SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Mirjana Domazet-Lošo

USPOREDBA POSTUPAKA DUBINSKE ANALIZE PRIMIJENJENIH NAD BIOLOŠKIM PODACIMA

MAGISTARSKI RAD

Zagreb, 2006.

Magistarski rad izrađen je na Zavodu za primijenjeno računarstvo, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

Mentor: prof.dr.sc. Mirta Baranović

Magistarski rad ima: 84 stranice

Magistarski rad br.:

Povjerenstvo za ocjenu u sastavu:

1. Prof.dr.sc. Damir Kalpić – predsjednik
2. Prof.dr.sc. Mirta Baranović – mentor
3. Prof.dr.sc. Mladen Varga – Ekonomski fakultet, Sveučilište u Zagrebu

Povjerenstvo za obranu u sastavu:

1. Prof.dr.sc. Damir Kalpić – predsjednik
2. Prof.dr.sc. Mirta Baranović – mentor
3. Prof.dr.sc. Mladen Varga – Ekonomski fakultet, Sveučilište u Zagrebu

Rad je obranjen 13. lipnja 2006.

Zahvaljujem prof.dr.sc. Mirti Baranović na pomoći
prilikom izrade ovog rada.

Sadržaj

1. Uvod

Dubinska analiza podataka (eng. *data mining*) je netrivijalni postupak izlučivanja novih saznanja iz postojećih podataka ili uočavanja anomalija u podacima bez nužnog poznavanja semantike podataka [1], [2], [3], [5], [6]. Kao nova grana računalne znanosti nastala je krajem 90-ih godina prošlog stoljeća pod okriljem IBM-a, koji drži vlasništvo nad nekim patentima [3]. Dubinska analiza podataka je presjek statistike, raspoznavanja uzoraka, strojnog učenja, tehnologija baza podataka i skladišta podataka.

Osnovni zadaci koji su postavljeni pred dubinsku analizu podataka mogu su podijeliti u dvije skupine: prepoznavanje svojstava podataka (eng. *descriptive task*) i predviđanje ponašanja poznatih podataka (eng. *predictive task*) [5]. Nova saznanja mogu biti statistički modeli, pravila ponašanja koja slijede podaci ili odstupanja u podacima. Modeli dubinske analize mogu generirati mnoštvo novih saznanja i pravila, ali se samo manji dio njih može smatrati korisnima. Korisnost i zanimljivost pojedinog pravila se određuju prema tome da li je pravilo razumljivo krajnjem korisniku, da li pravilo donosi neku novu informaciju ili potvrđuje neku pretpostavku koju je korisnik postavio.

Popularnost primjene dubinske analize posljedica je neučinkovitosti standardnih sustava za praćenje i analizu podataka da u velikim količinama podataka pronađu nove i korisne informacije, što se posebno odnosi na tehnologiju baza podataka i skladišta podataka. Sustavi za upravljanje bazama podataka su orijentirani prema svakodnevnim transakcijama nad normaliziranima bazama podataka, pa s obzirom na goleme količine podataka koji se u bazama podataka godinama prikupljaju, te sporost i složenost upita koji se obavljaju nad podacima čine baze podataka neprikladnima za analizu podataka. Skladišta podataka su orijentirana prema analizi prikupljenih podataka, gdje su analitički izvještaji dobiveni na temelju smjernica koje zadaje korisnik i ne donose inovativnost u vezama između pojedinih podataka. Potrebu za donošenjem novih spoznaja na temelju postojećih podataka pružila je dubinska analiza podataka, čime je postala nadogradnja na postojeće sustave za upravljanje bazama podataka i skladištima podataka [5], [6]. Često se dubinska analiza povezuje u kontekstu vezanom uz OLAP (*On-Line Analytical Processing*) čija je namjena analiza podataka. Međutim, analiza kroz OLAP alate služi kako bi se opovrgle ili dokazale pretpostavke korisnika i isključivo opisuje podatke bez mogućnosti predviđanja. Dubinska analiza podataka je istovremeno opisni i prediktivni alat, koji u oba slučaja neovisno o korisniku, sam donosi nova saznanja [6]. Kao nova tehnologija, poveznica OLAP tehnologije i dubinske analize podataka, nastao je OLAM (*On-Line Analytical Mining*) [51]. Prednost OLAM tehnologije je korištenje konsolidiranih, čistih i integriranih podataka koji se već nalaze u skladištima podataka.

Razlika između statistike i strojnog učenja s jedne strane i dubinske analize podataka s druge strane je u veličini skupova podataka nad kojima se obavljaju ispitivanja. U statistici i strojnom učenju koriste se puno manji skupovi podataka u odnosu na veličinu skupova podataka nad kojima se primjenjuje dubinska analiza podataka, pa zbog toga, mnogi algoritmi koji imaju primjenu u statistici i strojnom učenju postaju neprimjenjivi pri dubinskoj analizi podataka, te su nužne njihove prilagodbe.

Primjena dubinska analize raširena je u mnogim granama: u bankarskim aplikacijama koristi se npr. za utvrđivanje kreditne sposobnosti klijenta; u marketingu se

koristi kako bi se prepoznala ciljana skupina potrošača koji svi dijele iste interese (profiliranje kupaca); osiguravajuća društva koriste dubinsku analizu kako bi im, među ostalima, pomogla u utvrđivanju graničnih slučajeva; u medicini se koristi kao pomoć pri prepoznavanju simptoma određenih bolesti, te za utvrđivanje u kojoj mjeri pojedini lijekovi i terapije utječu na pacijente; u aplikacijama elektroničkih trgovina koriste se za prepoznavanje sklonosti kupaca, te je na temelju njih moguće sugerirati kupcu sljedeću kupovinu (npr. na temelju knjiga koje je do tog trenutka kupac kupio, sugeriraju mu se novi naslovi); u farmaceutskoj industriji koristi se kako bi se utvrdila veza između pojedinih gena i bolesti [6].

U ovom će radu biti prikazana primjena dubinske analize u biologiji. Istraživanje ponašanja bioloških sustava rezultira mnoštvom prikupljenih informacija o njihovoj dinamici i funkcioniranju, a velike količine tako prikupljenih podataka često ne pružaju mogućnost izlučivanja bitne informacije. Zbog toga se za izlučivanje bitnih informacija koriste metode dubinske analize podataka. Ovdje je prikazan postupak kojim se na temelju podataka prikupljenih biološkim istraživanjima određuje koji geni utječu na promatrani biološki sustav.

U sljedećem je poglavlju opisana veza dubinske analize podataka s cjelokupnim procesom otkrivanja znanja u podacima. U trećem su poglavlju opisane metode smanjivanja dimenzijske složenosti kao neizostavnog postupka u pripremi podataka opisanih mnoštvom atributa (kakvi su biološki podaci) za samu analizu. U četvrtom su poglavlju opisane osnovne metode dubinske analize s detaljnijim prikazom metoda razvrstavanja koje su korištene u praktičnom dijelu rada. Peto poglavlje opisuje način evaluacije rezultata. S obzirom da je rezultat praktičnog dijela rada klasifikacija testnog skupa podataka, evaluacija rezultata se odnosi na evaluaciju algoritama klasifikacije. U šestom je poglavlju opisana struktura podataka koja će se koristiti u testiranju, te priprema podataka za analizu. Rezultati dobiveni klasifikacijom testnog skupa podataka prikazani su u sedmom poglavlju.

2. Dubinska analiza podataka i proces otkrivanja znanja

Dubinska analiza podataka i *proces otkrivanja znanja u podacima* (eng. *Knowledge Discovery in Databases* ili akronim *KDD*) često se koriste kao sinonimi, iako je dubinska analiza samo jedan od koraka u procesu otkrivanja znanja [5] kojoj prethodi mukotrpna i dugotrajna priprema podataka. Pokazalo se da sama priprema podataka, a to su koraci pročišćavanja podataka, te njihova integracija i transformacija čine 60% - 95% ukupnog vremena koje je potrebno u cjelokupnom procesu otkrivanja znanja u podacima [6]. Onaj zanimljiviji dio, a to je sama dubinska analiza podataka čini tek manji dio ukupno utrošenog vremena. Analiza bioloških podataka prikazana u ovom radu u potpunosti odgovara takvim vremenskim odnosima – otprilike dvije trećine ukupnog vremena posvećenog praktičnom dijelu rada odnosilo se na pripremu podataka.

