Obije vrste rade na sličan način, samo što se stabla regresije koriste kada se predviđa vrijednost ciljane varijable koja je kontinuirana, a stabla klasifikacije kada je kategorijska. Kod regresijskih stabla, dobivena vrijednost listova je zapravo, promatrana srednja vrijednost, koja spada pod tu regiju. Vrijednost lista kod stabla klasifikacije, se procjenjuje za svaki list zasebno, kao najčešća vrijednost. Oba stabla kreiraju particije, odnosno, dijele prostor varijabli u regije. Također i jedno i drugo stablo slijedi pohlepni pristup TOP-DOWN, bolje poznat kao binarno dijeljenje. Kažemo da je TOP-DOWN pristup jer ide od korijena u kojem se nalaze sve instance, te ih dijeli na pohlepan način, tražeći najbolju varijablu po kojoj će granati, bez obzira na odabir budućih varijabli, po kojima će granati. [19]

4.3. Grananje

Svako grananje stabla započinje sa definiranjem problema ili postavljanjem pitanja na koje odgovor daje list. To pitanje se onda rastavlja na moguća rješenja koja mogu, ali ne moraju biti konačna, a svako od tih rješenja se preispita, razmatrajući rezultat koji treba ili ne treba dodatnu odluku. Ukoliko je to potrebno, proces se nastavlja i razmatraju se rezultati, za nove odluke. [12]

Kako bi uopće mogli započeti proces grananja, potrebno je odabrati korijen, odnosno prvi atribut, po kojem će se granati i koji će dati najbolji rezultat klasifikacije primjera. Iako odabir atributa može biti nasumičan, velika je vjerojatnost da će rezultati biti jako loši. Postoji nekoliko načina pomoću kojih možemo testirati i odabrati najbolji atribut za grananje, te na taj način dobiti što bolji i točniji rezultat.

Sam odabir se kvantificira, koristeći mjeru nereda (*eng. impurity meassure*). Ako nakon grananja, sve instance koje su odabrale neku granu, pripadaju istoj klasi, velimo da je taj podskup, odnosno čvor, čist. U tom slučaju, nema potrebe za daljnjim grananjem, te možemo postaviti završni čvor ili list sa oznakom klase. Ukoliko neki čvor nije čist, potrebno ga je dalje razgranati, kako bi se smanjila nečistoća, odnosno nered. Postoji više atributa po kojima možemo granati, a tražimo onaj kojim će se najviše smanjiti nered, jer želimo kreirati najmanje stablo. Ukoliko podskup nakon razdvajanja ima manji nered, trebat će manje dodatnih grananja. [8]

4.3.1. Mjere nereda

Kod većine algoritama koji se koriste za induciranje stabla odlučivanja, funkcije grananja, u obzir uzimaju jedan atribut, za koji smatraju da je najbolji. Odabir atributa, može se vršiti pomoću nekoliko mjera nereda, a neke od najpopularnijih će biti objašnjene u nastavku.

S obzirom na nasumičnu varijablu x, koja može imati k diskretnih (različitih) vrijednosti, koje su raspodijeljene prema $P = (p_1, p_2, \dots, p_k)$, mjera nereda je funkcija $\Phi: [0,1]^k \rightarrow R$ koja zadovoljava slijedeće uvjete: [4]

- $\Phi(P) \geq 0$
- $\Phi(P)\ je\ minimalan\ ako\ \exists i\ takav\ da\ je\ komponenta\ p_i = 1$
- $\Phi(P)\ je\ maksimalan\ ako\ \forall p_i, 1 \leq i \leq k, p_i = 1/k$
- $\Phi(P)\ je\ simetričan\ s\ obzirom\ na\ komponente\ od\ P$
- $\Phi(P)\ je\ svugdje\ različita\ unutar\ dosega$

4.3.1.1. Entropija

Entropija (*eng. entropy*) je mjera nereda ili nesigurnosti neke nasumične varijable, koja je vezana uz njezine vrijednosti. Radi se o mjeri koja nam govori o čistoći proizvoljne varijable ili proizvoljnog skupa koji sadrži pozitivne i negativne primjere nekog ciljanog koncepta. Općenito, entropija neke nasumične varijable V sa vrijednostima v_k, gdje svaka ima vjerojatnost $P(v_k)$ definirana kao: [17], [14]

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

Pomoći će definirati $B(q)$ kao entropiju nasumične Boolean varijable, koja je istinita s vjerojatnošću q:

$$q = \frac{p}{p+n}$$

$$B(q) = -((\mathrm{q} \log_2 q) + (1-q)\log_2(1-q))$$

Nasumična varijabla koja ima samo jednu vrijednost, nema nesigurnost i njezina entropija je stoga, jednaka nuli. Što je veća entropija, to je teže donositi zaključke, prema tim informacija. [17]

Ukoliko imamo neki atribut A sa d različitih vrijednosti, on dijeli skup podataka za trening E na d pod skupova. Svaki od tih pod skupova ima pk pozitivnih i nk negativnih primjera. Ukoliko se odlučimo za tu granu, trebati će nam dodatnih $B(pk/(pk+nk))$ bitova informacija, da bi odgovorili na pitanje. Nasumično odabran primjer iz skupa za treniranje, ima $k-tu$ vrijednost za atribut sa vjerojatnošću $(pk+nk)/(p+n)$, pa je stoga očekivana, preostala entropija atributa A nakon grananja: [17]

$$Entropija\ nakon\ grananja\ (A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(qk)$$

Preostala entropija se dakle, odnosi na vrijednost entropije, nakon što se skup svih primjera razdvoji koristeći atribut A. Ta vrijednost je suma entropija svih pod skupova od A, ponderirani udijelim primjera koji pripadaju $dk,$ odnosno nekoj kategoriji od A. [14]

Kod logaritmiranja se i dalje koristi baza 2, jer entropija određuje najmanji broj bitova koji su potrebni za kodiranje točnosti klasifikacije instance. Točnije rečeno, veli nam najmanji broj bitova, koji su u prosjeku potrebni po simbolu. za prikazivanje vrijednosti neke promatrane varijable. Također je dobro imati na umu, da ako imamo $d > 2$ klasa, najveća entropija je $\log_2 d$ kada je vjerojatnost $q = 1/d$. [20], [14]

Entropija se mjeri u rasponu [0,1]. Ovisno o broju klasa, može biti veća od 1, ali i dalje vrijedi isti princip. Veći broj znači da je veći nered, što znači da je odabrani atribut gori odabir od nekog, s manjom entropijom. Kao što je vidljivo iz grafa, entropija će biti najveća na sredini, a najmanja kada se nalazimo na 0 ili 1 osi x.

4.3.1.2. Informacijska dobit

Informacijska dobit (*eng. Information gain*) je mjera efektivnosti nekog atributa, koja nam govori koliko iznosi očekivano smanjenje nereda (entropije), prouzrokovano razdvajanjem skupa svih primjera prema odabranom atributu. Ta vrijednost je broj sačuvanih bitova, kada se kodira ciljana vrijednost nasumičnog člana od S, kada znamo vrijednost atributa A. [14]

$$Dobivene\ informacije(S, A) = Entropija(S) - Entropija\ nakon\ grananja\ (A)$$

$$= B\left(\frac{p}{p+n}\right) - Entropija\ nakon\ grananja\ (A)$$

Jednostavnije rečeno, informacijska dobit je mjera koja nam govori kako dobro neki atribut razdvaja primjere iz skupa podataka prema ciljanoj klasifikaciji, odnosno, koliko će se smanjiti nered cijelog skupa podataka (korijena), ako se odlučimo za grananje po odabranom atributu. Što je dobiveni broj veći, to znači da je entropija, odnosno nered nakon grananja, bio puno manji.

4.3.1.3. Gini indeks

Gini indeks ili Gini koeficijent je zapravo statistička mjera raspodijele, koju je razvio Talijanski statističar Corrado Gini, 1912. godine. Iako se koristi kao mjerilo ekonomske nejednakosti, koje mjeri raspodjelu dohotka među stanovništvom, može se koristiti za mjerenje bilo koje raspodijele. [7]

Gini indeks je vjerojatnost pogrešnog klasificiranja nasumično odabranog elementa u skupu podataka, ako je nasumično označen prema raspodjeli klasa u skupu podataka.“ [21]

Radi se o mjeri koja se koristi, kako bi se kvantitativno ocijenilo grananje, te nam daje odgovor na pitanje, kolika je vjerojatnost da se pogrešno klasificira neki podatak. Baš kao i entropija, ima isti raspon vrijednosti [0,1], gdje nula znači da je nered mali, odnosno da ga nema, a jedan znači da je veliki nered.

Recimo da imamo skup podatak S i atribut A sa d klasa. Neka je $q(i)$ vjerojatnost da ćemo odabrati neku instancu ili podatak klase di, tada je Gini indeks definiran kao:

$$G(S) = \sum_{i=1}^{d} q(i) \times (1 - q(i))$$

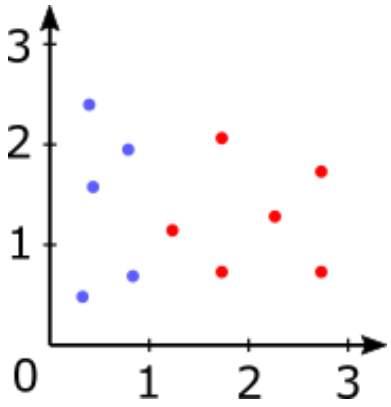
$$G(A) = \sum_{i=1}^{d} \frac{Sp}{S} \times Gi = \frac{pi + ni}{p + n} \times Gi$$

$$Gini\ gain = \ G(S) - G(A)$$

Najbolji odabir je onaj, koji će najviše smanjiti nered skupa podataka S , nakon što ga podijelimo po atributu A. Gini gain je broj, koji nam govori, koliko nereda je uklonjeno grananjem, a što je veći, to bolje. Dobiva se tako, da se od gini indeksa početnog cijelog skupa, oduzmu ponderirane (*eng. weighted*) mjere nereda dobivenih grana, od originalnog nereda. [21]

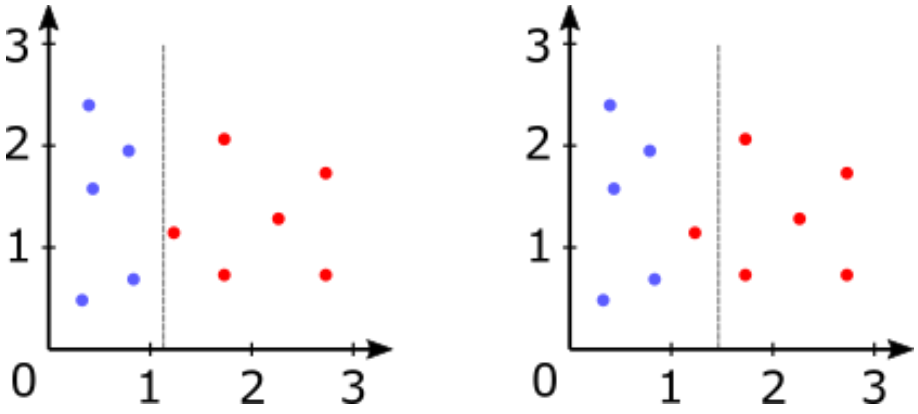
4.3.2. Primjer računanja nereda

Recimo da imamo skup podataka S, koji se sastoji od jedanaest elemenata i koji je prikazan pomoću grafa ispod. Taj graf predstavlja sve elemente skupa podataka ili bolje rečeno, sve podatke, koji se nalaze u početnom čvoru (korijenu).



Slika 5 Graf skupa S

Prije smo napomenuli da je sveukupno 11 točaka, od čega imamo 5 plavih i 6 crvenih. Prva korak je da se ustanovi trenutno stanje ili bolje rečeno, trenutni nered, kako bi se kasnije moglo usporediti sa stanjem, nakon grananja.



Slika 6 Primjer dvaju različitih grananja

Grafovi iznad, predstavljaju dva različita grananja, po dva različita atributa. Prvo grananje dijeli skup na dva podskupa, jedan u kojem imamo 5 plavih i jedan u kojem imamo 6 crvenih točaka. Drugo grananje pak ga dijeli, na jedan podskup u kojem imamo 5 plavih i 1 crvenu točku i jedan, u kojem imamo 5 crvenih točaka. Odmah možemo jasno vidjeti koje je bolje grananje, ali ćemo to dodatno potvrditi i brojčano dokazati. U nastavku će biti prikazani primjeri mjera nereda entropija i gini indeks.

4.3.2.1. Računanje Entropije

$$P(plava) = \frac{broj\ plavih\ točaka}{broj\ plavih\ točaka + broj\ crvenih\ točaka} = \frac{5}{5+6} = \frac{5}{11} \approx 0.46$$

$$P(crvena) = \frac{broj\ crvenih\ točaka}{broj\ plavih\ točaka + broj\ crvenih\ točaka} = \frac{6}{5+6} = \frac{6}{11} \approx 0.54$$

1. Entropija trenutnog stanja:

$$E(S) = -\sum_{k} P(v_k) \log_2 P(v_k)$$

$$E(S) = -(0.46 \log_2 0.46 + 0.54 \log_2 0.54) = \mathbf{0.99403}$$

2. Entropija nakon prvog grananja:

$$Entropija\ lijeve\ grane = -\left(\frac{5}{5}\log_2\frac{5}{5} + 0\right) = -(0+0) = \mathbf{0}$$

$$Entropija\ desne\ grane = -\left(\frac{6}{6}\log_2\frac{6}{6} + 0\right) = -(0+0) = \mathbf{0}$$

$$ENG\ (S) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B(qk)$$

$$ENG\ (S) = \left(\frac{5}{11} \times 0\right) + \left(\frac{6}{11} \times 0\right) = \mathbf{0}$$

3. Informacijska dobit (*eng. Information gain*) nakon prvog grananja:

$$Dobivene\ informacije = Entropija(S) - Preostala\ entropija$$

$$Dobivene\ informacije\ nakon\ prvog\ grananja = \mathbf{0.99403} - \mathbf{0} = \mathbf{0.99403}$$

1. Entropija nakon drugog grananja:

$$Entropija\ lijeve\ grane = -\left(\frac{5}{6}\log_2\frac{5}{6}+\frac{1}{6}\log_2\frac{1}{6}\right) = \mathbf{0.650022}$$

$$Entropija\ desne\ grane = -\left(\frac{5}{5}\log_2\frac{5}{5}+0\right) = -(0+0) = \mathbf{0}$$

$$ENG\ (S) = \sum_{k=1}^{d}\frac{p_k+n_k}{p+n}B\left(\frac{p_k}{p_k+n_k}\right) = \sum_{k=1}^{d}\frac{p_k+n_k}{p+n}B(qk)$$

$$ENG\ (S) = \left(\frac{6}{11}\times \mathbf{0.650022}\right)+\left(\frac{5}{11}\times 0\right) = \mathbf{0.35456}$$

2. Informacijska dobit (*eng. Information gain*) nakon drugog grananja:

$$Dobivene\ informacije\ nakon\ drugog\ grananja = \mathbf{0.99403} - \mathbf{0.35456} = \mathbf{0.6395}$$

Iz grafa prvog grananja se jasno vidi, da je skup bio savršeno podijeljen. Linija je dijelila skup na dva čista podskupa, u kojem su se nalazili isti elementi, te je zato entropija svake grane iznosila 0. Informacije dobivene nakon prvog grananja, jednake su entropiji prije grananja, a to znači, da smo nered sveli na 0 i da ga nema nakon grananja.

Na drugom grafu možemo vidjeti, da smo također dobili dva podskupa. Prvi podskup je imao 6 točaka gdje je jedan bila crvene, a preostalih 5 plave boje. Iz tog razloga smo u toj grani imali entropiju, a u drugoj nismo. Ako još usporedimo informacije dobivene prvim grananjem i one dobivene drugim grananjem, možemo vidjeti, da je prvo grananje bolje od drugog.

4.3.2.2. Računanje Gini indeksa

$$(plava) = \frac{broj\ plavih\ točaka}{broj\ plavih\ točaka + broj\ crvenih\ točaka} = \frac{5}{5+6} = \frac{5}{11} \approx 0.46$$

$$P(crvena) = \frac{broj\ crvenih\ točaka}{broj\ plavih\ točaka + broj\ crvenih\ točaka} = \frac{6}{5+6} = \frac{6}{11} \approx 0.54$$

1. Gini indeks trenutnog stanja:

$$G(S) = \sum_{i=1}^{d} q(i) \times (1 - q(i))$$

$$G(S) = \big(0.46 \times (1 - 0.46)\big) + \big(0.54 \times (1 - 0.54)\big) = 0.2484 + 0.2484 = \mathbf{0.4968}$$

2. Gini indeks nakon prvog grananja:

$$G(SL) = \left(\frac{5}{5} \times \left(1 - \frac{5}{5}\right)\right) + \left(\frac{0}{5} \times \left(1 - \frac{0}{5}\right)\right) = 0 + 0 = \mathbf{0}$$

$$G(SD) = \left(\frac{6}{6} \times \left(1 - \frac{6}{6}\right)\right) + \left(\frac{0}{6} \times \left(1 - \frac{0}{6}\right)\right) = 0 + 0 = \mathbf{0}$$

$$G(S) = \sum_{i=1}^{d} \frac{Sp}{S} \times Gi = \frac{pi + ni}{p + n} \times Gi$$

$$GNG(S) = \left(\frac{5}{11} \times 0\right) + \left(\frac{6}{11} \times 0\right) = \mathbf{0}$$

3. Gini dobit nakon prvog grananja:

$$Gini\ gain = G(S) - GNG(A)$$

$$Gini\ gain = \mathbf{0.4968} - \mathbf{0} = \mathbf{0.4968}$$

1. Gini indeks nakon drugog grananja:

$$G(SL) = \left(\frac{5}{6} \times \left(1 - \frac{5}{6}\right)\right) + \left(\frac{1}{6} \times \left(1 - \frac{1}{6}\right)\right) = \mathbf{0.27777}$$

$$G(SD) = \left(\frac{5}{5} \times \left(1 - \frac{5}{5}\right)\right) + \left(\frac{0}{5} \times \left(1 - \frac{0}{5}\right)\right) = 0 + 0 = \mathbf{0}$$

$$G(S) = \sum_{i=1}^{d} \frac{Sp}{S} \times Gi = \frac{pi + ni}{p + n} \times Gi$$

$$GNG(S) = \left(\frac{6}{11} \times 0.27777\right) + \left(\frac{5}{11} \times 0\right) = \mathbf{0.151511}$$

2. Gini dobit nakon drugog grananja:

$$Gini\ gain = G(S) - GNG(A)$$

$$Gini\ gain = \mathbf{0.4968 - 0.151511 = 0.3453}$$

Ukoliko usporedimo grafove i rezultate dobivene nakon prvog i drugog grananja, lako je ustanoviti, koje je bolje grananje. Jasno se vidi da je prvo grananje, puno bolje od drugog, pošto je gini indeks za dobivene podskupe jednak nuli, a gini dobit prvog grananja, je veći od onog, drugog grananja.

4.4. Obrezivanje i kriteriji zaustavljanja

Proces dijeljenje skupa podataka ili bolje rečeno grananje, traje tako dugo, dok sve grane stabla nisu čiste, odnosno, dok se u svakoj grani ne nalazi samo jedna vrsta primjera. Kada bi se grananje provodilo na ovaj način, kreiralo bi se stablo koje je preveliko, nepregledno i previše kompleksno. Također bi došlo do problema pretreniranosti (*eng. overfitting*-a), koji se ne javlja samo kod stabla odlučivanja, već i kod drugih metoda strojnog učenja.

Većina ljudi preferira doći do rezultata u što kraćem vremenu ili kroz što manji broj koraka, a to isto vrijedi i za stabla odlučivanja. Ljudi preferiraju jednostavnija stabla, jer su puno lakša za shvatiti, a kompleksnost stabla također utječe i na njegovu preciznost. Stabla koja su kompliciranа i imaju puno listova, imaju lošu sposobnost generalizacije. Iako ispravno klasificiraju sve instance iz skupa za treniranje, moguće da neće tako dobro funkcionirati na novim, dosad neviđenim slučajevima. Bolje je imati jednostavnije stablo koje će bolje generalizirati, nego kompleksnije, koje će imati savršene rezultate sa skupom za treniranje modela.

Neke od metrika koje se koriste za određivanje kompleksnosti su broj čvorova, broj, te broj korištenih atributa dubina stabla. Sama kompleksnost se regulira sa kriterijima zaustavljanja, te „obrezivanjem“ stabla. Postoje dva pristupa obrezivanju, a prvi je da se rast stabla ili bolje rečeno grananje, zaustavlja prije nego što dođemo do kraja procesa. Takav pristup koristi kriterije zaustavljanja, pomoću kojih se određuje kada je potrebno zaustaviti proces grananja, a to svrstavamo u grupu metoda obrezivanja unaprijed (*eng. pre pruning*). [14]

Česta pravila zaustavljanja: [4]

- Sve instance u skupu za treniranje pripadaju jednoj kategoriji y
- Dosegnuti je maksimalni nivo stabla
- Broj slučajeva u listu, je manji od minimalnog broja slučajeva, čvora roditelja
- Da je čvor rastavljen, broj slučajeva u jednom ili više čvorova koji su dijete, bio bi manji od minimalnog broja klasa slučajeva za čvorove, dijete
- Najbolji kriterij rastavljanja, nije veći od određene granice

Postoj nekoliko stvari za izbjegavanje problema pretreniranosti. Možemo postaviti minimalan broj instanci nekog čvora, koje su potrebne kako bi ga mogli dalje granati. Na taj način se sprječava, da model ne razvije, neka previše specifična pravila. Prilikom postavljanja minimalnog broja, treba imati na umu, da prevelika vrijednost ovog parametra, može pridonijeti pod-treniranosti. Također se može postaviti minimalni broj primjera u završnom čvoru (listu) i maksimalni broj završnih čvorova ili listova, koji se može definirati umjesto maksimalne dubine stabla. Može se definirati i maksimalni broj atributa (*eng. features*) koji se razmatraju za najbolje grananje. [19]

Drugi pristup obrezivanja je malo drugačiji, jer dozvoljavaju stablu da završi grananje, te da na taj način postane pre-trenirano. Tek nakon toga se kreće sa obrezivanjem stabla. Od originalnog skupa podataka se odvoji jedan skup koji će se neće koristiti za treniranje, nego samo za obrezivanje. Za svako odabrano pod-stablo se onda, na njegovo mjesto, postavi i testira list. Ako su performanse lista jednake ili bolje od pod-stabala, zadržavamo ga i obrezujemo, a u suprotnom, vraćamo pod-stablo. Takve metode spadaju u kategoriju naknadnog obrezivanja (*eng. post pruning*). Ovaj pristup se pokazao puno boljim, jer postoje poteškoće vezane uz predviđanja, kada prestati s grananjem. [8], [14]

4.5. Algoritmi

Prije smo spomenuli induktivno učenje, kod kojeg se, koristeći skup podataka, u kojem se nalazi velik broj primjera, prema kojima se onda uče i kreiraju pravila generalizacije. Na taj način možemo brže, jednostavnije i točnije naučiti pravila, koja se onda mogu primijeniti, na dosad neviđenim primjerima.

Algoritam za induciranje ili induktor (*eng. inducer*) je entitet, koji dohvaća skup podataka za treniranje, te iz njega kreira model koji generalizira odnose i veze, između ulaznih i izlaznih (ciljanih) atributa. Kao ulaz, uzima specifične podatke za trening, sa odgovarajućim oznakama i pomoću njih proizvodi model za klasificiranje (*eng. classifier*). Takav model može se koristiti za klasificiranje dosad neviđenih primjera, bilo da se eksplicitno pridruže nekoj klasi (*eng. crisp classifier*) ili koristeći vektor vjerojatnosti, koji predstavlja uvjetnu vjerojatnost dane instance da ona pripada klasi (vjerojatnosni klasifikator). [4]

Algoritmi za induciranje automatski konstruiraju stablo odlučivanja od danog skupa podataka, a cilj je pronaći optimalno stablo odlučivanja, sa najmanjom greškom generalizacije. Također je moguće ciljati na najmanji broja čvorova ili pak, na najmanju dubinu stabla. [4]

4.5.1. ID3

Većina algoritama koji su razvijeni za stabla odlučivanja su zapravo, samo varijacije jednog temeljnog algoritma ID3, koji koristi pohlepni TOP-DOWN pristup. Algoritam započinje odabirom najboljeg atributa za početno grananje. Da bi odabrao najbolji atribut, mora ih sve procijeniti statističkim testiranjem kako bi vidio kako dobro klasificira primjere za trening. Čvor dijete, se onda kreira, za svaku moguću vrijednost odabranog atributa, a primjeri za trening se sortiraju, prema primjerenom čvoru djetetu. Cijeli taj proces se onda ponavlja, za svaki čvor dijete. Na taj način se provodi ova pohlepna potraga za prihvatljivom stablom, u kojoj algoritam, nikad ne ide korak unatrag, kako bi razmotrio prethodno grananje. [14]

Ovo je jedan od jednostavnijih algoritama koji se koriste za stabla odlučivanja, koji kao kriterij grananja, koristi već prethodno spomenutu dobit informacija (*eng. infromation gain*), a grana stablo tako dugo, dok sve instance ne pripadaju jednoj vrijednosti ciljanog atributa ili dok najveća dobit informacija, nije veći od nula. [4]

Glavan prednost ID3 algoritma je njegova jednostavnost, no moglo bi se reći, da je to ujedno i njegov nedostatak. Ne garantira optimalno rješenje, te je moguće da će zaglaviti u lokalnim optimumima, jer koristi pohlepnu strategiju. Kako bi se to izbjeglo, moguće je vratiti se natrag, prilikom traženja. Često se zna desiti, da dođe do pretreniranosti (*eng. overfitting-a*), pa obično generira mala stabla, ali ne najmanja moguća. Također ne može raditi sa nedostajućim i numeričkim vrijednostima, pa je potrebno kontinuirane podatke pretvoriti u nominalne skupine. [4]

4.5.2. C4.5

Zbog nedostataka prethodno spomenutog algoritma, preferira se korištenje C4.5 algoritma. Radi se o rekurzivnom algoritmu, koji se koristi za generiranje stabla odlučivanja, koji je zapravo unaprijeđeni i prošireni ID3 (*eng. Iterative Dichotomiser 3*) algoritam. [20], [4]

Najznačajnija poboljšanja koja su vezana uz C4.5 su obrezivanje stabla, ili bolje rečeno, uklanjanje grana koje ne doprinose točnosti i njihova zamjena listovima. U stanju je raditi sa numeričkim podacima i sa skupom podataka, u kojem imamo nedostajuće vrijednosti atributa, te sa kontinuiranim vrijednostima, tako da ih razdvoji na dva podskupa, tražeći najbolju granicu, za najveći omjer dobiti. [4]

Iako radi na sličan načina kao i CART algoritam, provjeravajući svaki čvor u potrazi za onim koji je najbolji za grananje, tako dugo dok je grananje moguće, postoje neke razlike. Algoritam C4.5 nije ograničen na samo binarno grananje, već može proizvesti stablo sa čvorovima, koji se dijele na više dijelova. Za kategorijske atribute u pravilu, dijeli čvor, na toliko grana koliko odabrani atribut ima različitih kategorija. Osim ovih razlika, najveći je vjerojatno način, na koji C4.5 mjeri homogenost, za što koristi koncept smanjenja entropije, odnosno dobit informacija (*eng. information gain*). Za svako grananje, algoritam odabire ono kod koje ima najveću dobit informacija i taj postupak se nastavlja tako dugo dok je moguće grananje, a nakon toga, algoritam započinje sa obrezivanjem stabla. [20]

4.5.3. CART

Naziv algoritma CART je skraćenica za *Classification and Regression Trees*, a njegova glavna karakteristika je da kreira binarna stabla, gdje svaki čvor, ima točno 2 izlazna. Čvorovi se granaju koristeći Twoing kriterij, a dobiveno stablo se obrezuje koristeći tehniku *Cost-Complexity Pruning*. Pri induciranju stabla, može razmatrati trošak pogrešne klasifikacije, te dozvoliti korisniku da prije induciranja, priloži vjerojatnosnu distribuciju. Još jedna važna karakteristika CART algoritma je da je u stanju kreirati stabla regresije, kod kojih listovi predviđaju realni broj a ne klasu. [4]

4.6. Evaluacija stabla

Algoritam za induciranje stabla kao ulaz uzima skup za treniranje, odnosno učenje, te iz njega kreira stablo, koje može klasificirati dosad neviđene instance. Stablo klasificiranja, ali i sam algoritam koji se koristi za njegovo kreiranje, se mogu evaluirati, koristeći određene kriterije. Ta evaluacija je važna za razumijevanje stabla klasifikacije, ali i za podešavanje parametara u iterativnom procesu otkrivanja znanja u podacima, jer nam govori o efektivnosti dobivenog modela. Razvoj efikasnih indikatora za procjenu kvalitete rezultata analize, je važan problem, kod otkrivanja znanja u podacima, a evaluacija performansi stabla klasifikacije, je osnovni aspekt strojnog učenja. [4]

4.6.1. Matrica konfuzije

Radi se o mjeri performansi u strojnom učenju, te se koristi se kao indikator svojstava klasifikacijskog pravila i sadrži broj elemenata, koji su ispravno ili neispravno klasificirani, za svaku klasu. Velika prednost je ta, da se lako može vidjeti, ako sustav miješa dvije klase i krivo ih označava. Za svaku instancu u testnom skupu, uspoređuje njezinu stvarnu klasu kojoj pripada, sa onom, koju joj je zadao trenirani model.

Potrebno je razumijete četiri pojma koja se koriste kod matrica konfuzije, jer se koriste za popunjavanje matrice. Prvo imamo, istinito pozitivan ili TP (*eng. true positive*), kada se predvidi da je nešto pozitivno i to je uistinu tako. Istinito negativno ili TN (*eng. true negative*), kada se predvidi da je nešto negativno i to je istinito. Osim tih vrijednosti koje su istinite, također imamo i greške, odnosno lažno pozitivno ili FP (*eng. false positive*) kada se predvidi da je nešto istinito, a nije. Te lažno negativno ili FN (*eng. false negative*), kada se predvidi da je nešto negativno ,a nije.

Tablica 1 Matrica konfuzije [4]

	Predviđen negativno	**Predviđeno pozitivno**
Negativni primjeri	A / TN	B / FP
Pozitivni primjeri	C / FN	D / TP

Na temelju vrijednosti, u tablici konfuzije iznad, mogu se izračunati neke iznimno korisne mjere, koje pomažu u određivanju efikasnosti i kvalitete dobivenog modela.

Točnost, koja nam govori o tome, koliko je slučaja od svih predviđeno točno, odnosno, koliko je istinito negativnih i istinito pozitivnih, u odnosu na sve predviđene vrijednosti.

$$Točnost = \frac{(a+d)}{(a+b+c+d)}$$

Stopa pogrešne klasifikacije nam govori, koliko je od svih predviđanja, bilo netočnih.

$$Stopa\ pogrešne\ klasifikacije = \frac{(b+c)}{(a+b+c+d)}$$

Preciznost nam govori o tome, koliko je od svih pozitivno predviđenih primjera, zapravo bilo ispravnih.

$$Preciznost = \frac{d}{(b+d)}$$

Osjetljivost (*eng. sensitivity*) je također poznato kao opoziv (*eng. recall*), a govori o tome koliko je od svih pozitivno predviđenih primjera, zapravo predviđeno ispravno, te je mjera, koja bi trebala biti što veća. Daje omjer istinito pozitivnih uzoraka, koji su stvarno pozitivni i sveukupnog broja, koji od uzorka su procijenjeni kao pozitivni. [4]

$$Opoziv = \frac{d}{(c+d)}$$

Lažna pozitivna stopa, suprotno od opoziva, nam govori koliko dobro model prepoznaje negativne primjere ili bolje rečeno, koliko je od negativnih primjera bilo onih, koji su predviđeni kao pozitivni.

$$Lažna\ pozitivna\ stopa = \frac{b}{(a+b)}$$

Istinita negativna stopa nam govori, koliko je negativnih primjera predviđeno ispravno, negativno.

$$Istinita\ negativna\ stopa = \frac{a}{(a+b)}$$

Lažna negativna stopa nam govori, koliko je od ukupnog broja pozitivnih primjera, predviđeno krivo.