Cjeloviti proces otkrivanja znanja u podacima sastoji se od sljedećih koraka (slika 1.1.):

1. *pročišćavanje podataka* (eng. *data cleaning*) – iz izvora podataka uklanjaju se podaci koji predstavljaju šum ili nisu konzistentni s drugim podacima

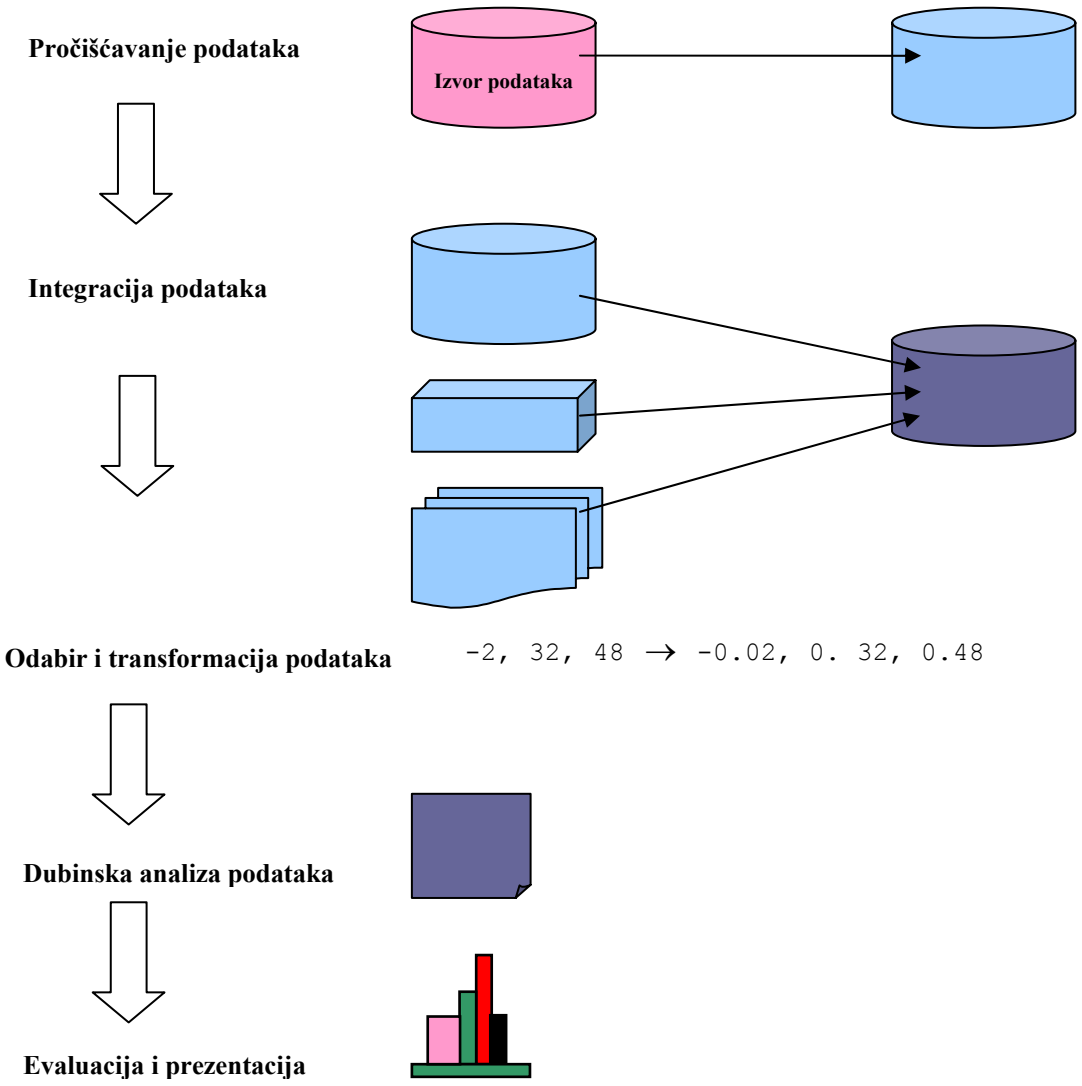
2. *integracija podataka* (eng. *data integration*) – više različitih izvora podataka integrira se u jedan; pri tome se podaci, ako je moguće, reduciraju

3. *transformacija podataka* (eng. *data transformation*) – podaci se transformiraju kako bi se sveli na oblik što prikladniji za analizu; ovaj postupak može uključivati numeričku transformaciju podataka (normiranje numeričkih atributa na određene intervale, tipično od 0.0 do 1.0; -1.0 do 1.0; 0.5 do -0.5); redukciju broja zapisa i redukciju broja atributa; konstrukciju novih atributa na temelju postojećih; generalizaciju atributa itd.

4. *dubinska analiza podataka* – primjena metoda koje kao rezultat pružaju nova pravila i veze među podacima

5. *evaluacija dobivenih rezultata* (eng. *pattern evaluation*) – iz skupa novih pravila odabiru se ona koja ispunjavaju određene evaluacijske mjere

6. *prezentacija znanja* (eng. *knowledge presentation*) – prezentacija novih pravila i veza među podacima na način razumljiv korisniku, npr. korištenjem vizualnih alata



Slika 2.1. Proces otkrivanja znanja u podacima

2.1. Pročišćavanje podataka

Izvori podataka za dubinsku analizu mogu biti baze podataka, skladišta podataka, tekstualne datoteke, proračunske tablice (eng. *spreadsheet*), multimedijalni podaci ili podaci prikupljeni s web-stranica. S obzirom da podaci iz stvarnog svijeta često imaju nedostatke koje je potrebno ukloniti, ti se nedostaci uklanjaju u fazi pročišćavanja podataka: nadopunjuju se podaci koji nedostaju, uklanja se šum iz podataka ili, ako je takve podatke moguće uočiti, uklanjaju se ekstremi, odnosno podaci koji odstupaju od cjeline prema određenim vrijednostima atributa.

2.1.1. Nadopunjavanje podataka koji nedostaju

Podatak koji nedostaje ili nije poznat (eng. *missing value*) je *null* vrijednost nekog atributa u n-torci ili uzorku podatka. Kako postupati s uzorkom koji sadrži nepoznatu vrijednost atributa ovisi o važnosti tog atributa. Ako je atribut nevažan za analizu ili je broj n-torki koje sadrži *null* vrijednost za neki atribut zanemariv, tada se takve n-torke mogu isključiti iz analize. Ako je atribut važan za analizu, onda se *null*-vrijednost mora zamijeniti s nekom drugom vrijednosti, npr. to može biti neka vrijednost dobivena numeričkim proračunom ili može biti specijalna oznaka "NEPOZNATO" s obzirom da se pokazalo da je ponekad ključan *nedostatak* neke informacije [6].

Uobičajene taktike koje se primjenjuju u slučajevima kada nedostaje vrijednost nekog atributa [5], [11], [12], [13]:

1. *ispuštanje n-torke iz daljnje analize* – ovaj postupak je uobičajen kada je nepoznata *oznaka razreda* kojemu n-torka pripada ili ako za neki uzorak nedostaju vrijednosti velikog broj atributa unutar n-torke. U ostalim slučajevima ova se metoda koristi rijetko, a posebno se ne preporučuje koristiti ako u skupu pokaznih uzoraka postoji puno n-torki kojima nedostaju vrijednosti pojedinih atributa.

2. *ručna nadopuna podataka koji nedostaju* – ovaj postupak je najčešće neizvediv u praksi s obzirom da broj n-torki kojima nedostaju vrijednosti pojedinih atributa može biti vrlo velik, pa bi vrijeme koje bi korisnik proveo za nadopunjavanje tih vrijednosti bilo veliko

3. *korištenje globalne konstante kojom će se nadopuniti vrijednosti koje nedostaju* – ovaj postupak podrazumijeva zamjenu svih vrijednosti koje nedostaju s istom oznakom, npr. "NEPOZNATO", čime je onda lakše uočiti ako postoji konceptualna veza između svih n-torki kojima je vrijednost nekog atributa "NEPOZNATO".

4. *zamjena srednjom vrijednosti atributa dobivenom na temelju poznatih vrijednosti tog atributa* – nedostatak ovog postupka je mogućnost dobivanja pogrešnih procjena zbog pogreške cirkularnosti. Naime, vrijednost koja se dobije ovom metodom za vrijednost nekog atributa može izravno utjecati na predviđanje razreda kojemu pojedini uzorak pripada, jer upravo ta vrijednost možda utječe na određivanje razreda. Drugi veliki problem kod ove metode je što ne postoji način kamo svrstati uzorke za koje treba predvidjeti razred, a nedostaje im vrijednost atributa koji se određivao prema pripadnosti razredu.

5. *zamjena srednjom vrijednosti atributa dobivenom na temelju poznatih vrijednosti tog atributa gdje su korištene samo one n-torke koje pripadaju istom razredu kao i n-torka kojoj nedostaje vrijednost tog atributa* – ova je metoda slična, ali preciznija od prethodnog postupka, iako i ovdje može doći do predviđanja pogrešne vrijednosti zbog pogreške cirkularnosti

6. *zamjena s najvjerojatnijom vrijednosti atributa* – najvjerojatnija vrijednost atributa se može dobiti korištenjem regresije, Bayesovog formalizma ili korištenjem stabla odlučivanja (npr. koriste se ostali atributi u n-torci kako bi se predvidjela vrijednost atributa koja je nepoznata). Ovaj postupak se često koristi, jer koristi najviše

dostupnih podataka kako bi predvidio nepoznatu vrijednost. Svi postupci iz ove kategorije su vrlo složeni i povećavaju programsku i vremensku kompleksnost cjelokupnog modela.

7. *pridruživanje svih poznatih vrijednosti atributu kojemu nedostaje vrijednost* – uzorak koji sadrži nepoznatu vrijednost atributa *A* zamjenjuje se skupom uzoraka, gdje je svaka pojava atributa *A* zamijenjena s jednom od poznatih vrijednosti tog atributa. U slučaju da su za neki uzorak nepoznate vrijednosti nekoliko atributa, onda je potrebno prvo za jedan atribut nadomjestiti njegovu vrijednost sa svim poznatim vrijednostima atributa, zatim za drugi atribut, itd.

8. *pridruživanje svih poznatih vrijednosti atributu kojemu nedostaje vrijednost uz ograničenje* – isto kao prethodni postupak uz ograničenje da se poznate vrijednosti pojavljuju samo uz one uzorke koji pripadaju istom razredu kao i onaj uzorak kojemu nedostaje vrijednost atributa

9. *generiranje indukcijskih DNF (disjunktna normalna forma) pravila* – generiraju se indukcijska pravila tako da je svaki razred određen s jednakim brojem pravila, gdje je svako pravilo jednake težine. Novi uzorak se razvrstava u onaj razred čija su pravila najviše ispunjena, tj. za koje pravilo ima najviše glasova [11].

2.1.2. Ispravljanje podataka s prisutnim šumom

Šum (eng. *noise*) je slučajna pogreška ili varijanca mjerene varijable, odnosno vrijednosti atributa. Često se koristi kako bi se izolirali oni uzorci podataka koji se ne mogu svrstati niti u jedan razred. U praksi takvi slučajevi mogu upućivati na slučajne ili namjerne pogreške.

Uobičajene taktike koje se primjenjuju u slučajevima kada je vrijednost nekog atributa šum [5]:

1. *ujednačavanje* (eng. *binning*) – particioniranje n-torki (uzoraka) u jednakobrojne skupove ili posude (eng. *bin*); a zatim izjednačavanje vrijednosti atributa unutar skupa izmjenom vrijednosti tako da nove vrijednosti postanu jednake:
    a. srednjoj vrijednosti ili medijanu svih vrijednosti u skupu
    b. bližoj između donje i gornje granice vrijednosti u skupu

2. *grupiranje* (eng. *clustering*) – n-torke sa sličnim svojstvima se grupiraju, tako da one n-torke koje odstupaju od svih grupa ostanu izolirane, pa se mogu smatrati podacima koji odstupaju od grupa (eng. *outliers*)

3. *kombinirano računalna i ljudska introspekcija podataka* – kombiniranim pristupom mogu se detaljno pregledati podaci koji odstupaju od svih grupa (eng. *outliers*)

4. *regresija* (eng. *regression*) – podaci se mogu izravnati tako da odgovaraju nekoj funkciji, npr. ako se koristi linearna funkcija, onda se postupak naziva linearna regresija ili višestruka linearna regresija

2.1.3. Identifikacija uzoraka koji odstupaju od cjeline (ekstrema)

Ekstremi ili uzorci koji odstupaju od cjeline (ostalih uzoraka) ili su nekonzistentni s njom (eng. *outliers*) mogu nastati pogrešnim unosom podataka ili nedosljednom varijabilnosti podataka, ali mogu ukazivati i na neke iznimne vrijednosti u uzorcima na koje posebno treba obratiti pozornost (*Ono što je jednoj osobi šum, drugoj može biti signal* [6]).

Većina algoritama koji se koriste u dubinskom istraživanju podataka nastoji minimizirati utjecaj podataka koji odstupaju od cjeline, ali u iznimnim prilikama te podatke se ne smije zanemariti (npr. u aplikacijama za osiguravajuća društva kada je potrebno otkriti prijevaru). U takvim slučajevima, potrebno je dubinski istražiti podatke koji odstupaju od cjeline (eng. *outlier mining*).

Osnovni princip uočavanja uzoraka koji odstupaju od cjeline je sljedeći: neka je zadan skup od n uzoraka i broj uzoraka k za koje se očekuje da odstupaju od cjeline. Potrebno je pronaći k uzoraka koji najviše odstupaju od svih ostalih podataka (najmanje su slični ostalim podacima ili je nekonzistencija najveća u odnosu na ostale podatke). Pronalazak uzoraka koji najviše odstupaju od cjeline može se promatrati kao dva problema:

(i) potrebno je definirati koji se uzorci mogu smatrati nekonzistentnim u zadanom skupu uzoraka
(ii) potrebno je odabrati učinkovitu metodu koja će pronaći nekonzistentne uzorke

Osnovne metode koje se koriste za računalni odabir uzoraka koji odstupaju od cjeline [5]:

1. *statistička metoda* – nad skupom uzoraka za koji vrijedi neka razdioba (npr. normalna razdioba) ili je nad skupom uzoraka definiran vjerojatnosni prostor korištenjem testa nesklada traži se skup nekonzistentnih uzoraka

2. *mjerenje udaljenosti između uzoraka* (eng. *distance-based approach*) – definira se mjera udaljenosti d između uzorka u iz skupa uzoraka U u odnosu na ostale uzorke iz tog skupa; ako je uzorak u udaljen barem d od barem p-tog dijela uzoraka iz skupa U, tada se takav uzorak smatra ekstremom

3. *mjerenje odstupanja uzoraka od cjeline* (eng. *deviation-based approach*) – ekstremom se smatra onaj uzorak koji ne odgovara glavnim osobinama ostalih uzoraka iz skupine, odnosno kaže se da je uzorak *devijantan* u odnosu na ostale uzorke iz skupine

2.2. Integracija podataka

Integracija podataka (eng. *data integration*) se provodi kada podaci dolaze iz više različitih izvora, te ih je potrebno integrirati u jedan izvor i, ako je moguće, reducirati. Pri tome se javlja problem različite definicije podataka u različitim izvorima. Na primjer, potrebno je prepoznati isti atribut koji se u različitim izvorima različito zove ili ima drugačije vrijednosti (vrijednost atributa *spol* u jednoj aplikaciji može biti *M* i *Ž*, a u drugoj *0* i *1*) ili atribut koji u različitim izvorima ima isti naziv, ali različito značenje (problem homonima i sinonima [6]).

Postupci koji se koriste kod integracije podataka [5]:

1. *shematska integracija* (eng. *schema integration*) – integriraju se sheme različitih izvora podataka, a najveći problem pri toj integraciji je identifikacija istih entiteta u različitim izvorima podataka (npr. ako se isti entiteti ili atributi razlikuju po nazivima)

2. *otkrivanje duplikata* (eng. *duplicate tuples*) – prepoznavanje istih n-torki koje se pojavljuju u različitim izvorima, kako bi ih se u integrirani sustav unijelo samo jedanput

3. *uočavanje i razrješavanje konfliktnih situacija* (eng. *detection and resolution of data value conflicts*) – uočavanje n-torki koje opisuju isti entitet u različitim sustavima, ali se vrijednosti atributa tog entiteta razlikuju u pojedinim izvorima

4. *zalihost* (eng. *redundancy*) – ako se uoči visoka korelacije između dva atributa koji opisuju jedan entitet, tada je dovoljno zadržati samo jedan atribut, jer je drugi atribut ovisan o njemu.
   Korelacija između dva atributa A i B se određuje *Pearsonovom korelacijom*: $r_{A,B} = \frac{\sum_{i=1}^{n}(A_i - \overline{A})(B_i - \overline{B})}{(n-1)\sigma_A \sigma_B}$,
   gdje su srednje vrijednosti atributa A i B: $\overline{A} = \frac{\sum_{i=1}^{n} A}{n}$ i $\overline{B} = \frac{\sum_{i=1}^{n} B}{n}$,
   te njihove standardne devijacije: $\sigma_A = \sqrt{\frac{\sum_{i=1}^{n}(A - \overline{A})^2}{n-1}}$ i $\sigma_B = \sqrt{\frac{\sum_{i=1}^{n}(B - \overline{B})^2}{n-1}}$

5. *redukcija podataka* (eng. *data reduction*) – uključuje postupak agregacije i generalizacija podataka (primjenom hijerarhijskog koncepta niži koncepti se zamjenjuju s višim konceptom)

2.3. Transformacija podataka

Posljednji korak pripreme podataka je transformacija podataka u željeni oblik – npr. numeričke vrijednosti se mogu normirati na određeni interval (najčešće su to intervali od 0.0 do 1.0 i od -1.0 do 1.0) ili se numerički atributi mogu pretvoriti u binarne atribute. Transformacija podataka uključuje i konstrukciju novih atributa, npr. iz datuma rođenja osobe može se odrediti dobna skupina osobe.