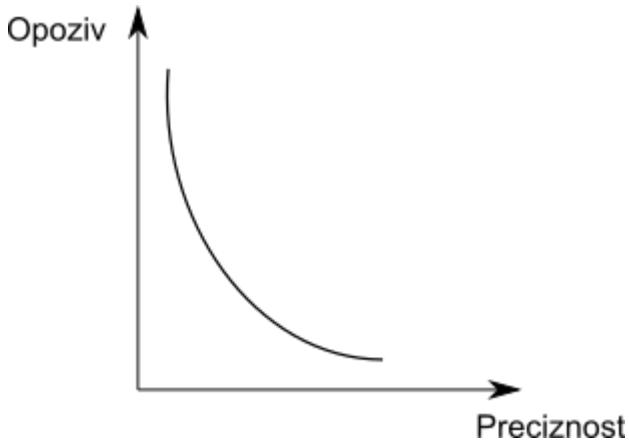
$$Lažna\ negativna\ stopa = \frac{c}{(c+d)}$$

4.6.2. F-mjera

U praksi nas često zanimaju brojke, koje su vezane uz lažne pozitivne i lažne negativne pojave. Razlog tome je taj da ovisno o situaciji ili problemu koji rješavamo, koristeći generirani model, mogu nastati veliki troškovi ili pak teške posljedice.

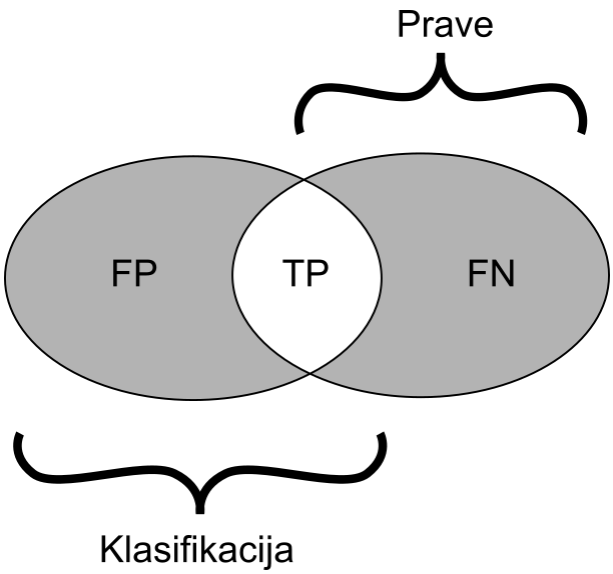
Prije smo spomenuli dvije mjere, koje se koriste za procjenu kvalitete nekog dobivenog modela, koji će se koristiti za klasifikaciju. Radi se o mjerama preciznost i opoziv (*eng. recall*), koje ovisno o važnosti problema koji želimo riješiti uz pomoć strojnog učenja, mogu imati i igrati veliku ulogu, da li ćemo reći da li je neki model dobar ili loš.

Obično postoji određena razmjena između mjera, jer nije moguće imati model koji ima visok opoziv i preciznost. Kada pokušamo poboljšati jednu mjeru, druga će se kao rezultat toga, pogoršati. Stoga je potrebno pronaći, neku određenu ravnotežu, između tih dviju mjera. [4]



Slika 7 Dijagram preciznosti i opoziva [4]

Obzirom da aritmetička sredina može navoditi na krive zaključke, najbolje je koristiti harmonijsku sredinu ,koja daje bolju ideju o „prosjeku“. Ta mjera nam pomaže, istovremeno usporediti opoziv i preciznost, dvaju modela. [4]



Slika 8 Grafički prikaz F mjere [4]

F-mjera se može intuitivno objasniti pomoću gornje ilustracije gdje desna elipsa, predstavlja sve lažno negativne skupove, lijeva sve lažno pozitivne, a mjesto presjeka tih dviju elipsa, istinite pozitivne vrijednosti (TP). Intuitivan način mjerenja adekvatnosti nekog modela klasifikacije je da se izmjeri u kojoj mjeri se ta dva skupa preklapaju, odnosno, kolika je veličina neosjenčane površine. Pošto apsolutna veličina nije značajna, trebalo bi ju normalizirati, tako da se izračuna proporcionalna površina. [4]

$$F = Omjer\ neosjenčane\ površine$$

$$F = \frac{2 \times (True\ Positive)}{False\ Positive + False\ Negative + 2 \times (True\ Positive)}$$

$$F = \frac{2 \times Preciznost \times \text{Opoziv}}{Preczinost + \text{Opoziv}}$$

F mjera može imati vrijednosti u rasponu od 0 do 1. Ima najveću vrijednost kada se dva skupa preklapaju, a najnižu kada su međusobno razdvojeni. Svaka točka na dijagramu preciznosti i opoziva može imati drugačiju F-mjeru, a drugačiji klasifikatori, drugačije grafove preciznosti i opoziva. [4]

5. KDD proces nad skupom podataka

Prije smo opisali proces otkrivanja znanja u podacima (KDD), koji se provodi kroz nekoliko koraka, koji su svi iterativni. Također je bilo napomenuto, da je važno razumijevanje procesa i njegovih koraka, ali i same problemske domene i cilja koji želimo postići. Taj isti proces ćemo primijeniti i demonstrirati, u nastavku.

5.1. Uvod u praktični dio

U prvom dijelu rada upoznali smo se s teorijom strojnog učenja, a u slijedećem dijelu, prelazimo na praktični dio, u kojem ćemo izraditi stablo odlučivanja, koje će služiti kao temelj, za izradu inteligentnog sustava. Radi se o nadzornom učenju, pošto je za svaku instancu obređena klasa kojoj pripada, te će cilj zadatka biti predviđanje, a zadatak rudarenja će biti klasifikacija, koristeći metodu stabla odlučivanja.

Za izradu samog stabla ćemo koristiti, javno dostupni skup podataka u odabranoj domeni, koji ćemo prvo pred procesirati i pripremiti, a onda kasnije, obrađivati u popularnom programu za strojno učenje.

5.2. WEKA

Za praktični dio ovog rada će se koristiti program zvan „*Weka*“. Radi se o isprobanom i testiranom softwareu, otvorenog tipa, namijenjenom za strojno učenje. Može se koristiti putem grafičkog korisničkog sučelja, standardnih terminalskih aplikacija ili Java API-ja. Široko se koristi za učenje, istraživanje, te čak, ima primjenu i industriji. Osim što sadrži puno raznih ugrađenih alata za standardne zadatke strojnog učenja, relativno je jednostavan za korištenje i najbolje je to, što je besplatan. [22]

Slika 9 Logo programskog alata Weka [22]

5.3. Odabir i prikupljanje podataka

Iako je ovo zapravo drugi korak koji se provodi, nakon upoznavanja sa domenom i određivanjem ciljeva, pošto se ovdje radi o sekundarnim podacima, prvog ćemo ga obraditi. Podaci koji će se obrađivati, vezani su uz karakteristike gljiva, koje je potrebno znati, kako bismo mogli razlikovati otrovne od jestivih.

Skup podataka koji je odabran za izradu stabla odlučivanja, čija će svrha biti klasifikacija, preuzet je sa stranice „*Kaggle*“, koja je podružnica tvrtke Google LLC. Radi se o internetskoj zajednici znanstvenika, koji se bave podacima i strojnim učenjem. Sam skup podataka je dodan UCI repozitoriju strojnog učenja, još prije nekih tridesetak godina. U skupu se nalaze opisi hipotetičkih uzoraka, koji odgovaraju 23 vrste gljiva Agaricus i Lepiota. Vrste su izvučene iz terenskog vodiča Društva Audubon za gljive Sjeverne Amerike (1981), u kojem je također navedeno, da ne postoji jednostavno pravilo za utvrđivanje jestivosti gljive. Svaka vrsta spada u jednu od tri klasa; jestiva, definitivno otrovna ili nepoznata jestivost i ne preporučuje se. Posljednja klasa je spojena s otrovnom klasom. [23]

Izvorni skup podataka u sebi sadrži 8124 instanci gljiva, odnosno primjera, koji su svi opisani, koristeći 23 atributa. U skupu atributa također je jedan, koji sadrži oznaku, odnosno klase, da li je gljiva o kojoj se radi, jestiva ili ne. U sljedećoj tablici, nalazi se popis atributa i njihovih klasa.

Tablica 2 Atributi i klase skupa podataka dio 1 od 3 [23]

Redni broj	*Attributes*	*Classes*	Atribut	Klase
1	class	edible=e poisonous=p	klasa	jestiva = e otrovna = p
2	cap-shape	bell=b conical=c convex=x flat=f knobbed=k sunken=s	Oblik šešira	zvono = b konusni = c konveksan = x ravni = f ispupčen = k upušten = s
3	cap-surface	fibrous=f grooves=g scaly=y smooth=s	Površina šešira	vlaknasta = f žljebovi = g ljuskava = y glatka = s
4	cap-color	brown=n buff=b cinnamon=c gray=g green=r pink=p purple=u red=e white=w yellow=y	Boja šešira	smeđa = n boja kože = b boja cimeta = c siva=g zelena = r ružičasta = p ljubičasta = u crvena=e bijela = w žuta = y
5	bruises	bruises=t no=f	Može dobit modricu	da = t ne = f
6	odor	almond=a anise=l creosote=c fishy=y foul=f musty=m none=n pungent=p spicy=s	Miris	badem = a anis=l čađa / ugljik = c sličan ribi = y truli=f ustajao = m nijedan = n oštar = p pikantan = S
7	gill-attachment	attached=a descending=d free=f notched=n	Privitak Listića	vezani = a silazno=d slobodna = f urezani = n
8	gill-spacing	close=c crowded=w distant=d	Razmak između listića	blizu = C natiskan = w udaljen = d

Tablica 3 Atributi i klase skupa podataka dio 2 od 3 [23]

Redni broj	*Attributes*	*Classes*	Atribut	Klase
9	gill-size	broad=b narrow=n	Veličina listića	široko = b sužavanje = n
10	gill-color	black=k brown=n buff=b chocolate=h gray=g green=r orange=o pink=p purple=u red=e white=w yellow=y	Boja listića	crna = k smeđa = n boja kože = b čokoladna = h siva = g zelena = r narančasta = o ružičasta = p ljubičasta = u crvena = e bijela = w žuta = y
11	stalk-shape	enlarging=e tapering=t	Oblik stabljike	širenje = e se sužava = t
12	stalk-root	bulbous=b club=c cup=u equal=e rhizomorphs=z rooted=r missing=?	Stabljika-korijen	gomoljasta = b klupčasta = c šalica = u jednaka = e korjenasta = z ukorijenjena = r nedostaje = ?
13	stalk-surface-above-ring	fibrous=f scaly=y silky=k smooth=s	Površina stabljike iznad prstena	vlaknasta = f ljuskava = y svilena = k glatka = s
14	stalk-surface-below-ring	fibrous=f scaly=y silky=k smooth=s	Površina stabljike ispod prstena	vlaknasta = f ljuskava = y svilenkasta = k glatka = s
15	stalk-color-above-ring	brown=n buff=b cinnamon=c gray=g orange=o pink=p red=e white=w yellow=y	Boja stabljike iznad prstena	smeđa = n boja kože = b boja cimet = c siva=g narančasta = o ružičasta = p crvena=e bijela = w žuta = y
16	stalk-color-below-ring	brown=n buff=b cinnamon=c gray=g orange=o pink=p red=e white=w yellow=y	Boja stabljike iznad prstena	smeđa = n boja kože = b boja cimeta = c siva = g narančasta = o ružičasta = p crvena = e bijela = w žuta = y

Tablica 4 Atributi i klase skupa podataka dio 3 od 3 [23]

Redni broj	*Attributes*	*Classes*	Atribut	Klase
17	veil-type	partial=p universal=u	Veo-vrsta	djelomični = p univerzalni = u
18	veil-color	brown=n orange=o white=w yellow=y	Veo-boja	smeđa = n narančasta = o bijela = w žuta = y
19	ring-number	none=n one=o two=t	Broj-prstena	nijedan = n jedan = o dva = t
20	ring-type	cobwebby=c evanescent=e flaring=f large=l none=n pendant=p sheathing=s zone=z	Vrsta prstena	paučinast = c prolazan = e postepeno širi = f veliki = l nijedan = n viseći = p obloga = s zona = z
21	spore-print-color	black=k brown=n buff=b chocolate=h green=r orange=o purple=u white=w yellow=y	Boja otiska spora	crna = k smeđa = n boja kože = b čokoladna = h zelena = r narančasta = o ljubičasta = u bijela = w žuta = y
22	population	abundant=a clustered=c numerous=n scattered=s several=v solitary=y	Populacija	obilna = a klasteri = c brojna = n raspršene = poštom više = v usamljena = y
23	habitat	grasses=g leaves=l meadows=m paths=p urban=u waste=w woods=d	Stanište	trave = g listovi = l livade = m staze = p urbana područja = u otpada = w šuma = d

5.4. Problemska domena i određivanje cilja

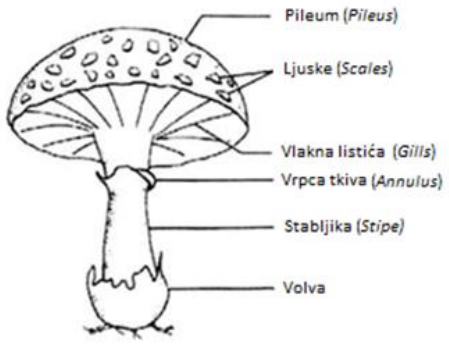
Ovo je zapravo prvi korak koji se odnosi na razumijevanje domene i određivanje cilja. Kad god se obrađuju podaci vezani uz bilo koju domenu, važno je da osoba koja ih obrađuje, ima barem neka temeljna znanja i razumijevanje. Ukoliko to nije slučaj, bilo bi dobro da ima na raspolaganju nekog, tko je stručan u tom području i tko će ju savjetovati, prilikom obrađivanja podataka.

Pošto se ovaj skup podataka obrađuje kako bi se bolje mogao shvatiti teorijski dio rada, detaljno poznavanje problemske domene nije potrebno. Unatoč tome, odlučio sam se malo bolje informirati, barem o onim pojmovima, koje predstavljaju atributi, pomoću kojih su opisane instance u skupu podataka.

Smatram da ću na taj način biti u boljem stanju evaluirati, te proizvesti bolji i kvalitetniji model stabla odlučivanja, koje će se koristi za klasifikaciju gljiva u skupine jestivih i otrovnih gljiva, na temelju njihovih karakteristika.

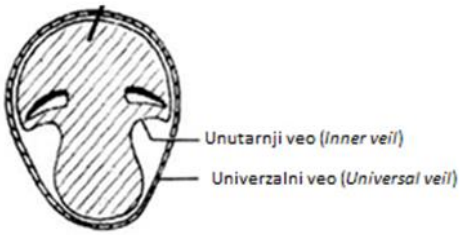
5.4.1. Upoznavanje s problemskom domenom

Gljive su jedan od najrasprostranjenijih organizama koji postoji, te ima puno različitih vrsta gljiva. Kao i većinu stvari u prirodu, često ih klasificiramo prema tome, da li su za nas, na bilo kakav način korisne ili ne, a najčešće prema tome da li su jestive ili nejestive, te da li su ljekovite ili otrovne. Gljiva se sastoji od puno različitih dijelova, čije karakteristike mogu poslužiti kako bismo ih lakše raspoznali, a na slici ispod, možete vidjeti neke od dijelova, koji su opisani u skupu podataka.

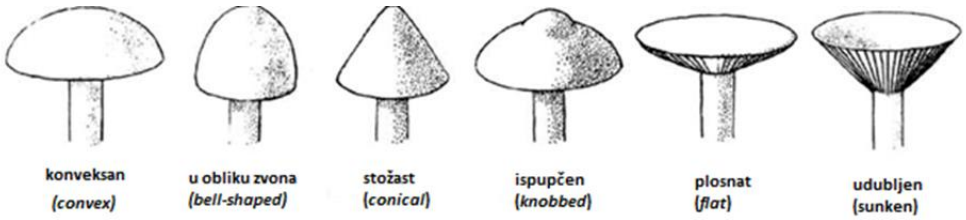


Slika 10 Glavni dijelovi gljive [24]

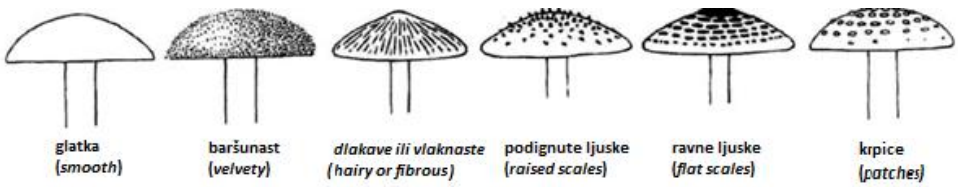
Pileum gljive je površina, koja pokriva gornji prošireni dio šešir gljive (cap) i koji djeluje kao zaštita njenog unutarnjeg sustava za proizvodnju spora. Zatim imamo donji dio šešira (himenij), čija je svrha stvaranje i izbacivanje spora u okolinu. Donji dio šešira je najčešće sačinjen od niza vlakna (*eng.* gills), koja nalikuju na tanke listiće i može biti spojen sa stabljikom na nekoliko različitih načina. Od šešira prema tlu, imamo stabljiku (*eng.* stipe), čija struktura može poduprijeti promjer šešira, a okružuje ju vrpca tkiva (*eng.* annulus). Pri samom dnu gljive imamo dio, zvan volva, koja je ovalnog oblika i omogućuje razvoj gljive. Volva je dosta važna, jer nam može pomoći ne samo identificirati vrstu, već i radi li se o jestivoj ili otrovnoj vrsti. Radi se zapravo o dijelu univerzalnog vela, koji pokriva gljivice tijekom rasta i ima funkciju lagane zaštite. [25]



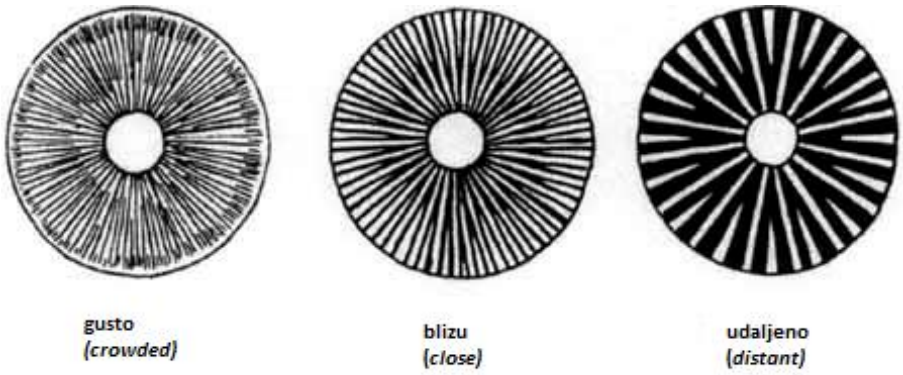
Slika 11 Presjek mlade gljive [24]



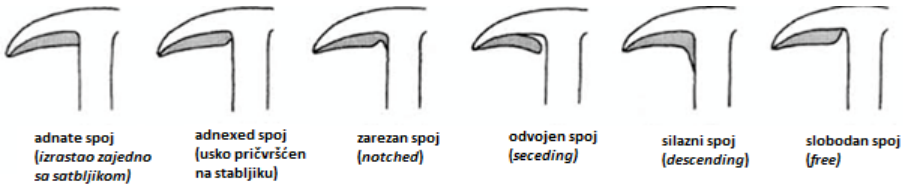
Slika 12 Oblici šešira gljiva [24]



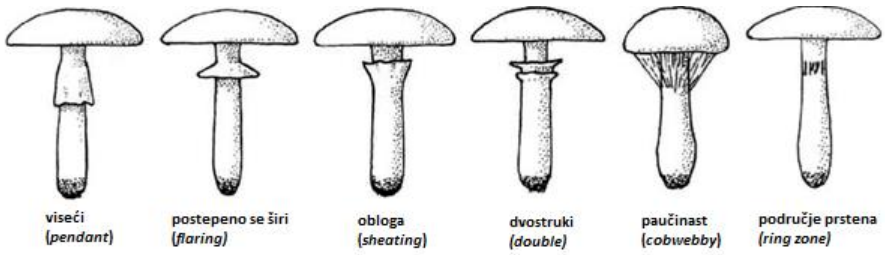
Slika 13 Površine šešira [24]



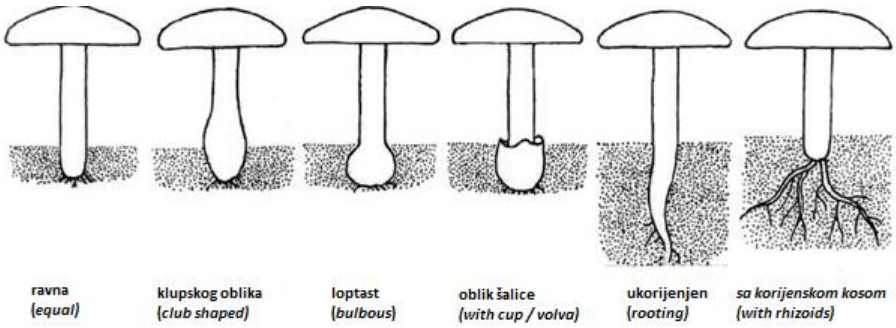
Slika 14 Razmaci vlakna [24]



Slika 15 Vrste spojeva na stabljiku [24]



Slika 16 Vrste prstena gljiva [24]



Slika 17 Vrste stabljika [24]

5.5. Pred procesiranje podataka

Iako Weka uglavnom koristi ARFF (*eng. attribute-relation file format*), također je u stanju raditi sa drugim vrstama datoteka. Jedna od tih vrsta je popularni CSV format (*eng. comma separeted value*), u kojem je pohranjen skup podataka kojeg ćemo obrađivati.

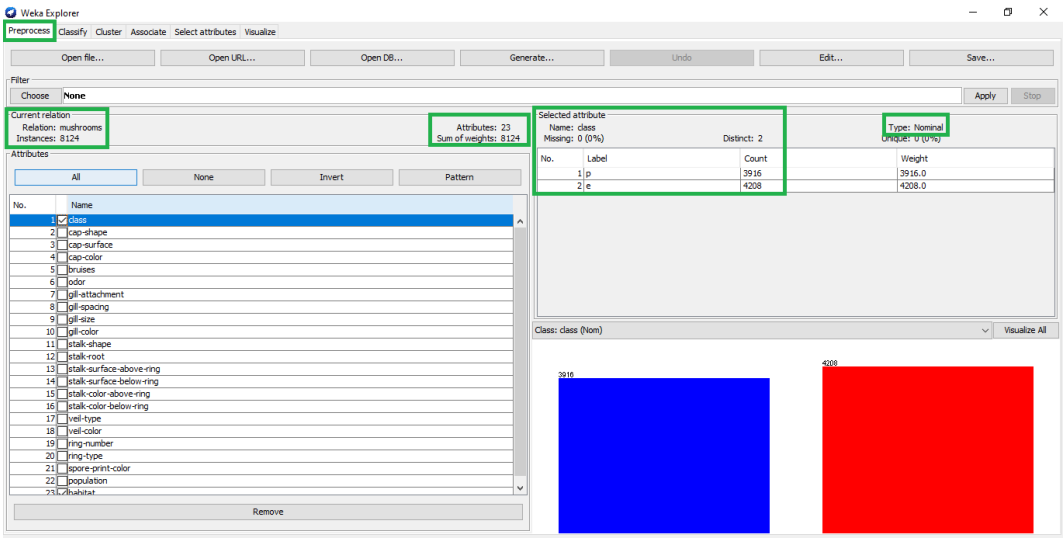
Iako je moguće naići na skup podataka koji je kvalitetno obrađen i kod kojeg su podaci odmah spremni za korištenje, u praksi je uglavnom, suprotna situacija. Upravo zbog toga je uvijek potrebno, pregledati svaki skup podataka, kako bismo bili sigurni da imamo kvalitetne podatke, koje možemo dalje obrađivati.

Neki od problema koji se često znaju desiti je da imamo, zapise koji sadrže nedostajuće vrijednosti ili da imaju krive vrijednosti, koje su van mogućih vrijednosti. Također, može biti da se u skupu podataka nalaze kopije, odnosno više istih zapisa, što također može utjecati na rezultat zbog broja javljanja zapisa s istim vrijednostima. Postoje mnogi razlozi zašto pojedine vrijednosti podataka nedostaju, moguće da je osoba koja je bilježila podatke napravila grešku, da ispitanik nije znao ili nije htio dati podatak, da je skup podataka spojen sa starijim, kod kojeg se neki podaci nisu uopće uzimali u obzir, možda je došlo do kvara na mjernom uređaju ili nešto sasvim drugo.

Također postoji i više načina kako postupiti sa tim nedostajućim vrijednostima. Najčešće se brišu, ukoliko se radi o nekom manjem broju zapisa, a također se može dodati kao zasebna klasa, može se staviti najčešća ili pak nasumična vrijednost i tako dalje. Najbolju odluku u vezi toga, što učiniti sa takvim podacima, može dati osoba koja je stručnjak u toj domeni.

5.5.1. Pregled i vizualizacija podataka

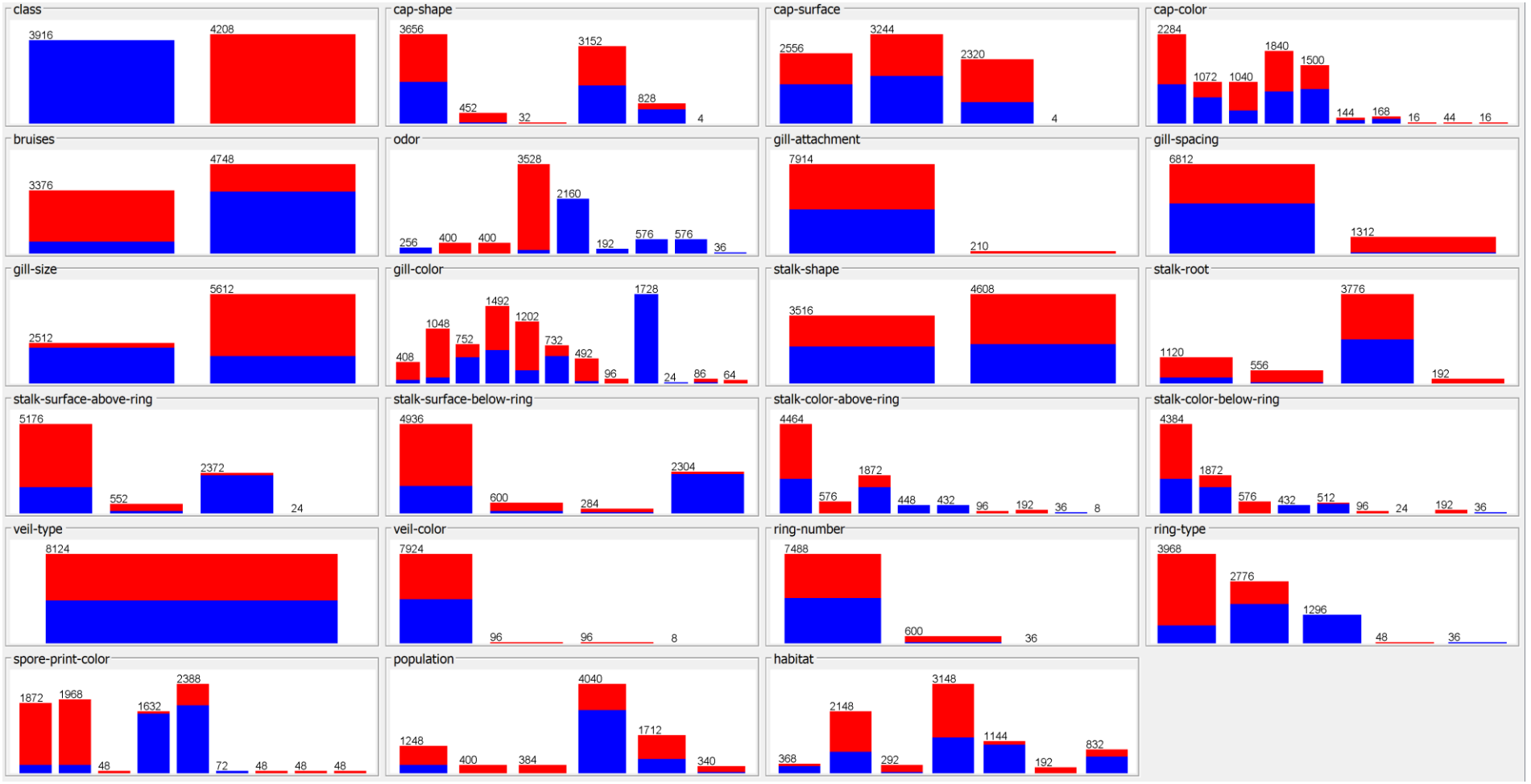
Prva stvar koju ćemo učiniti, zbog prethodno navedenih razloga, je otvoriti naš skup podataka i pregledati zapise. Tražit ćemo nepravilnosti u skupu podataka, kao što su negativne vrijednosti, gdje su zapravo dozvoljene samo pozitivne i slično. Također ćemo sagledati nepostojeće vrijednosti i druge stvari koje bi mogle, praviti probleme dalje, u budućnosti.



Slika 18 Prikaz podataka učitanog skupa

Kao što možete vidjeti na slici iznad, nalazimo se u prozoru za, pred procesiranje u „*Weka Exploreru*“, u kojem smo otvorili naš skup podataka. Zelenim okvirima su označene osnovne informacije, vezane uz skup podataka. Prve stvari koje se odmah mogu uočiti su, broj instanci, odnosno primjera, kojih ima 8124, te broj atributa, kojih ima 23. Trenutno je označen atribut sa klasama koje pokušavamo predvidjeti, odnosno, radi li se o gljivi koja je jestiva ili otrovna. Možemo vidjeti da imamo 3916 otrovnih i 4208 jestivih gljiva.