Osnovne metode transformacije podataka [5]:

1. *izjednačavanje uzoraka* – vrijednosti pojedinih atributa se izjednačuju kako bi se uklonili podaci sa šumom (eng. *smoothing*)

2. *agregacija podataka* (eng. *aggregation*) – ako se podaci prikupljaju u skladištu podataka, onda se agregiraju kako bi se lakše konstruirale kocke

3. *generalizacija* – podaci koji se nalaze na nižem konceptualnom nivou generaliziraju se na viši konceptualni nivo (npr. šifre gradova se mogu zamijeniti sa šiframa županija)

4. *konstrukcija novih atributa* – novi atributi se stvaraju na temelju poznatog skupa atributa (npr. u analizi se atributi *visina* i *dužina* mogu zamijeniti s atributom *površina*)

5. *redukcija broja atributa* ili *smanjenje dimenzijske složenosti* – atributi koji su irelevantni za postupak analize se izbacuju iz skupa atributa, kako bi se s manjim skupom atributa ubrzala i poboljšala analiza što je detaljno opisano u 3. poglavlju

6. *normiranje atributa* na jednake intervale (npr. svi atributi se skaliraju na intervale od -1.0 do 1.0 ili od 0.0 do 1.0):
   a. *min-max normiranje* (vrijednost atributa se linearno skalira prema minimalnoj i maksimalnoj vrijednosti koju atribut poprima):
   $$v' = \frac{v - min_A}{max_A - min_A}(novi\_max_A - novi\_min_A) + novi\_min_A,$$
   gdje je v vrijednost atributa A koja se normalizira, a v' je normalizirana vrijednost; min_A i max_A su minimalna i maksimalna vrijednost koju atribut A poprima, $novi\_min_A$ i $novi\_max_A$ minimalna i maksimalna vrijednost normaliziranog intervala

   b. *normiranje prema nuli kao srednjoj vrijednosti* (eng. *zero mean normalization*):
   $$v' = \frac{v - \overline{A}}{\sigma_A},$$
   gdje je v vrijednost atributa A koja se normalizira, a v' je normalizirana vrijednost; $\overline{A}$ je srednja vrijednost atributa A, a σ_A standardna devijacija atributa A

   c. *normiranje decimalnim skaliranjem* (eng. *normalization by decimal scaling*):
   $$v' = \frac{v}{10^j},$$
   gdje je v vrijednost atributa A koja se normira, a v' je normalizirana vrijednost; a j je najmanji cijeli broj takav da vrijedi $Max(|v'|) < 1$

3. Metode smanjivanja dimenzijske složenosti

Neka su podaci prikazani kao skup uzoraka (zapisa ili n-torki) koji je opisan skupom atributa (obilježja ili dimenzija). Svaki je uzorak karakteriziran vrijednostima koje poprima za pojedine atribute iz skupa atributa, te razredom (eng. *class*) kojemu pripada.

Smanjivanje broja atributa (eng. *feature selection*) je postupak uklanjanja irelevantnih ili redundantnih atributa (obilježja) iz skupa atributa. Time se smanjuje broj podataka koje je potrebno pretraživati kako bi se korištenjem određenog algoritma učenja za uzorak iz zadanog skupa odredio razred kojemu uzorak pripada. Smanjen broj atributa ubrzava postupak određivanja pripadnosti pojedinom razredu, te povećava preglednost i olakšava razumijevanje pravila.

Za smanjenje broja atributa koriste se *metode odabira podskupa atributa* (eng. *attribute subset selection*). Odabrani podskup atributa treba što više ogledavati distribuciju vjerojatnosti pojedinih razreda kakva se dobiva korištenjem svih atributa nad zadanim skupom uzoraka.

Algoritam za odabir podskupa atributa (eng. *feature selection algorithm* - FSA) služi za odabir optimalnog podskupa atributa. Pokazano je da algoritam učenja koji umjesto cijelog skupa atributa koristi dobro odabrani podskup atributa, daje točnije rezultate od algoritma učenja koji koristi cijeli skup atributa [14], [15], [16], [17], [20]. Osim točnosti, povećava se i brzina dobivanja rezultata, jer algoritam učenja djeluje na manjem skupu atributa, a manji skup atributa pridonosi jednostavnosti i lakšem razumijevanju dobivenih pravila.

Potrebno je razlučiti postupak ekstrakcije atributa (eng. *feature extraction*) od postupka odabira podskupa atributa (eng. *feature selection*) [18]. *Metode ekstrakcije atributa* stvaraju (ekstrahiraju) nove atribute transformacijom ili kombinacijom postojećih atributa. *Metode odabira podskupa atributa* na temelju zadanog skupa atributa odabiru najbolji podskup. Često ekstrakcija atributa prethodi odabiru podskupa relevantnih atributa, a neki od ekstrahiranih atributa se odbacuju zbog malog utjecaja na razvrstavanje uzoraka u razrede.

3.1. Odabir optimalnog podskupa atributa kod vrlo velikog skupa atributa za biološke podatke

Biološka empirička procjena je da svaki gen može imati jedno od dva stanja: ili je gen uključen, tj. aktivan u nekom procesu ili gen nije uključen u neki proces. U skladu s tim, razred gena može biti *aktivan* ili *neaktivan*, ovisno o aktivnosti gena u procesu. Ako neki atribut, kojim je gen opisan, ne pruža diskriminatornu razliku između dva moguća razreda klasifikacije, tada taj atribut nije bitan za klasifikaciju. Karakteristika bioloških podataka je mnoštvo atributa kojima su opisani uzorci (geni). Broj atributa može biti sumjerljiv, pa i veći od broja uzoraka (npr. skup uzoraka od nekoliko stotina uzoraka može biti opisan i s nekoliko tisuća atributa).

Bolji se rezultati o razvrstavanju uzoraka u pojedine razrede dobivaju kada algoritam učenja djeluje nad smanjenim skupom atributa, nego za slučaj kada algoritam učenja djeluje nad cijelim skupom atributa [14], [15], [16], [17], [20]. To se posebno

pokazalo za slučajeve kada je skup uzoraka puno manji od broja atributa što je slučaj s podacima prikupljenim genetskim istraživanjima .

Paradoksalna pojava koja se naziva *fenomen postizanja maksimuma* (eng. *peaking phenomenon*) je pojava kada reducirani skup atributa (od desetak ili nekoliko desetaka atributa) gotovo uvijek daje bolje rezultate pri razvrstavanju uzoraka, nego cijeli skup atributa (koji može brojati i nekoliko tisuća atributa). Osim što su rezultati gotovo uvijek lošiji kada se algoritam učenja primjenjuje nad cijelim skupom atributa, vrijeme potrebno da algoritam za učenje dođe do rješenja puno je dulje, nego kada algoritam učenja djeluje nad malim brojem atributa. Objašnjenje pojave je sljedeće [16]: najčešće korišteni klasifikatori (npr. stabla odlučivanja) rade tako da procjenjuju nepoznate parametre i zamjenjuju ih s vrijednostima dobivenima prema gustoći razreda. Za fiksan broj uzoraka u skupu uzoraka, a s istovremenim povećanjem broj atributa (a s time i broja nepoznatih parametara koje treba procijeniti na temelju uzorka), pouzdanost procjene parametara se smanjuje. Uzorci mogu sadržavati šum ili atributi mogu postati irelevantni za klasifikaciju (razvrstavanje).

U praktičnom dijelu ovog rada pokazalo se da je neizvedivo u realnom vremenu dobiti rezultate klasifikacije, osim ako se cijeli skup od nekoliko tisuća atributa ne smanji na najviše nekoliko stotina. Isto tako je uočeno da osim dobitka na brzini, klasifikacija s manjim brojem atributa daje bolje rezultate razvrstavanja, odnosno vrijedi fenomen postizanja maksimuma.

3.2. Definicija odabira optimalnog podskupa atributa

Definicija odabira optimalnog podskupa atributa mogu se podijeliti u sljedeće kategorije [19]:

1. *Idealizirana definicija*: pronaći dovoljan i nužan minimalan podskup atributa koji ispunjava ciljani koncept [25]

2. *Klasična definicija*: pronaći M atributa iz zadanog skupa atributa s N članova takav da je $M < N$ i vrijednost funkcije koja predstavlja kriterij je optimirana preko svih podskupova veličine M [21]

3. *Poboljšana točnost predviđanja*: cilj odabira podskupa atributa je izabrati one atribute koji poboljšavaju točnost predviđanja ili smanjuju veličinu strukture tako da je neznačajna razlika u sposobnosti predviđanja između suženog i nesuženog skupa [22]

4. *Približavanje originalnoj distribuciji uzoraka po razredima*: cilj odabira podskupa atributa je odabrati malen podskup takav da distribucija po razredima prema atributima iz podskupa bude što bliža distribuciji po atributima iz cijelog skupa [22].

3.2.1. Definicija odabira podskupa atributa s obzirom na evaluacijsku mjeru

Neka je A skup atributa nad zadanim skupom uzoraka U, a A' odabrani podskup atributa: $A' \subseteq A$. Neka je $J(A')$ *evaluacijska mjera* koja mjeri koliko je dobar odabrani podskup atributa. Određivanje evaluacijske mjere detaljnije je objašnjeno u poglavlju 3.5.

Evaluacijska mjera $J(A')$ je definirana kao:

$J: A' \subseteq A \rightarrow \Re$

Cilj je pronaći optimum (obično maksimum) evaluacijske mjere.

Odabir podskupa atributa može se definirati na sljedeće načine [17], [21]:

1. Neka je $|A'| < |A|$. Potrebno je pronaći $A' \subset A$ takav da je $J(A')$ maksimalna.

2. Neka je J_0 minimalna dopuštena vrijednost mjere J. Potrebno je pronaći $A' \subseteq A$ gdje je $|A'|$ minimalan uz uvjet $J(A') > J_0$.

3. Pronaći kompromis između minimiziranja $|A'|$ i maksimiziranja $J(A')$.

3.3. Općeniti algoritam za određivanje optimalnog podskupa atributa

Neka je U skup uzoraka, a A skup atributa nad U. Neka je $|A| = n$. Neka je J evaluacijska mjera kojom se mjeri uspješnost odabranog podskupa A', gdje J treba biti maksimizirana.