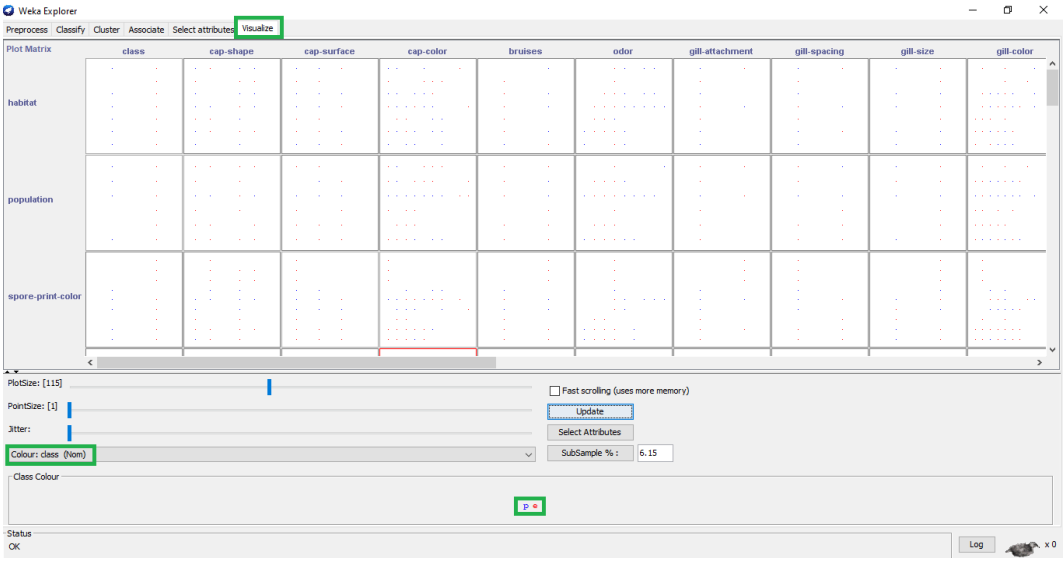
Na slici također možete primijetiti stupčasti, dijagram u kojem su prikazana dva stupca. Crvenom bojom su prikazane jestive gljive, a plavom otrovne. Ovdje je trenutno prikazan samo atribut sa te dvije klase, no također, možemo vidjeti i distribucije klasa drugih atributa, što je korisna funkcionalnost. Samo je potrebno kliknuti na gumb „*Visualize All*“ i otvorit će se novi prozor, sa dijagramima svih atributa.



Slika 19 Dijagrami distribucija klasa svih atributa skupa podataka

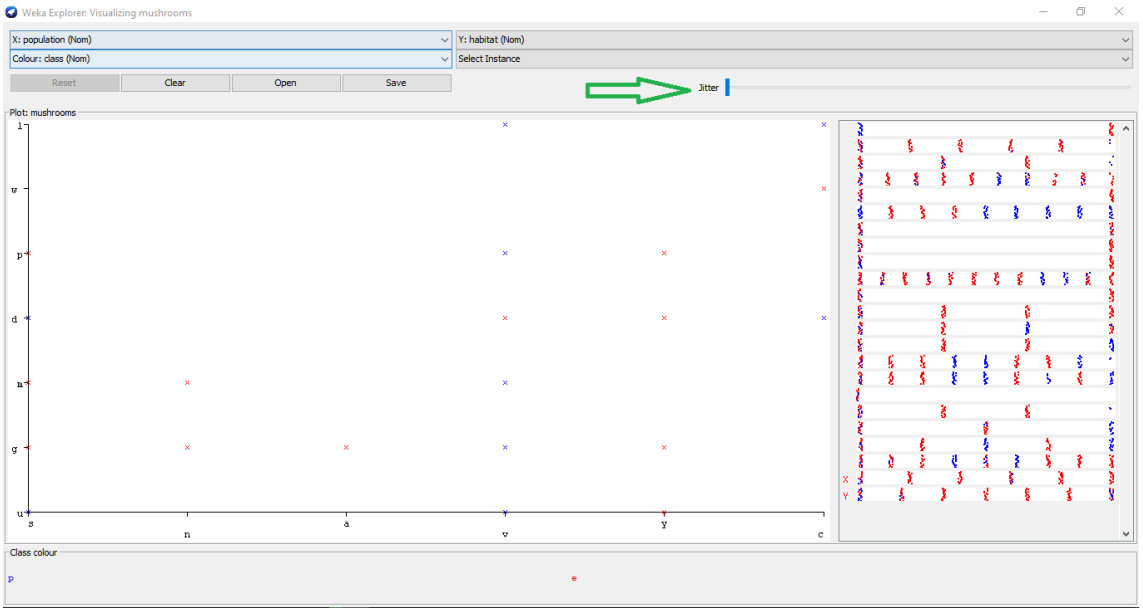
Weka također omogućuje vizualizaciju podataka, gdje međusobno uspoređuje atribute skupa podataka. Samo je potrebno kliknuti na gumb „*Visualize*“ i otvorit će se novi prozor, u kojem su prikazani plot dijagrami svih parova atributa, koje možete pobliže pregledati i još puno više.

Na slijedećoj slici možete vidjeti sve grafove koji su generirani za naš skup podataka. Na slici je posebno označena klasa, koja se koristi za bojanje vrijednosti i oznake.

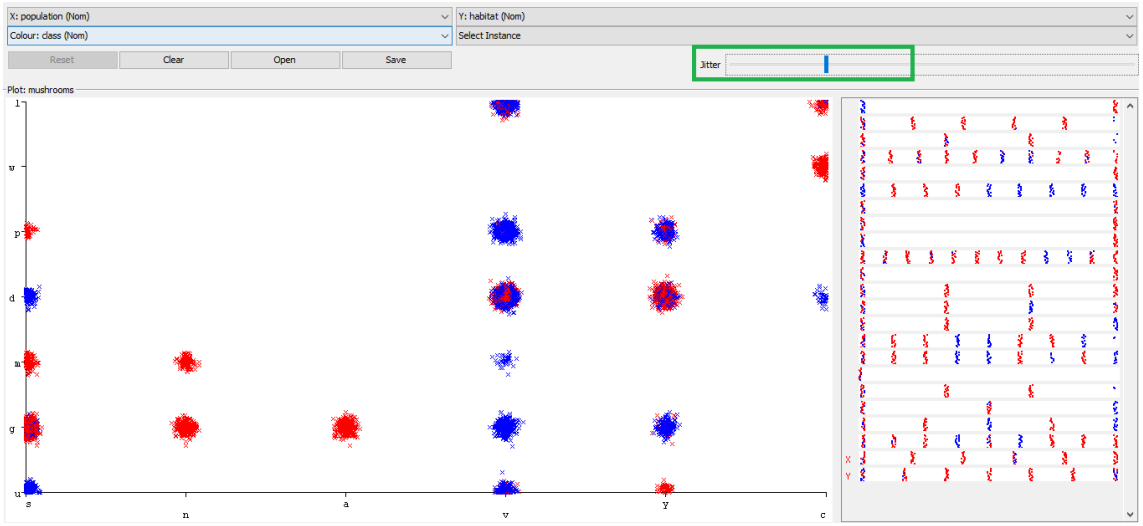


Slika 20 Matrica svih dijagrama uspoređenih atributa

Ako sada kliknete na neki od prikazanih grafova, otvorit će se novi prozor gdje će biti prikazana usporedba dvaju atributa i njihove klase na x i y osima. Ponekad se može činiti da ima malo instanci, ali zapravo su vrijednosti pridružene samo toj klasi, pa su na istoj točki na grafu. U takvim slučajevima možemo malo pomaknuti „*Jitter*“ kontroler, kako bismo utjecali na osi i dobili bolju sliku.

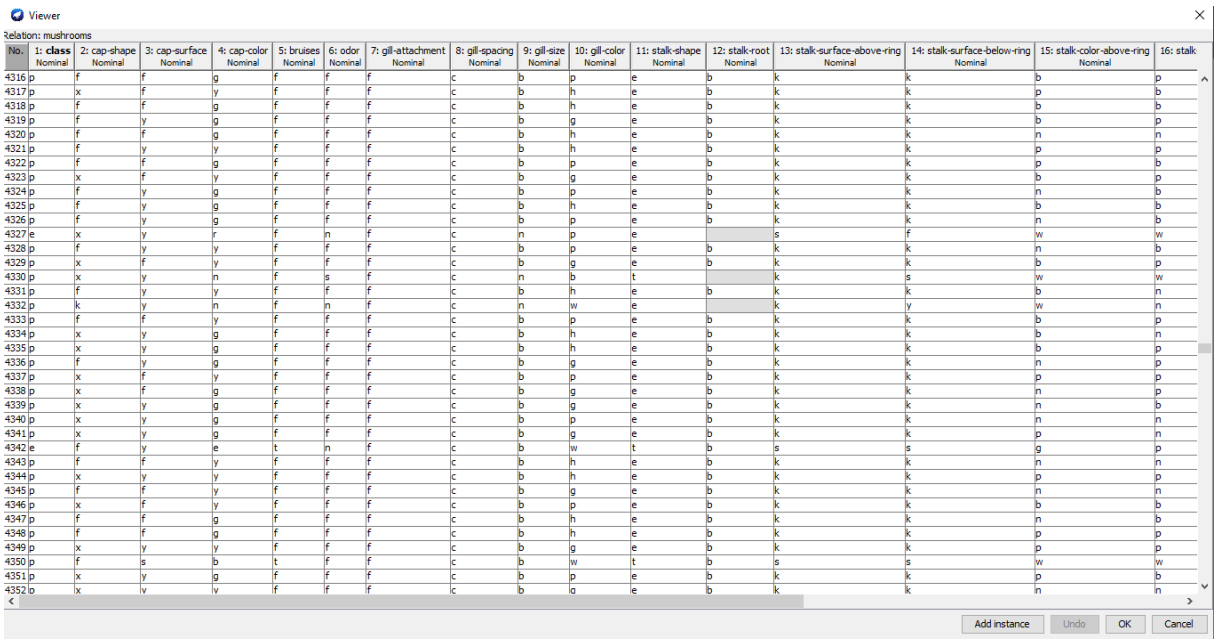


Slika 21 Dijagram prije korištenja kontrolera "Jitter"



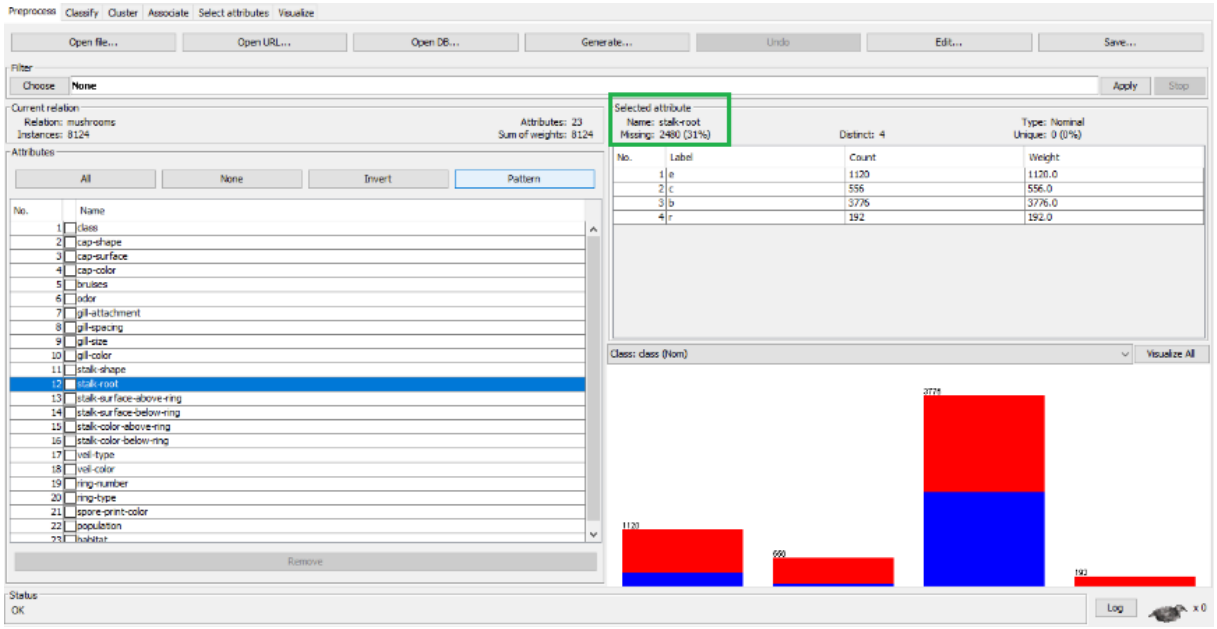
Slika 22 Dijagram nakon korištenja kontrolera "Jitter"

Ukoliko na početnom prozoru, za pred procesiranje, kliknete na gumb „*Edit*“ , otvara prozor prikazan na slijedećoj slici. Prikazuju se podaci u poljima, slično kao u programu „*MS Excel*“, te ih je također moguće mijenjati. Ako malo pobliže pogledate sliku, vidjet ćete, da je automatski određen tip svakog atributa. Druga stvar koju je moguće zamijetiti, su 3 prazne kučice u stupcu atributa „*stalk-root*“ , koje znače da za tu instancu, nema vrijednosti za taj atribut.



Slika 23 Pregled podataka u Weka Viewer prozoru

Kako ne bismo morali za svaki atribut gledati gdje nemamo zapisane vrijednosti za neku instancu, možemo na brzinu proći kroz atribute i pogledati informacije koje nam program prikazuje za svaki od njih.



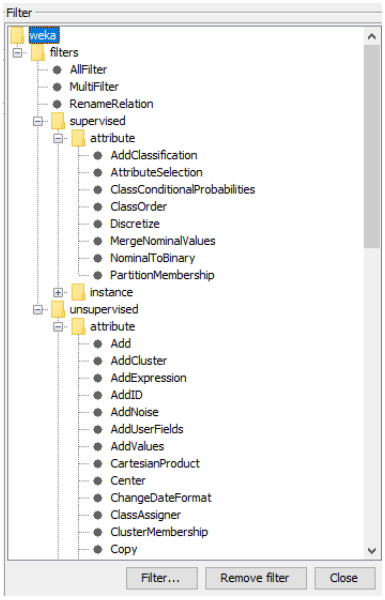
Slika 24 Prikaz atributa s nedostajućim vrijednostima atributa

Nakon pregledavanja svih atributa, jedini koji je imao nedostajuće vrijednosti je bio atribut „*stalk-root*“, sa 2480 nedostajućih vrijednosti. Kao što je prije bilo napomenuto, postoji više načina rješavanja ovog problema, no ponekad to nije ni potrebno, ovisno o algoritmu strojnog učenja, koji se koristi.

5.6. Čišćenje, reduciranje i transformacija podataka

Pošto u ovome skupu nemamo numeričkih podataka, ne moramo brinuti o stvarima kao što su skaliranje, korištenjem metode standardizacija ili recimo normalizacije podataka, te o čudnim vrijednostima ili njihovo pretvaranje u iste mjerne jedinice.

Program koji koristimo, ima velik broj filtera koji omogućuju brzu obradu i time transformaciju podataka, na različite načine. Na slici ispod su prikazani filtri programa, koji se dijele na dvije kategorije, nadzirni (*eng. supervised)* i ne-nadzirni (*eng. unsupervise)*“ , koje se ponovo dijele, na one koji rade s atributima ili samo sa instancama. Pri korištenju nadzirnih filtera, treba biti oprezan, kako bi se osiguralo da se rezultati ocjenjuju pošteno, jer oni koriste vrijednosti klase, a to je problem koji se ne pojavljuje kod nenadziranih filtara.



Slika 25 Popis filtera

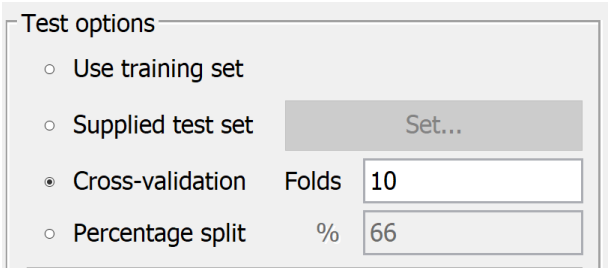
Obzirom na nedostajuće podatke u stupcu atributa, mogli smo jednostavno izbrisati sve zapise kod kojih vrijednosti nedostaju, pošto se kod ovako velikog skupa, radi o relativno malom broju. Također smo mogli izbrisati cijeli stupac i time reducirati broj dimenzija, ili pak naći neki drugi atribut, koji ima korelaciju sa ovim i koristit ga, za predviđanje nedostajućih vrijednosti. Ali obzirom na manjak znanja o domeni i činjenicu da je skup podataka već prethodno dobro pripremljen, odlučio sam ostaviti podatke takve kakvi jesu.