Neka je L lista odabranih podskupova atributa iz skupa A, gdje svaki podskup u listi ima određenu težinu prema evaluacijskih mjeri J. Ovisno o algoritmu koji se primjenjuje, lista može sadržavati jedan ili više podskupova koji su svi optimalno rješenje, tj. $|L| \geq 1$.

Neka *Rješenje* predstavlja najbolje rješenje pronađeno u pojedinoj iteraciji algoritma.

Na slici 3.1.prikazan je općeniti algoritam za odabir optimalnog podskupa atributa [17]. Funkcija `Odredi_početni_skup` na temelju zadanog skupa A, odabire početni podskup. Ovisno o algoritmu, to može biti prazan skup, cijeli skup A ili jedan ili više podskupova A odabranih prema nekom kriteriju. Operator *GS* je generator novih stanja, odnosno dodaje ili oduzima atribute za novi podskup atributa.

```
Ulaz:
   U – skup uzoraka sa skupom atributa A, |A| = n
   J – evaluacijska mjera koju treba maksimizirati
   GS – operator koji generira novi skup stanja
Izlaz:
   Rješenje – optimalni podskup atributa (svaki podskup ima određenu
              težinu)

Postupak:
   L := Odredi_početni_skup(A); // lista odabranih podskupova
   Rješenje := {najbolji podskup iz L prema J}
   ponavljaj
      L := Strategija_potrage (L, GS(J), A);
      A' := {najbolji podskup iz L prema J};
      ako je J(A') ≥ J(Rješenje) ili
            (J(A') = J(Rješenje) i |A'| < |Rješenje|)
      onda Rješenje := A';
```

```
dok nije Završi(J, L)
```

Slika 3.1. Općeniti algoritam za odabir optimalnog podskupa atributa

3.3.1. Definicija relevantnosti atributa

Neka je zadan algoritam učenja (klasifikator) C i skup uzoraka U s atributima A_1, A_2, ..., A_n s distribucijom D nad skupom uzoraka. *Optimalni podskup atributa* A' je podskup skupa svih atributa za koji vrijedi da je točnost klasifikatora $K = C(D)$ maksimalna.

Optimalni podskup ne mora biti jedinstven za zadani skup atributa zato što je moguće postići jednaku točnost klasifikatora za različite optimalne podskupove.

Definicija jako relevantnog atributa

Neka je Y oznaka razreda kojemu pripada uzorak. Neka je S_i skup atributa koji sadrži sve atribute osim A_i, tj. $S_i = \{A_1, A_2, ..., A_{i-1}, A_{i+1}, ..., A_m\}$.

Atribut A_i je **jako relevantan** ako i samo ako postoje neki a_i, y i s_i za koje je $p(A_i = a_i, S_i = s_i) > 0$ takvi da vrijedi:

$$p(Y = y \mid A_i = a_i, S_i = s_i) \neq p\,(Y = y \mid S_i = s_i)$$

Definicija slabo relevantnog atributa

Neka je Y oznaka razreda kojemu pripada uzorak. Neka je S_i skup atributa koji sadrži sve atribute osim A_i, tj. $S_i = \{A_1, A_2, ..., A_{i-1}, A_{i+1}, ..., A_m\}$.

Atribut A_i je **slabo relevantan** ako i samo ako nije jako relevantan i postoji podskup atributa S_i' ($S_i' \subset S_i$) za koje postoje neki a_i, y i s_i' za koje $p\,(A_i = a_i, S_i' = s_i') > 0$ takvi da vrijedi:

$$p(Y = y \mid A_i = a_i, S_i' = s_i') \neq p\,(Y = y \mid S_i' = s_i')$$

Definicija relevantnog atributa

Atribut X je **relevantan** ako je ili jako ili slabo relevantan.

Definicija irelevantnog atributa

Atribut X je **irelevantan** ako nije niti jako niti slabo relevantan.

3.3.2. Relevantnost atributa i optimalni podskup atributa

Bayesov optimalni klasifikator prema svojoj definiciji mora koristiti sve jako relevantne atribute i možda neke slabo relevantne atribute. Drugi algoritmi razvrstavanja, koji se koriste u praksi, mogu dati bolje rezultate uklanjanjem nekih atributa, pa čak i ako su to jako relevantni atributi.

Općenito vrijedi:

(i) relevantni atribut ne mora biti sadržan u optimalnom podskupu atributa

(ii) irelevantni atribut može biti sadržan u optimalnom podskupu atributa

Međutim, praksa je pokazala da su rijetki slučajevi kada su irelevantni atributi uključeni u optimalni podskup atributa.

Primjer (relevantnost atributa ne povlači da je atribut sadržan u optimalnom podskupu atributa):

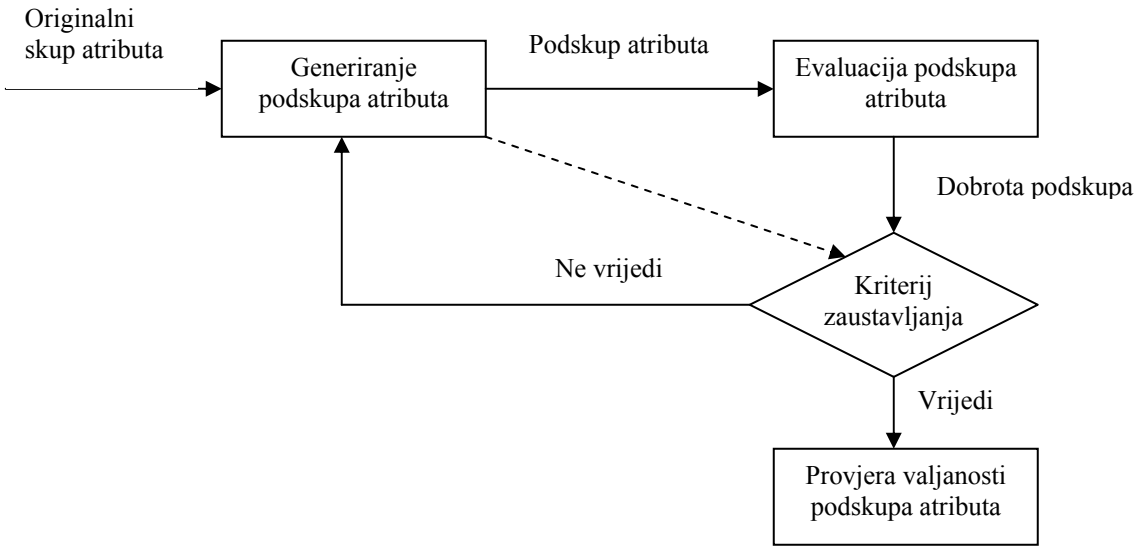
Neka je prostor mogućih uzoraka $\{0, 1\}^3$, tj. prostor čine tri logička atributa A_1, A_2, A_3, gdje svaki može poprimiti vrijednost 0 ili 1. Neka je u prostoru jednolična distribucija i neka je zadana relacija koja opisuje razred kojem pripada uzorak:

$$f(A_1, A_2, A_3) = (A_1 \wedge A_2) \vee A_3$$

Prema definiciji relevantnosti atributa, svi atributi u izrazu su relevantni, međutim ako prostor hipoteza čini konjunkcija relevantnih atributa, tada je optimalni podskup: $\{A_3\}$, a točnost predviđanja je 87.5%, što je najviša točnost unutar zadanog prostora hipoteza.

3.3.3. Osnovni koraci u odabiru podskupa relevantnih atributa

Na slici 3.2. prikazan je dijagram postupka odabira optimalnog podskupa atributa iz originalnog skupa atributa [8].



Slika 3.2. Postupak odabira optimalnog podskupa atributa

Osnovni koraci u postupku odabira relevantnih atributa [19]:

1. *generiranje podskupa atributa*
   Na temelju trenutnog skupa atributa generira se podskup atributa korištenjem algoritma za traženje optimalnog podskupa atributa.
   Početni skup atributa može biti:
   (i) prazan skup
   (ii) cijeli originalni skup
   (iii)slučajni podskup originalnog podskupa.
   Za slučajeve (i) i (ii) u svakom koraku pretraživanja atributi se dodaju, odnosno uklanjaju iz trenutnog podskupa atributa, a u slučaju (iii) atributi se mogu dodavati ili uklanjati iz trenutnog podskupa ili se podskup za svaku novu iteraciju generira ponovnim slučajnim odabirom.

2. *evaluacija podskupa atributa*

Evaluacija podskupa atributa se sastoji u mjerenju dobrote generiranog podskupa pri čemu se koristi evaluacijska funkcija. Dobivena mjera dobrote se uspoređuje s dosadašnjim najboljim rezultatom, pa ako je za novogenerirani podskup mjera dobrote veća od dotadašnje najbolje, nova mjera dobrote postaje najbolji novi rezultat.

3. *provjera kriterija zaustavljanja*
   Provjera da li je ispunjen kriterij zaustavljanja se koristi kako postupak odabira optimalnog podskupa atributa ne bi trajao beskonačno ili do iscrpljenja računalnih resursa. Kriterij zaustavljanja može ovisiti o načinu generiranja podskupa atributa ili načinu evaluacije generiranog podskupa.

   Kriterij zaustavljanja koji ovisi o *načinu generiranja podskupa atributa* može se podijeliti u dvije grupe:
   (i) da li je odabran zadani broj atributa
   (ii) da li je obavljen zadani broj iteracija

   Kriterij zaustavljanja temeljen na *evaluaciji podskupa atributa* može biti:
   (i) da li dodavanje (ili uklanjanje) atributa iz generiranog podskupa daje bolje rezultate, nego sam podskup
   (ii) da li je, prema korištenoj evaluacijskoj funkciji, dobiven optimalan podskup atributa

4. *provjera valjanosti podskupa atributa*

Provjera valjanosti podskupa atributa nije dio samog odabira podskupa atributa, ali služi kako bi se korištenjem različitih testova provjerila ispravnost dobivenog podskupa (npr. usporedba s rezultatima dobivenim drugim metodama ili, kod umjetnih skupova uzoraka, usporedba sa zadanim optimalnim podskupom koji je poznat unaprijed).

3.4. Metode generiranje podskupa atributa

Neka je N broj atributa u početnom skupu atributa. U tom slučaju broj podskupova koji su kandidati za optimalan podskup je 2^N, što je vrlo velik prostor pretraživanja čak i za manje N.
Generiranje podskupa atributa se može razmatrati prema dva glavna kriterija:
(i) načinu pretraživanja skupa atributa
(ii) da li je algoritam učenja uključen u način pretraživanja skupa atributa

3.4.1. Podjela metoda prema načinu pretraživanja skupa atributa

Tri su osnovna pristupa pretraživanja prostora podskupova skupa atributa [19], [23]:

1. *cjelovito pretraživanje* (eng. *complete search*) – pretraživanje cijelog skupa atributa kako bi se prema zadanoj evaluacijskoj funkciji odredio najbolji podskup skupa atributa. U [24] se navodi kako cjelovita pretraga (eng. *complete search*) skupa podataka ne mora ujedno biti i iscrpljujuća (eng. *exhaustive search*), ali je u oba slučaja vrijeme pretraživanja cijelog prostora podskupova zadanog skupa atributa $O(2^N)$.
   Da bi se smanjio prostor pretraživanja koriste se različite heurističke funkcije, a

optimalnost rješenja je garantirana korištenjem principa povratnog pretraživanja (eng. *backtracking*). Primjeri metoda koje uključuju povratno pretraživanje su: metoda grananja i ograničavanja (eng. *branch and bound*), metoda traženja prvog najboljeg (eng. *best first search*) i zrakasto pretraživanje (eng. *beam search*).

2. *heurističko pretraživanje* (eng. *heuristic search*) – primjenjuje se heuristička metoda kojom se *pohlepno* (eng. *greedy*) pretražuje skup atributa, a uvijek se gleda samo jedan korak unaprijed sve dok se ne postigne zadovoljavajuća performansa. U svakom se trenutku gleda samo najbolje rješenje za jedan korak unaprijed, a ne sveukupno najbolje rješenje (*kao loš igrač šaha, koji gleda samo jedan potez unaprijed* [6]). Ove metode su učinkovite u praksi i mogu dovesti blizu procjeni optimalnog rješenja. Vrijeme pretraživanja je $O(N^2)$ ili manje.
   Primjeri metoda koje koriste princip heurističkog pretraživanja: odabir najboljeg atributa za sljedeći korak (eng. *stepwise forward selection*), eliminacija najgoreg atributa u sljedećem koraku (eng. *stepwise backward elimination*), kombinacija odabira najboljeg i najgoreg atributa u sljedećem koraku (eng. *combination of forward selection and backward elimination*), *Relief* [25], itd.

3. *slučajno pretraživanje* (eng. *randomized search*) – koristi se metoda slučajnog generiranja podskupa. Iako je vrijeme pretraživanja cijelog prostora podskupova $O(2^N)$, pretražuje se samo onoliko podskupova koliko je unaprijed zadano. Optimalnost dobivenog podskupa ovisi o dostupnim resursima.
   Primjeri metoda koje koriste princip slučajnog generiranja podskupova: genetski algoritam, LVF [26], LVW [27].

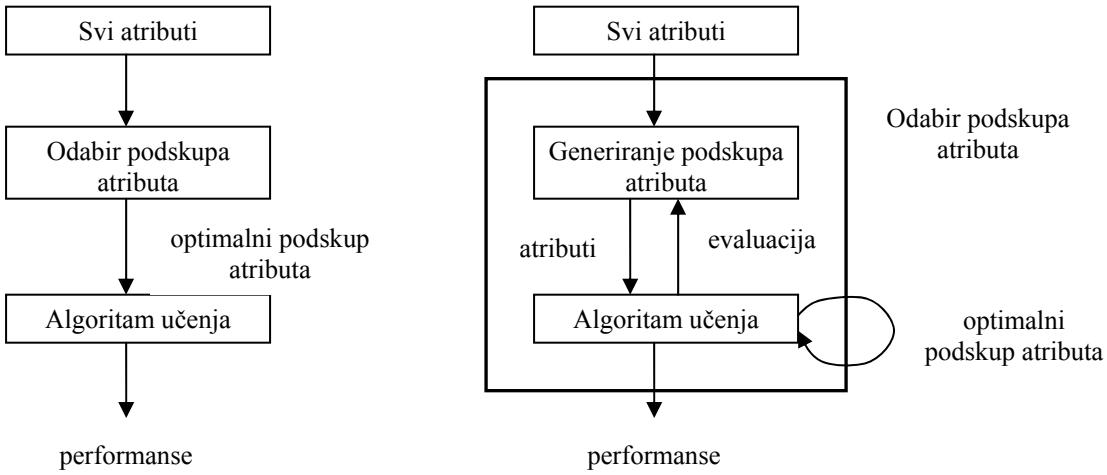
3.4.2. Podjela metoda prema načinu da li je algoritam učenja uključen u način pretraživanja skupa atributa

Podjela na tri osnovna pristupa pretraživanja skupa atributa ovisno o tome da li je algoritam učenja uključen u pretraživanje ili nije [23], [28]:

1. *metoda filtra* (eng. *filter approach*) – u ovom postupku odabir relevantnih atributa je odvojen od algoritma učenja (postupak odabira podskupa atributa je pretprocesiranje za sam algoritam učenja). Podskup relevantnih atributa se određuje korištenjem statističkih metoda koje obrađuju empiričku distribuciju uzoraka.
   Primjeri metoda filtra: LVF ([26]), Relief ([25]).

2. *metoda omotača* (eng. *wrapper approach*) – relevantni atributi se odabiru i evaluiraju samim algoritmom učenja.
   Neka su zadani klasifikator C i skup atributa A. Metoda omotača pretražuje prostor podskupova od A, a zatim se provjeravaju performanse klasifikatora C na svakom testiranom podskupu atributa (npr. može se koristiti metoda unakrsne provjere).
   Primjer je određivanje optimalnog podskupa atributa korištenjem stabla odlučivanja [5]: stablo odlučivanja se konstruira na temelju postojećeg skupa podataka, a svi atributi koji se pojave u stablu smatraju se relevantnim atributima, a svi atributi koji se ne pojave u stablu odlučivanja smatraju irelevantnim.
   Primjeri metode omotača: genetski algoritam – algoritam učenja koji u iterativnom procesu generira nove i bolje podskupove, LVW [27].

3. *kombinirani pristup* (eng. *hybrid of filter and wrapper approaches*) [15], [16], [28] – pristup koji koristi i filtarski i omotački pristup odabiru atributa, a koristi se kada je broj atributa vrlo velik. Kombinirani pristupi su opisani u [15], [28] za slučaj kada je zadan skup od 72 uzorka i 7129 atributa.
   Primjer kombiniranog pristupa (metoda se sastoji od tri koraka) [15]:
   a. pretprocesiranje kojim se sužava skup atributa od nekoliko tisuća atributa na skup atributa od nekoliko stotina atributa korištenjem metode slučajnih stabala (eng. *random forest method*)
   b. metoda filtra koja sužava podskup dobiven prvim korakom
   c. evaluacija koja se primjenjuje na algoritme učenja (npr. kNN, stabla odlučivanja, itd.).



Slika 3.3. Metoda filtra i metoda omotača

3.4.3. Usporedba metode filtra i metode omotača

Algoritmi temeljeni na metodi omotača pokazuju točnije rezultate od algoritama temeljenih na metodi filtra, ali vrijeme računanja je nekoliko redova veličine veće od filtarskih metoda. U pristupu problemima s velikim brojem atributa (nekoliko tisuća atributa), a to vrijedi i za biološke podatke, filtarske metode imaju veliku prednost zbog manjeg vremena potrebnog za računalnu obradu podataka, a u nekim slučajevima (kao što je slučaj za skup podataka korišten u praktičnom dijelu rada) u realnom vremenu metode omotača nisu ni izvodljive.

Drugi nedostatak metode omotača je evaluacija rezultata korištenjem metode unakrsne provjere (eng. *cross-validation*) koja može dovesti do povećane vjerojatnosti pronalaska podskupa atributa koji sasvim slučajno daje dobre rezultate na uzorku podataka. Općenito, u prostoru podskupova koji je izrazito velik, unakrsna provjera može dovesti do prevelikog prilagođavanja testnim podacima (eng. *overfitting*).