5.7. Odvajanje podataka za algoritam

U ovom dijelu odvajamo određeni dio podataka, kako bi ga kasnije mogli koristiti za testiranje modela. Ovaj korak je iznimno važan, jer osigurava da podaci ili bolje rečeno instance, koje će se koristiti za testiranje, nisu sudjelovale u kreiranju i treniranju modela. Uobičajena praksa je da se uzima veći dio podataka (oko 70% ili više) za treniranje modela, a neki manji dio za testiranje.

5.7.1. Opcije testiranja

Weka ima nekoliko opcija, točnije rečeno četiri opcije, koje se mogu koristiti za testiranje dobivenog modela i koje su prikazane na slici ispod.



Slika 26 Opcije testiranja modela

Prva opcija je da se za testiranje dobivenog modela koristi skup podataka koji je korišten za treniranje, odnosno za generiranje modela. To nije dobra ideja, jer će rezultati evaluacije modela biti previše optimistični ili bolje rečeno, točnost modela će biti puno veća, nego što zapravo je.

Slijedeća opcija je, zadavanje izdvojenog skupa podataka koji smo odvojili od originalnog i koji ćemo koristiti za testiranje. Prilikom postavljanja postavki testiranja, se jednostavno zada datoteka, koja sadrži skup podataka za testiranje i koja će se koristiti za evaluaciju modela.

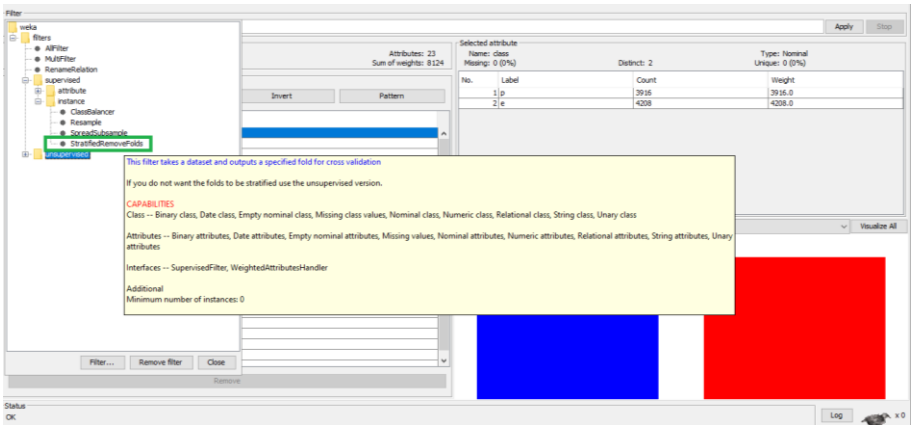
Predzadnja opcija koju možemo odabrati je, unakrsna validacija (*eng. Cross-validation)*. To je metoda koja podijeli skup na n dijelova (u ovom slučaju 10) i ponavlja se n puta. Prvih n dijelova se koristi za treniranje, a zadnji dio se koristi za testiranje. Tako podijeljeni dijelovi skupa se onda dalje koriste, ali se svaki put koristi neki drugi, dosad nekorišteni dio, za testiranje. Na kraju se uzima prosjek svih 10 testiranja kako bi se dobio rezultat, a na kraju program provodi algoritam za učenje još jedan put, kako bi se dobio model koji možemo koristiti.

Najniža metoda testiranja je „*Percentage split*“, odnosno razdvajanje skupa na dva dijela, prema postotku dijeli skup na dva dijela. Prvi dio se koristi za treniranje modela, a ostatak se koristi za testiranje, slično kao i prethodno opisana metoda, gdje zadajemo datoteku u kojoj se nalazi skup podataka za testiranje modela.

Ukoliko imamo puno podataka, kao što je to slučaj ovdje, može se koristiti „*Percentage-split*“ ili pak odvajanje podataka u drugu datoteku, koju možemo koristiti za testiranje. U suprotnom bi bilo bolje koristiti, unakrsnu validaciju, kako bi se dobila bolja procjena. Sve ovisi o količini podataka, broju atributa i broju klasa koje se nalaze u skupu podataka, koji se koristi za strojno učenje.

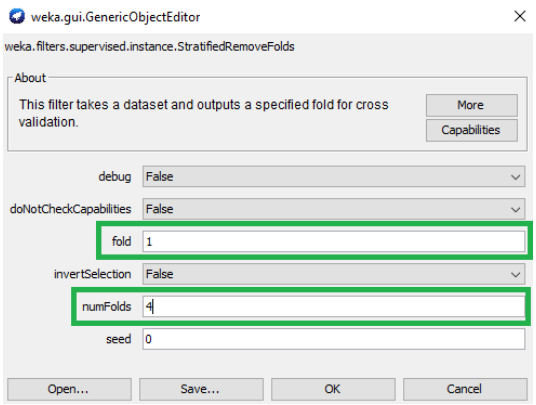
5.7.2. Podjela skupa podataka

Prilikom provođenja algoritma i testiranja modela kojeg ćemo kreirati koristit ćemo metodu sa zasebnom datotekom, koja sadrži skup podataka za testiranje. Prva stvar koju je potrebno napraviti kako bismo „fizički“ razdvojili zapise je, učitati naš skup podataka, u program. Nakon toga, treba odabrati jedan ili više filtera, koji će učiniti to što želimo. U ovome slučaju ćemo koristiti „*StratifiedRemoveFolds*“ , iz grupe nadzirnih filtera. Odabrali smo ovaj, jer dijeli skup na n dijelova i pritom u obzir uzima distribuciju klasa. To je važno, kako ne bismo dobili skup u kojem se nalaze samo otrovne ili samo jestive gljive.



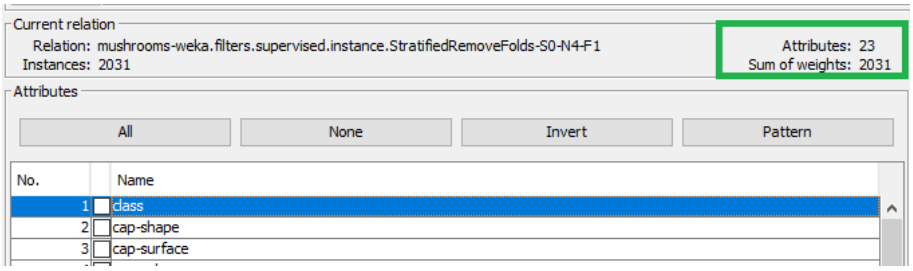
Slika 27 „StratifiedRemoveFolds“ filter i njegov opis

Slijedeći korak je podešavanje postavki koje su prikazane na slijedećoj slici.



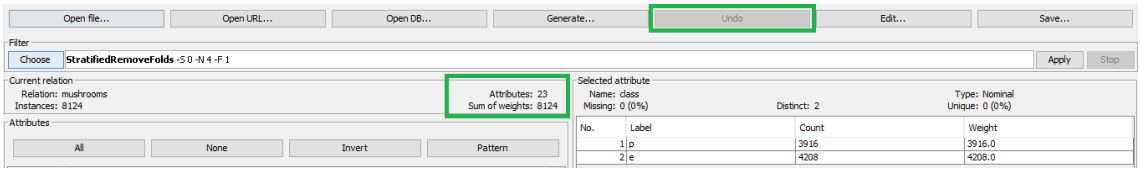
Slika 28 Postavke filtera za skupa za testiranje

Kao što možete vidjeti, podijelili smo skup na 4 jednaka dijela i odabrali smo prvi dio koji ćemo izbaciti, odnosno sačuvati za testiranje. Sve postavke i sam filter postaju aktivni, tek nakon što kliknemo na gumb „*Apply*“, odnosno, primjeni. Na slici ispod možete zapaziti, da je od skupa u kojem je bilo više od osam tisuća zapisa, sada ostala samo ¼ , odnosno 25%. Te podatke sada treba spremiti kao zasebni skup podataka i nakon toka, vratiti originalni skup, u prvobitno stanje.



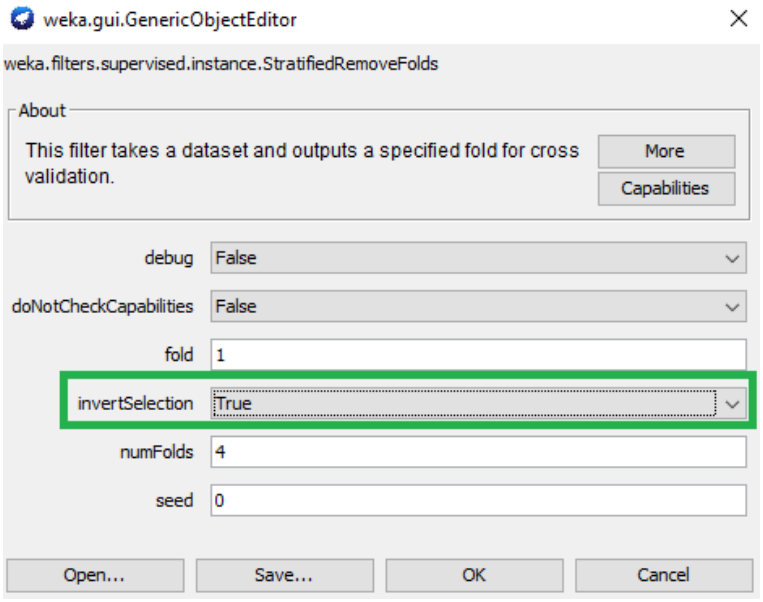
Slika 29 Informacije o testnom skupu

Kako bi vratili skup u stanje prije nego što je primijenjen filter, potrebno je samo kliknuti na gumb „*Undo*“. Time ćemo ponovo imati na raspolaganju svih 8124 zapisa.



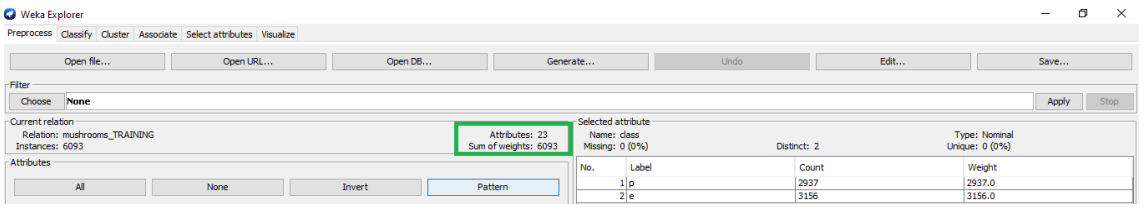
Slika 30 Stanje skupa nakon poništavanja učinka filtera

Pošto je skup podataka koji će se koristiti za testiranje spremljen u posebnoj datoteci, sada treba te iste zapise izbaciti iz starog skupa, koji ćemo koristiti za treniranje modela. Za to je potrebno ponovo otvoriti postavke filtera i samo odabrati inverzan odabir „*invertSelection*“, kako bi nam ostalo 75% podataka, bez onih 25% koje smo spremili i koje ćemo kasnije koristiti, za testiranje.



Slika 31 Postavke filtera za skupa za treniranje

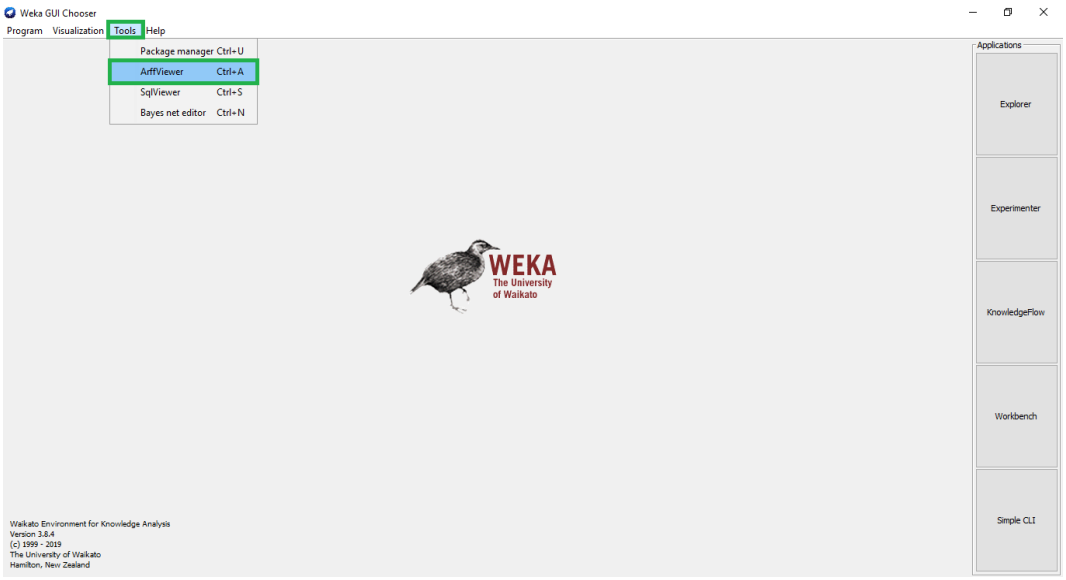
Nakon što se primjeni filtar, imat ćemo 6093 zapisa u skupu podataka, ali će njihova distribucija ostati približno ista, kao što je bila na početku u originalnom skupu, sa 8124 zapisa.



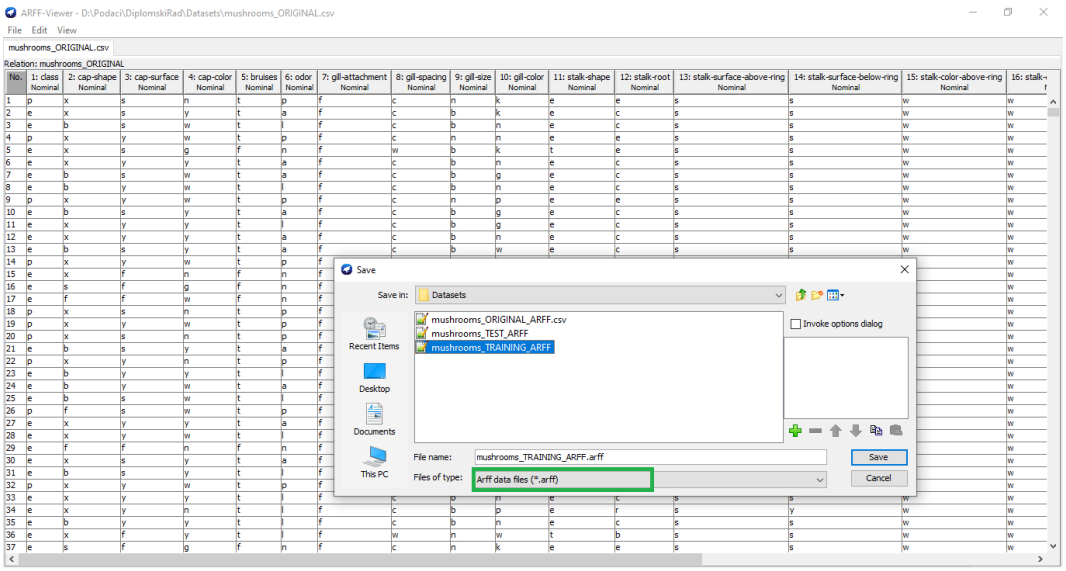
Slika 32 Informacije o skupu za treniranje

5.7.3. Pretvaranje u ARFF format

Sad kada smo razdvojili sve podatke, još ih je potrebno pretvoriti iz „*CSV*“ u format, koji je bolje razumljiv programu. Za to je potrebno otvoriti „*ArffViewer*“ u glavnom izborniku, kao što je to prikazano na slici ispod. Nakon toga, još treba otvoriti datoteku za treniranje i pohranit ju u „*.arff*“ formatu i isto to napraviti za datoteku, u kojoj su pohranjene vrijednosti za testiranje.