Najveći nedostatak metode filtra je potpuno ignoriranje učinka odabranog podskupa atributa na performanse algoritma učenja.

3.5. Evaluacijske funkcije

Evaluacijska funkcija služe kako bi se prema njoj odredila relevantnost, odnosno dobrota (eng. *goodness*) atributa. Evaluacijska funkcija mjeri sposobnost jednog atributa ili skupa atributa da se, ovisno o njihovim vrijednostima, uzorci podataka svrstaju u neki od razreda. Važno je napomenuti da je relevantnost atributa ovisna o odabranoj evaluacijskoj funkciji, a nije svojstvo samog atributa. Prema tome, korištenjem različitih evaluacijskih funkcija mogu se dobiti različiti optimalni podskupovi zadanog skupa atributa.
Neka je evaluacijska funkcija, koju treba maksimizirati, definirana s:

$J: A' \subseteq A \rightarrow \Re$,

gdje je A' podskup originalnog skupa atributa A. Skup A se sastoji od N atributa. Skup uzoraka je $\mathbf{X}$, gdje je svaki uzorak N-dimenzionalni vektor $\mathbf{x} \in \mathbf{X}$; $\mathbf{x} \in \Re^N$.
Uzorak može pripadati jednom od razreda iz skupa razreda $\Omega = \{\omega_1,..,\omega_m\}$ takav da vrijedi c: $\Re^N \rightarrow \Omega$. Funkcija c preslikava prostor uzoraka u skup razreda.

Prema vrijednostima koje mjere, evaluacijske funkcije se dijele u pet kategorija [19]:

1. mjera udaljenosti ili mjera divergencije (eng. *distance* i *divergence*)
2. mjera informacija ili mjera neizvjesnosti (eng. *information* i *uncertainty*)
3. mjera ovisnosti (eng. *dependence*)
4. mjera dosljednosti (eng. *consistency*)
5. mjera pogreške klasifikatora (eng. *classifier error rate*)

3.5.1. Mjera udaljenosti

Mjera udaljenosti (eng. *distance measure*) također se zove i *mjera odvojivosti* (eng. *separability*), *mjera divergencije* ili *mjera diskriminacije*. Za uzorke koji se mogu svrstati u jedan od dva razreda ω_1 i ω_2, za mjeru udaljenosti mora vrijediti:

1. atribut A_1 je relevantniji od atributa A_2, ako A_1 povlači veću razliku od atributa A_2 između uvjetne vjerojatnosti razreda ω_1 i uvjetne vjerojatnosti razreda ω_2
2. atributi A_1 i A_2 ne mogu razlikovati, ako je razlika između uvjetnih vjerojatnosti koje povlače atributi između razreda ω_1 i ω_2 jednaka 0
3. vrijednost mjere divergencije je veća, ako je odabrani podskup atributa bolji, odnosno bliži optimalnom rješenju

Mjera udaljenosti računa vjerojatnosnu udaljenost (divergenciju) između gustoća uvjetne vjerojatnosti za pripadnost uzorka $\mathbf{x}$ razredu ω_1 ($p(\mathbf{x}|\omega_1)$) i razredu ω_2 ($p(\mathbf{x}|\omega_2)$) [17]:

$$J = \int f\left[p(\mathbf{x} \mid \omega_2)p(\mathbf{x} \mid \omega_1)\right]d\mathbf{x}$$

Da bi funkcija J bila odgovarajuća mjera, funkcija f mora ispuniti uvjete tako da vrijedi:

1. $J \geq 0$
2. $J = 0$, ako su $p(\mathbf{x} \mid \omega_i)$ jednake

3. J je maksimalna, kada se $p(\mathbf{x} \mid \omega_i)$ ne preklapaju

Primjeri mjera udaljenosti:

a) Euklidova mjera udaljenosti: $f(a, b) = \sqrt{(a-b)^2}$

b) Chernoffova mjera udaljenosti: $f(a, b) = a^s b^{s-1}$, $s \in [0,1]$ i $J_{Che} = -\ln J$

c) Bhattacharyyeva mjera udaljenosti: $f(a, b) = \sqrt{ab}$ i $J_{Bha} = -\ln J$

d) Kullback-Lieblerova mjera udaljenosti: $f(a, b) = (a - b)(\ln a - \ln b)$ i $J_{KL} = J$

3.5.2. Mjera informacije

Mjera informacije (eng. *information measure*) određuje informacijsku dobit (eng. *information gain*) koju daje neki atribut. Možemo promatrati $\mathbf{x}$ i odrediti aposteriornu vjerojatnost $P(\mathbf{x} \mid \omega_i)$ da bi se vidjelo koliko je informacija prikupljeno o razredu kojemu pripada $\mathbf{x}$ uzimajući u obzir apriornu vjerojatnost. Ako su svi razredi jednako vjerojatni, tada je informacijska dobit minimalna, a entropija maksimalna.

Za mjeru informacije mora vrijediti: atribut A_1 je relevantniji od atributa A_2, ako je informacijska dobit od atributa A_1 veća od informacijske dobiti od atributa A_2.

Primjer mjere informacije:

Shannonova mjera entropije: $J_{Sha} = -\int p(\mathbf{x}) \sum_{i=1}^{m} P(\omega_i \mid \mathbf{x}) \log_2 P(\omega_i \mid \mathbf{x}) d\mathbf{x}$

3.5.3. Mjera ovisnosti

Mjera ovisnosti (eng. *dependence measure*) ili *mjera korelacije* (eng. *correlation measure*) je sposobnost da se na temelju vrijednosti jedne varijable odredi vrijednost druge varijable, a ovdje se koristi u smislu pronalaska korelacije između atributa A_1 i razreda ω_i.

Za mjeru ovisnosti mora vrijediti: atribut A_1 je relevantniji od atributa A_2, ako je atribut A_1 u većoj korelaciji s razredom ω_i od atributa A_2, odnosno ako atribut A_1 bolje predviđa razred ω_i od atributa A_2.

Također se može odrediti korelacija između atributa A_1 i ostalih atributa iz skupa atributa. Atribut A_1 je *redundantan*, ako postoji korelacija između atributa A_1 i ostalih atributa iz skupa atributa.

3.5.4. Mjera dosljednosti

Mjera dosljednosti (eng. *consistency measure*) traži minimalan skup atributa koji daje dovoljno veliku dosljednost na podskupu skupa uzoraka. Zahtijevanu preciznost zadaje korisnik. Razlikuje se od ostalih mjera po tome što se oslanja na pokazni uzorak podataka.

Neka je U skup svih uzoraka. Kažemo da je A' *nedosljedan* ako se u U nalaze dva uzorka koji su jednaki po svim atributima iz A', a pripadaju različitim razredima. Cilj je pronaći *minimalan* podskup originalnog skupa atributa čija je nedosljednost jednaka 0.

Neka je u_V broj uzoraka iz U koji su prema vrijednostima atributa iz A' jednaki uzorku $V \in U$. Neka je u_k broj uzoraka iz U koji su prema vrijednostima atributa iz A' jednaki uzorku V, a pripadaju razredu k.
Brojač nedosljednosti (eng. *inconsistency count*) za uzorak $V \in U$ je [52]:

$$BrojNedosljednosti(V) = u_V - \max_k u_k$$

Mjera nedosljednosti (eng. *inconsistency rate*) podskupa atributa A' na skupu uzoraka U je:

$$IR(A') = \sum_{V \in U} \frac{BrojNedosljednosti(V)}{|U|}$$

Monotonost mjere nedosljednosti je: $A_1 \subset A_2 \Rightarrow IR(A_1) \geq IR(A_2)$

Mjera dosljednosti je: $J(A') = \frac{1}{IR(A')+1}$, a poprima vrijednost iz intervala [0, 1].

3.5.5. Mjera pogreške klasifikatora

Mjera pogreške klasifikatora (eng. *classificator error measure*) je točnost koju odabrani podskup atributa daje za zadani klasifikator. S obzirom da se klasifikator koristi kao evaluacijska funkcija, metode koje koriste ovu evaluacijsku mjeru su metode omotača (eng. *wrapper methods*).

Klasifikator služi kao evaluacijsku funkcija za odabir relevantnih atributa, pa kada se u kasnijem postupku koristi razvrstavanje novih uzoraka prema tim relevantnim atributima, postiže se visoka točnost pri razvrstavanju. Nedostatak ovog pristupa je dugo vrijeme izvođenja.

Neka je klasifikator izgrađen korištenjem podskupom atributa A'; $A' \subset A$. Neka je U_{TE} skup testnih uzoraka, U_{TE}^+ skup testnih uzoraka za koje je klasifikator dao ispravne rezultate (korištenjem samo atributa iz A').