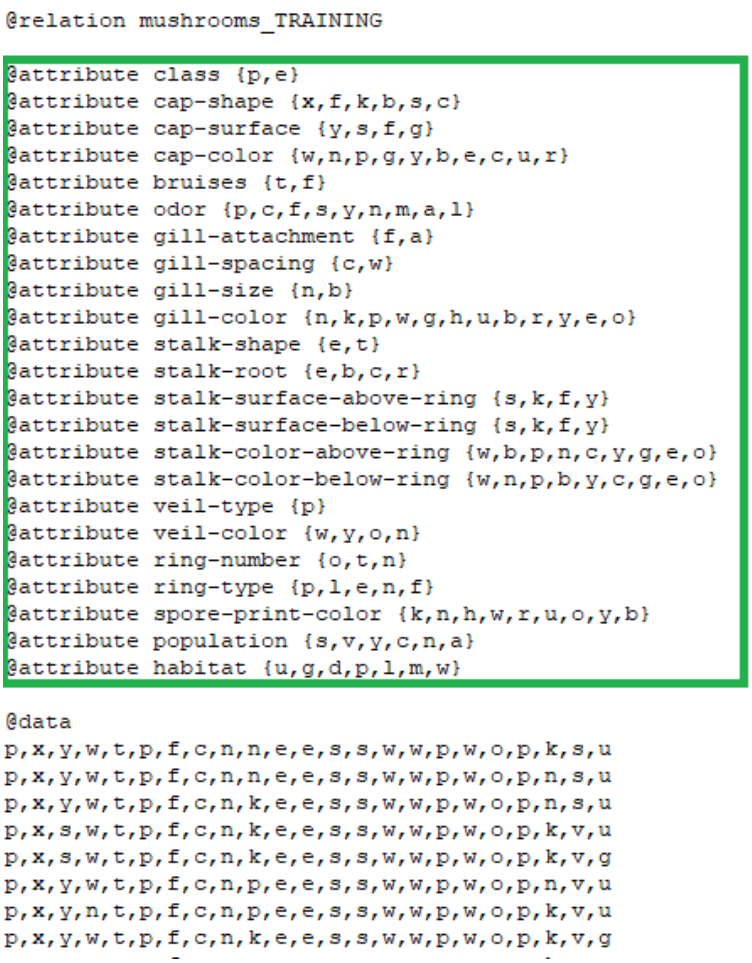


Slika 33 Lokacija alata "ArffViewer"



Slika 34 Učitani podaci i ispravna ekstenzija za pohranu datoteke

Kako ne bi došlo do greške kada budemo koristili te dvije datoteke prilikom testiranja, potrebno ih je otvoriti i provjeriti, da li su u sadržane sve klase, koje se koriste u atributima.



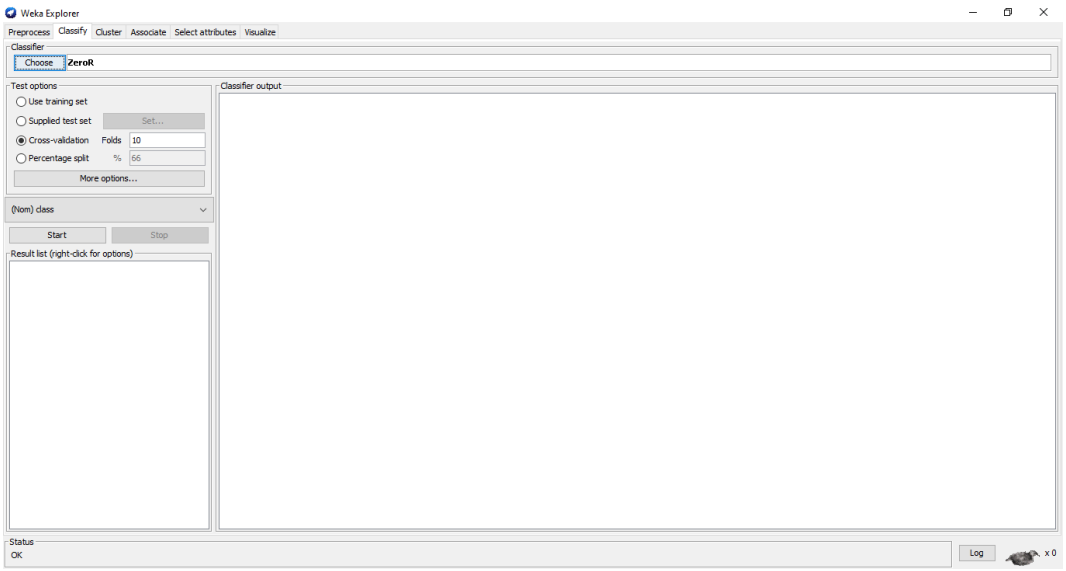
```
@relation mushrooms_TRAINING

@attribute class {p,e}
@attribute cap-shape {x,f,k,b,s,c}
@attribute cap-surface {y,s,f,g}
@attribute cap-color {w,n,p,g,y,b,e,c,u,r}
@attribute bruises {t,f}
@attribute odor {p,c,f,s,y,n,m,a,l}
@attribute gill-attachment {f,a}
@attribute gill-spacing {c,w}
@attribute gill-size {n,b}
@attribute gill-color {n,k,p,w,g,h,u,b,r,y,e,o}
@attribute stalk-shape {e,t}
@attribute stalk-root {e,b,c,r}
@attribute stalk-surface-above-ring {s,k,f,y}
@attribute stalk-surface-below-ring {s,k,f,y}
@attribute stalk-color-above-ring {w,b,p,n,c,y,g,e,o}
@attribute stalk-color-below-ring {w,n,p,b,y,c,g,e,o}
@attribute veil-type {p}
@attribute veil-color {w,y,o,n}
@attribute ring-number {o,t,n}
@attribute ring-type {p,l,e,n,f}
@attribute spore-print-color {k,n,h,w,r,u,o,y,b}
@attribute population {s,v,y,c,n,a}
@attribute habitat {u,g,d,p,l,m,w}

@data
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,s,u
p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,s,u
p,x,s,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,u
p,x,s,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,g
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,v,u
p,x,y,n,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,u
p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,g
```

Slika 35 Prikaz zapisa atributa, njihovih klasa i podataka u „.arff“ formatu

Najbolje bi bilo da su u datoteci za treniranje i za testiranje sadržane sve klase, premda se ne javljaju u samim podacima. Najlakši i najbrži način da to postignemo bi bio, da pretvorimo originalni skup sa svim podacima u „*.arff*“ format, te da nakon toga kopiramo ovaj dio i zalijepimo ga u datoteke, za treniranje i testiranje. Na taj način ćemo osigurati kompatibilnost između njih, prilikom testiranja i evaluacije.

5.8. Provođenje algoritma

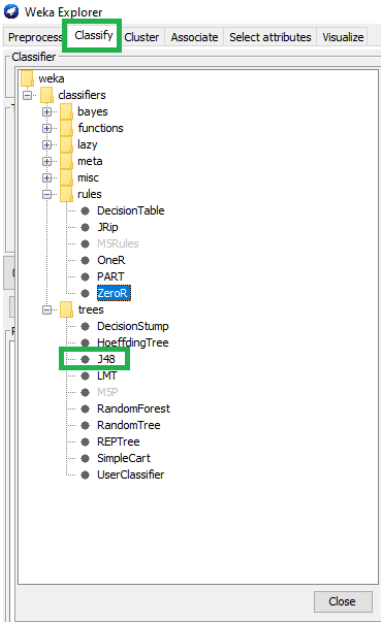
Kako bismo započeli sa samom klasifikacijom, potrebno je nakon završetka pred procesiranja podataka, kliknuti na „*Classify*“ gumb. Nakon toga se otvara slijedeći prozor, u kojem se vrši odabir algoritma koji će se koristiti i njegovo podešavanje, kako bismo dobili što bolje i kvalitetnije rezultate.



Slika 36 Prozor za provođenje klasifikacije

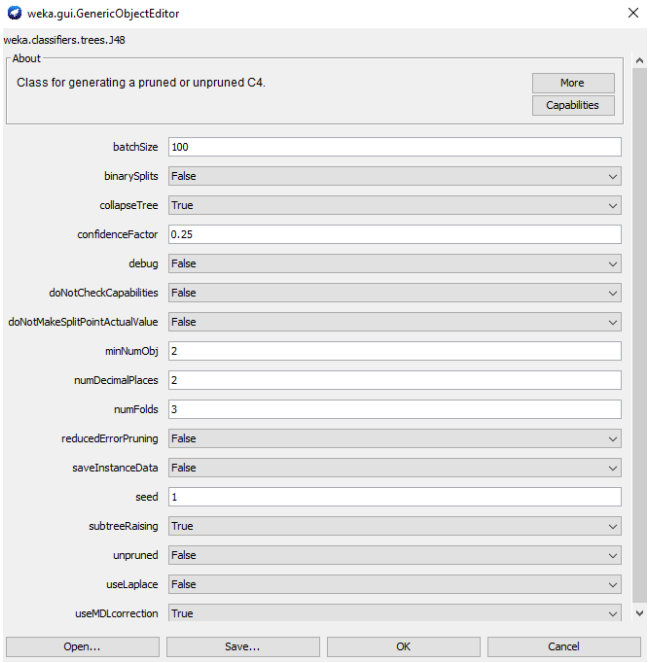
Na slici iznad je prikazan početni prozor sa unaprijed određenim postavkama. Prva stvar koju je potrebno učiniti je, odabrati jedan od brojnih algoritama koji se mogu koristiti. Neki algoritmi mogu raditi sa samo numeričkim, neki samo s nominalnim, a neki mogu raditi s kombinacijom tih tipova podataka. Ukoliko je algoritam sive boje, to znači da nije namijenjen za rad s tim tipom podataka.

Na slijedećoj slici se nalazi popis većine algoritama koji se i inače koriste za strojno učenje, a ukoliko neki koji želite nije na popisu, vrlo vjerojatno se može dodatno instalirati. Mi ćemo odabrati algoritam C4.5 iz skupine, koji se koriste za stabla odlučivanja i koji je ovdje implementiran, pod imenom J48.



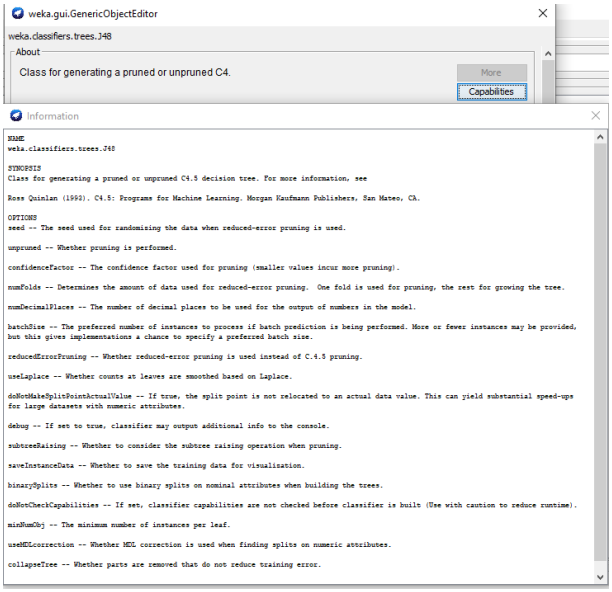
Slika 37 Algoritmi za klasifikaciju

Na slici ispod možete vidjeti unaprijed određene vrijednosti postavka koje program sam stavlja i koje su najčešće dobro podešene za generalnu upotrebu.



Slika 38 Zadane postavke za algoritam J48

Kao što možete vidjeti, ima jako puno postavki koje se mogu uključiti, isključiti i podesiti. Ukoliko niste sigurni što koja postavaka radi, odnosno koja je njezina svrha, možete kliknuti na gumb „*Capabilities*“. Otvorit će se novi prozor u kojem su dodatno pojašnjene postavke svakog algoritma koji odaberete.



Slika 39 Dodatne informacije i J48 algoritmu

Obzirom da ih ima jako puno i pošto je većina njih već jako dobro objašnjena u samom programu, opisat ćemo samo dvije koje su važne za ovaj primjer i koje će se mijenjati.

Prva opcija koja nas zanima je „*binarySplits*“. Pomoću nje, velimo programu da smije ili ne smije provoditi binarna grananja nad nominalnim atributima. Ukoliko isključimo ovu opciju, omogućiti ćemo grananje atributa po svim njegovim vrijednostima, odnosno klasama. Time će se možda dobiti malo veće i teže čitljivo stablo, no granat ćemo po svakom atributu samo jednom. Ukoliko pak uključimo binarno grananje, može se desiti da ćemo po istom atributu, granati više puta, što može imat utjecaj na rezultat.

Druga opcija koja nas zanima i koju ćemo podesiti na 10, zove se „*minNumObj*“ i vezana je uz najmanji broj objekta po listu. Ova opcija je korisna, jer pomoću nje možemo smanjiti problem pretreniranosti stabla.

5.9. Opis i značenje dobivenih rezultata

Zadnja stvar koju valja napomenuti, prije nego što započnemo s provođenjem algoritma je, da ćemo za testiranje koristiti metodu „*Supplied test set*“, za koju ćemo koristiti dva skupa podataka, koja smo dobili podijelim originalnog skupa. Prvi skup u kojem se nalazi 75% instanci ćemo koristiti za treniranje modela, a preostalih 25%, za njegovo testiranje. Nakon toga ćemo usporediti rezultat dobiven korištenjem binarnog grananja, sa onim kada ga ne koristimo .

5.9.1. Provođenje algoritma bez binarnog grananja

```
=== Summary ===

Correctly Classified Instances        2031               100      %
Incorrectly Classified Instances         0                 0      %
Total Number of Instances             2031

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  Class
                 1,000    0,000    1,000      1,000    1,000      p
                 1,000    0,000    1,000      1,000    1,000      e
Weighted Avg.    1,000    0,000    1,000      1,000    1,000

=== Confusion Matrix ===

    a    b   <-- classified as
  979    0 |    a = p
    0 1052 |    b = e
```

Slika 40 Sažetak rezultata bez korištenja binarnog grananja

Na slici iznad prikazan je rezultat evaluacije, odnosno testiranja modela. Kao što se može vidjeti, program je vrlo brzo proveo algoritam strojnog učenja nad podacima koje smo mu zadali, te su izračunate i prikazane mjere evaluacije. Iako *Weka* ima puno više mjera za evaluaciju dobivenog modela, smatram da je bolje za ovaj primjer koristiti samo one s kojima smo se dosad upoznali i koje smo prethodno obradili u teorijskom dijelu rada.

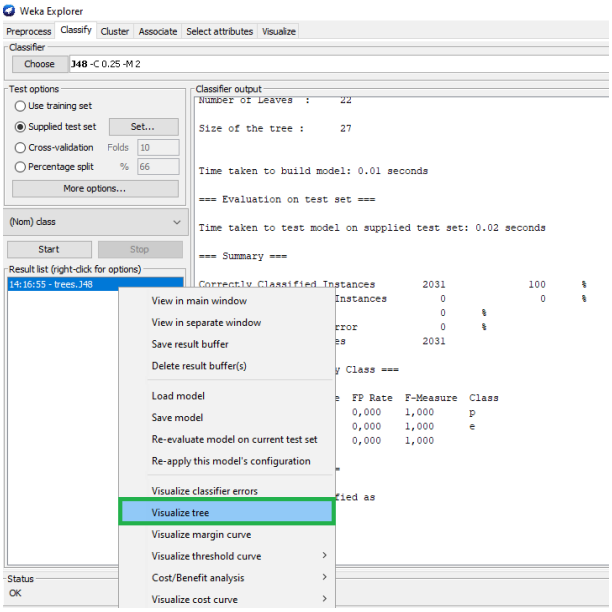
Ispravno klasificirane instance (*eng. Corecctly Classified Instances*) iznose 100%, odnosno, sve instance iz skupa podataka za testiranje su, ispravno klasificirane. To se također može vidjeti ako pogledamo matricu konfuzije, u kojoj su vrijednosti samo na glavnoj dijagonali.

Pošto je J48 algoritam, koji se koristi za dobivanje modela klasifikacije u obliku stabla odlučivanja, program također prikazuje dobiveni model stabla u tekstualnom obliku. Možete zamijetiti da se kraj svakog lista u zagradi, nalazi broj koji nam govori koliko instanci ima u tom listu.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

odor = p: p (191.0)
odor = c: p (144.0)
odor = f: p (1632.0)
odor = s: p (401.0)
odor = y: p (453.0)
odor = n
|   spore-print-color = k: e (950.0)
|   spore-print-color = n: e (1025.0)
|   spore-print-color = h: e (37.0)
|   spore-print-color = w
|   |   gill-size = n
|   |   |   stalk-surface-below-ring = s
|   |   |   |   cap-color = w: p (5.0)
|   |   |   |   cap-color = n: e (10.0)
|   |   |   |   cap-color = p: e (0.0)
|   |   |   |   cap-color = g: e (0.0)
|   |   |   |   cap-color = y: e (0.0)
|   |   |   |   cap-color = b: e (0.0)
|   |   |   |   cap-color = e: e (0.0)
|   |   |   |   cap-color = c: e (12.0)
|   |   |   |   cap-color = u: e (0.0)
|   |   |   |   cap-color = r: e (0.0)
|   |   |   stalk-surface-below-ring = k: e (0.0)
|   |   |   stalk-surface-below-ring = f: e (16.0)
|   |   |   stalk-surface-below-ring = y: p (32.0)
|   |   gill-size = b: e (396.0)
|   spore-print-color = r: p (53.0)
|   spore-print-color = u: e (0.0)
|   spore-print-color = o: e (41.0)
|   spore-print-color = y: e (29.0)
|   spore-print-color = b: e (36.0)
odor = m: p (26.0)
odor = a: e (301.0)
odor = l: e (303.0)

Number of Leaves  :     30

Size of the tree :  35
```

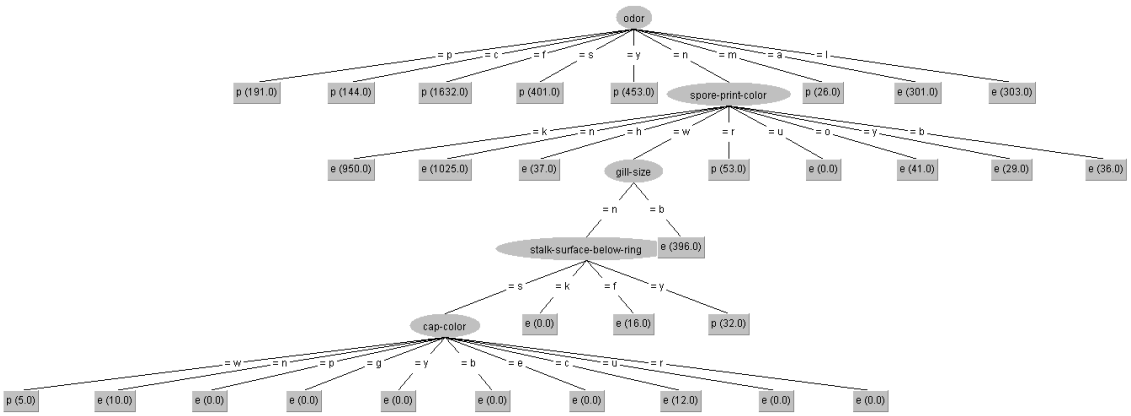
Slika 41 Tekstualni prikaz stabla bez korištenja binarnog grananja

Iako se ovdje radi o relativno malom stablu, ovaj tekstualni prikaz modela je svejedno malo teže pratiti. *Weka* na sreću, omogućuje puno pregledniju i bolju vizualizaciju dobivenog modela, za koju je samo potrebno desnim klikom miša, za rezultat, odabrati opciju „*Visualize Tree*“.



Slika 42 Odabir opcije za vizualizaciju stabla

Nakon što odaberemo opciju, otvara se novi prozor, u kojem je prikazan puno pregledniji model stabla odlučivanja, koji je generiran pomoću podataka koje smo proslijedili i algoritma kojeg smo odabrali i podesili.



Slika 43 Vizualni prikaz stabla odlučivanja bez korištenja binarnog grananja

Ova vizualizacija od prethodne, koja je bila u tekstualnom formatu, a dobiveni model stabla, iako je relativno velik, nije toliko kompliciran i nepregledan, a da ga ne bi i dalje lako moguće iščitati i slijediti pravila, od korijena do bilo kojeg lista.

5.9.2. Provođenje algoritma s binarnim grananjem

```
=== Summary ===

Correctly Classified Instances        2028               99.8523 %
Incorrectly Classified Instances         3                0.1477 %
Total Number of Instances             2031

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  Class
                 0,997    0,000    1,000      0,997    0,998      p
                 1,000    0,003    0,997      1,000    0,999      e
Weighted Avg.    0,999    0,002    0,999      0,999    0,999

=== Confusion Matrix ===

    a    b   <-- classified as
  976    3 |    a = p
    0 1052 |    b = e
```

Slika 44 Sažetak rezultata sa binarnim grananjem

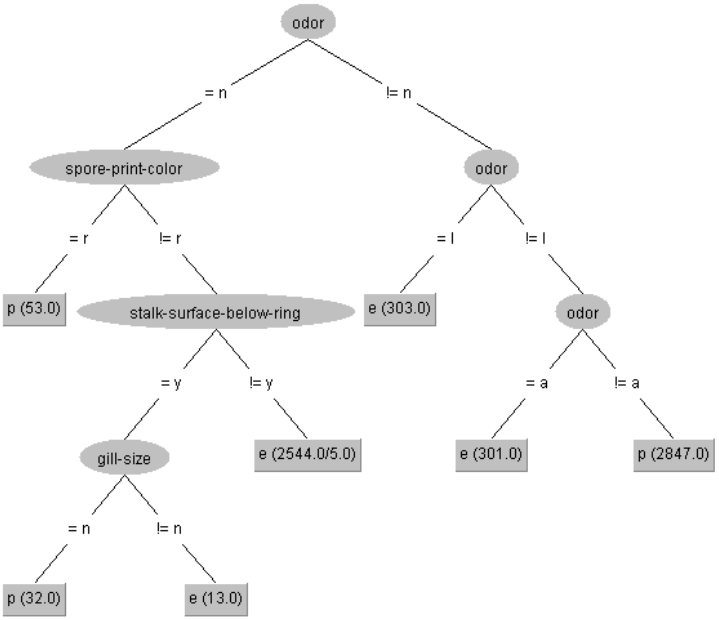
Prva stvar koja se može zamijetiti je, da su rezultati koje smo dobili s uključenim binarnim grananjem, malo gori (manje od 1%). Ako sad pogledamo matricu konfuzije, možemo vidjeti da su tri gljive, koje su zapravo otrovne, krivo klasificirane pod jestive, što i nije tako loše. Ovaj rezultat nam govori da, ukoliko bi našem modelu dali da neku novu instancu za klasifikaciju, vjerojatnost da će ta instanca biti krivo klasificirana, je manja od 1 %.

Na slijedećoj slici je ponovo prikazano stablo odlučivanja, no ovaj put se radi o modelu u kojem je korišteno binarno grananje.

```
J48 pruned tree
------------------

odor = n
|   spore-print-color = r: p (53.0)
|   spore-print-color != r
|   |   stalk-surface-below-ring = y
|   |   |   gill-size = n: p (32.0)
|   |   |   gill-size != n: e (13.0)
|   |   stalk-surface-below-ring != y: e (2544.0/5.0)
odor != n
|   odor = l: e (303.0)
|   odor != l
|   |   odor = a: e (301.0)
|   |   odor != a: p (2847.0)

Number of Leaves  :     7

Size of the tree :      13
```

Slika 45 Tekstualni prikaz stabla odlučivanja sa binarnim grananjem

Odmah se može vidjeti da je ovo stablo značajno manje od prethodnog, što bi se moglo smatrati poboljšanjem. Uvijek bi se trebalo pokušati proizvesti što manje stablo, a da se pritom ne gubi na točnosti i kvaliteti dobivenog modela. Unatoč tome, vi na kraju odlučujete o tome, da li je dobiveni model prihvatljiv ili ne. Na slici ispod je vizualno prikazan model stabla, koje smo dobili korištenjem binarnog grananja.



Slika 46 Vizualni prikaz stabla odlučivanja sa binarnim grananjem

Ako malo pobliže pogledamo sliku, možemo zamijetiti da smo korištenjem binarnog grananja dobili stablo, u kojem imamo više grananja po istom atributu, čega u prethodnom modelu, nije bilo. To je zato što, višestruko grananje iscrpi sve mogućnosti nekog nominalnog atributa, pa je manja vjerojatnost da ćemo više puta garanti po istom.

6. Zaključak

Kroz pisanje ovog rada i upoznavanjem sa teorijom koja je primarno vezana uz inteligentne sustave, rudarenje podataka, strojno učenje stabla odlučivanja, stekao sam razna znanja i bolje razumijevanje tih područja. Također sam dobio bolji smisao o kompleksnosti i važnosti, te različitim načinima, mogućnostima i područjima primjene, prethodno spomenutih pojmova.

Provođenjem procesa, čiji su cilj i svrha bili otkrivanje znanja u skupu podataka koji su vezani uz gljive, dobio sam bolje razumijevanje i smisao o važnosti kvalitetnih podataka. Puno su mi jasniji koraci vezani uz podataka, njihovo prikupljanje, čišćenje, obrada i cijeli proces koji je vezan uz pred procesiranje. Prethodno spomenute aktivnosti vezane uz podatke, s razlogom, uzimaju naviše vremena. Ključne su, kako bi se podaci što bolje pripremili za daljnju obradu i time pridonijeli što kvalitetnijem rezultatu.

Potrebno je prepoznati potencijalnu vrijednost podataka, kojih u novije vrijeme ima sve više i više. Kažem potencijalnu, jer bez obzira na količinu podataka koje imamo pohranjene i njihovu kvalitetu, oni neće vrijediti ništa, ukoliko ih ne obradimo i dobivene rezultate ne iskoristimo u nekom sustavu.

Strojno učenje se razvija velikom brzinom i već sada ima velik broj primjena u raznim područjima kao što su bankarstvo, medicina, promet i mnoga druga. Također imamo različite vrste i metode strojnog učenja, obzirom na cilj koji želimo postići ili zadatak koji želimo izvršiti. Unatoč svojim mnogobrojnim prednostima, strojno učenje također ima nedostatke, kao što su problem odabira najboljeg algoritma za provođenje strojnog učenja, njegovo podešavanje i na kraju evaluacija. Sve to zahtjeva znanje, ali i iskustvo osobe koja provodi strojno učenje. Važno je biti upoznat, ne samo sa algoritmima koji se koriste, nego i sa samom problemskom domenom.

Algoritmi za dobivanje modela stabla odlučivanja, s razlogom su jedni od najpopularnijih. Dobiveni model, koji zbog svog karakterističnog izgleda zovemo stablo odlučivanja, jednostavan je za razumijevanje i čitanje. S lakoćom možemo iščitati skup pravila od kojih se on zapravo sastoji i implementirati ih, u neki program ili sustav.

Popis literature

[1] Hopgood, Intelligent systems for engineers and scientists, 2nd ed. Boca Raton: CRC Press, 2001.

[2] F. Gržinić Kuljanac, "Inteligentni sustavi za modeliranje poslovnih procesa u razvoju informacijskog sustava", Semanticscholar.org, 2015. [Online]. Dostupno: https://www.semanticscholar.org/paper/Inteligentni-sustavi-za-modeliranje-poslovnih-u-Tita/02d107906bfa08a3355937890affcd1d63b9a4d3. [Pristupljeno : 21- 06- 2020].

[3] C. Longbing, "Data Science: A Comprehensive Overview", Dl.acm.org, 2017. [Online]. Dostupno: https://doi.org/10.1145/3076253. [Pristupljeno : 21- 06- 2020].

[4] L. Rokach and O. Maimon, Data mining with decision trees, 2nd ed. Singapore: World Scientific, 2015.

[5] "Data Science Croatia (Zagreb, Croatia)", Meetup. [Online]. Dostupno: https://www.meetup.com/DataScienceCroatia/. [Pristupljeno : 21- 06- 2020].

[6] Čavalić, "UTJECAJ KVALITETE PODATAKA I INFORMACIJA NA KVALITETU ODLUKE", Hrcak.srce.hr, 2016. [Online]. Dostupno: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=252983. [Pristupljeno : 21- 06- 2020].

[7] D. Sarkar, "Categorical Data: Strategies for working with discrete, categorical data", Medium, 2018. [Online]. Dostupno: https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63. [Pristupljeno : 21- 06- 2020].

[8] E. Alpaydin, Introduction to machine learning. Cambridge, Mass.: Mit Press, 2004.

[9] M. Pejić Bach, "Rudarenje podataka u bankarstvu", Hrcak.srce.hr, 2005. [Online]. Dostupno: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=41477. [Pristupljeno : 19- Mar- 2020].

[10] "Taxonomy definition and meaning | Collins English Dictionary", Collinsdictionary.com. [Online]. Dostupno: https://www.collinsdictionary.com/dictionary/english/taxonomy. [Pristupljeno : 21- 06- 2020].

[11] H. Hamilton, "Computer Science 831: Knowledge Discovery in Databases", www2.cs.uregina.ca, 2009. [Online]. Dostupno: http://www2.cs.uregina.ca/~dbd/cs831/index.html. [Pristupljeno : 21- 06- 2020].

[12] M. Chowdhury, A. Apon and K. Dey, Data analytics for intelligent transportation systems. Elsevier Inc., 2017.

[13] M. Smith, "What is learning? A definition and discussion", The encyclopedia of pedagogy and informal education, (1999-2020). [Online]. Dostupno: https://infed.org/mobi/learning-theory-models-product-and-process/. [Pristupljeno : 22- 06- 2020].

[14] T. Mitchell, Machine learning. New York: McGraw-Hill, 1997.

[15] R. Schapire, "COS 511: Theoretical Machine Learning", Cs.princeton.edu, 2008. [Online]. Dostupno: https://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf. [Pristupljeno : 21- 06- 2020].

[16] Anirudh. V K, "What Is Machine Learning: Definition, Types, Applications and Examples", It.toolbox.com, 2019. [Online]. Dostupno: https://it.toolbox.com/tech-101/what-is-machine-learning-definition-types-applications-and-examples. [Pristupljeno : 22- 06- 2020].

[17] S. Russell and P. Norvig, Artificial intelligence a modern approach, 3rd ed. Upper Saddle River (N.J.): Pearson, 2010.

[18] B. Divjak i A. Lovrenčić, Diskretna matematika s teorijom grafova. Varaždin: TIVA Tiskara Varaždin, 2005.

[19] "Tree Based Algorithms: A Complete Tutorial from Scratch (in R & Python)", Analytics Vidhya, 2016. [Online]. Dostupno: https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/#two. [Pristupljeno : 22- 06- 2020].

[20] J. Quinlan, C 4.5: programs for machine learning. San Mateo, Calif.: Morgan Kaufmann, 1993.

[21] V. Zhou, "A Simple Explanation of Gini Impurity", Victorzhou.com, 2019. [Online]. Dostupno: https://victorzhou.com/blog/gini-impurity/. [Pristupljeno : 22- 06- 2020].

[22] "WEKA The workbench for machine learning", Cs.waikato.ac.nz. [Online]. Dostupno: https://www.cs.waikato.ac.nz/ml/weka/. [Pristupljeno : 22- 06- 2020].

[23] "Mushroom Classification: Safe to eat or deadly poison?", Kaggle.com, 2016. [Online]. Dostupno: https://www.kaggle.com/uciml/mushroom-classification. [Pristupljeno : 22- 06- 2020].

[24] E. Bossenmaier i S. Kaminskyj, "Fungi of Saskatchewan", Usask.ca. [Online]. Dostupno: https://www.usask.ca/biology/fungi/glossary.html. [Pristupljeno : 22- 06- 2020].

[25] "5 dijelova gljiva i njegove značajke", Thpanorama. [Online]. Dostupno: https://hr.thpanorama.com/articles/biologa/las-5-partes-de-un-hongo-y-sus-caractersticas.html. [Pristupljeno : 22- 06- 2020].

Popis slika

Popis tablica