Pogreška klasifikatora se definira kao: $P_e = 1 - \frac{|U_{TE}^+|}{|U_{TE}|}$

Mjera pogreške klasifikatora se definira kao: $J = 1 - \hat{P}_e$

3.5.6. Usporedba evaluacijskih funkcija

Usporedba evaluacijskih funkcija prikazana je u tablici 3.1., a mjere su ocjenjivane na temelju tri kriterija:

1. *općenitost* – da li je dobiveni podskup atributa uporabljiv za različite klasifikatore (ne samo za jedan klasifikator)
2. *vremenska složenost* – kolika je vremenska zahtjevnost određivanja podskupa atributa prema zadanoj mjeri
3. *točnost* – kakva je točnost predviđanja testnih uzoraka za dobiveni podskup atributa

Tablica 3.1. Usporedba evaluacijskih funkcija

Evaluacijska funkcija	*Općenitost*	*Vremenska složenost*	*Točnost*
Mjera udaljenosti	da	niska	ovisi o podacima i klasifikatoru
Mjera informacije	da	niska	ovisi o podacima i klasifikatoru
Mjera ovisnosti	da	niska	ovisi o podacima i klasifikatoru
Mjera dosljednosti	da	srednja	ovisi o podacima i klasifikatoru
Mjera pogreške klasifikatora	ne	visoka	visoka

3.6. Način odabira podskupa atributa za svako sljedeće stanje

Nakon inicijalnog odabira podskupa atributa (A'), u svakom sljedećem koraku potrebno je prema nekom pravilu odrediti novi podskup atributa koji se zatim uspoređuje s do tada najboljim skupom atributa.
Operatori odabira podskupa atributa se mogu podijeliti u sljedećih pet kategorija [17]:

1. operator *unaprijed* (eng. *forward*)
2. operator *unatrag* (eng. *backward*)
3. *kombinirani unaprijed-unatrag* operator (eng. *compound*)
4. *težinski* operator (eng. *weighting*)
5. operator *slučajnosti* (eng. *random*)

Svi operatori djeluju tako da mijenjaju težinu t_i atributa A_i. Za težinski operator t_i je realni broj, a u svim ostalim slučajevima je $t_i \in \{0, 1\}$.

3.6.1. Operator unaprijed

Operator *unaprijed* dodaje novi atribut A_i u podskup A' tako da se njegovim dodavanjem najviše poveća J u odnosu na vrijednost $J(A')$ iz prethodnog koraka:

1. inicijalni podskup je $A' = 0$
2. svaki sljedeći podskup se dobiva kao:
   $A' := A' \cup \{A_i \in A \setminus A' \mid J(A' \cup \{A_i\})$ *donosi najveće povećanje u odnosu na* $J(A')\}$

Kriterij zaustavljanja može biti:

(i) $|A'| = n'$, ako je n' unaprijed definiran
(ii) vrijednost od J se nije povećala u zadnjih j koraka
(iii) vrijednost od J je veća od predefinirane vrijednosti J_0

Vrijeme izvođenja operatora je: O(N), gdje je N broj atributa u skupu atributa.
Najveći nedostatak ovog operatora je što ne uključuje mogućnost međudjelovanja atributa. Na primjer, ako su atributi A_1 i A_2 takvi da je J($\{A_1, A_2\}$) >> J($\{A_1\}$), J($\{A_2\}$), niti A_1 niti A_2 neće biti odabrani, iako zajedno pridonose poboljšanju evaluacijske mjere.

3.6.2. Operator unatrag

Operator *unatrag* uklanja atribut iz trenutnog podskupa A' tako da u svakom koraku evaluacijska mjera J bude bolja od one iz prethodnog koraka:

1. inicijalni podskup je $A' = A$, gdje je A skup svih atributa
2. svaki sljedeći podskup se dobiva kao:
   $A' := A' \setminus \{A_i \in A' \mid J(A' \setminus \{A_i\})$ *donosi najveće povećanje u odnosu na* $J(A')\}$

Kriterij zaustavljanja može biti:

(i) $|A'| = n'$, ako je n' unaprijed definiran

(ii) vrijednost od J se nije povećala u zadnjih j koraka

(iii)vrijednost od J je manja od predefinirane vrijednosti J_0

Vrijeme izvođenja operatora je: O(N), gdje je N broj atributa u skupu atributa. Međutim, računalno je vrijeme izvođenja puno dulje nego za operator *unaprijed*. Drugi problem je što i ovaj operator ne može otkriti sve interakcije među atributima, iako ih može otkriti više nego operator *unaprijed*.

3.6.3. Kombinirani unaprijed-unatrag operator

Kombinirani *unaprijed-unatrag* operator djeluje na sljedeći način: napravi se k koraka unaprijed (kao kod operatora *unaprijed*) i l koraka unatrag (kao kod operatora *unatrag*); ako je $k > l$, onda je rezultat korak unaprijed, a inače korak unatrag.

Kriterij zaustavljanja može biti:

(i) $k = l$.

Primjer: ako je $k = l = 1$, a dodan je atribut A_i i uklonjen atribut A_j, onda se obratna situacija može dogoditi u sljedećem koraku, ako kriterij zaustavljanja nije $k = l$.

(ii) $A_i = A_j$

U slijednom algoritmu za traženje podskupa atributa uvjet $k \neq l$ osigurava da će se provesti najviše N koraka, a vrijeme izvođenja je O(N^{k+l+1}).

3.6.4. Težinski operator

Težinski operator djeluje tako da se u konačnom rješenju nalaze svi atributi iz skupa A, gdje svaki atribut ima svoju težinu.

Sljedeće stanje se dobiva tako da se za svaki atribut izračunaju nove težine, što se najčešće dobiva uzorkovanjem zadanog skupa podataka.

3.6.5. Operator slučajnosti

Operator *slučajnosti* djeluje tako da u svakom koraku odabire novi slučajni podskup atributa koji može biti potpuno različit od podskupa iz prethodnog stanja.

Kriterij zaustavljanja može biti:

(i) $|A'| = n'$, ako je n' unaprijed definiran

(ii) vrijednost od J je veća od predefinirane vrijednosti J_0

3.7. Metoda koje se koriste za odabir optimalnog podskupa atributa za skupove podataka s velikim brojem atributa

3.7.1. Algoritam Relief

Algoritam *Relief* [17], [35] je filtarski algoritam koji odabire podskup atributa slučajnim putem, a u svakom koraku iteracije određuje se težina svakog odabranog atributa. Algoritam je motiviran algoritmom *k*-najbližih susjeda [60] i najbolje rezultate daje u kombinaciji s tim algoritmom učenja.

Princip rada algoritma je sljedeći: odabere se slučajni uzorak (V) iz skupa pokaznih uzoraka (U) i ažurira relevantnost atributa s obzirom na to koliko je odabrani uzorak udaljen od najbližeg uzorka iz iste (V_{ir}) i najbližeg uzorka iz drugog razreda (V_{dr}) (eng. *near-hit* i *near-miss*). Ova udaljenost se računa jer je atribut to relevantniji što više razdvaja uzorke V i V_{dr}, a manje razdvaja V i V_{ir}.

Loša strana algoritma je što ne radi dobro za redundantne atribute. Ako je većina atributa relevantna za koncept, odabrat će ih sve, iako je možda samo nekolicina dovoljna za opis koncepta.

Algoritam *Relief* radi samo za probleme binarne klasifikacije (problemi kada se uzorci mogu razvrstati samo u jedan od dva razreda), a njegovo poboljšanje *ReliefF* radi za više od dva razreda [47]. Algoritam je prikazan na slici 3.4.

```
Ulaz:
   U(A) – skup uzoraka U opisan skupom atributa A, |A| = n
   p – postotak uzoraka koje se uzima iz skupa uzoraka U
   d – mjera udaljenosti
Izlaz:
   T – polje težina atributa

Postupak:
   T[] = 0 // inicijalizacija težina svih atributa na 0
   ponavljaj p|U| puta
      V := Slučajni_Uzorak (U);
      V_ir := Najsličniji_uzorak_iz_istog_razreda (V, U);
      V_dr := Najsličniji_uzorak_iz_drugog_razreda (V, U);
      za i := 1 do n radi
         T[i] := T[i] + d_i(V, V_dr) – d_i(V, V_ir);
      kraj
   kraj
```

Slika 3.4. Algoritam *Relief*

3.7.2. Metoda LVF

Algoritam *LVF* (*Las Vegas Filter*) [17], [26] je filtarski algoritam koji u svakom koraku iteracije odabire novi slučajni podskup atributa A' i za novi podskup računa evaluacijsku mjeru $J(A')$. Ako novogenerirani podskup ima veći iznos evaluacijske mjere i manje atributa nego podskup s dotadašnjim najboljim rezultatom (*Najbolji*), novogenerirani podskup postaje najbolji podskup. Ako ima jednak broj atributa kao dotadašnji najbolji podskup i isti iznos evaluacijske mjere, onda se dodaje u listu jednako dobrih rješenja. Algoritam je prikazan na slici 3.5.

```
Ulaz
   U(A) – skup uzoraka U opisan skupom atributa A, |A| = n
   max – maksimalni broj iteracija
   J – evaluacijska mjera

Izlaz
   L – lista jednako dobrih rješenja (podskupova od A)

Postupak:
   L := []          // lista jednako dobrih rješenja
   Najbolji := A    // inicijalno najbolje rješenje
   J0 := J(A)       // najmanja dozvoljena vrijednost od J
   ponavljaj max puta
      A' := Slučajni_Podskup (A);
      ako je J(A') ≥ J0 onda
         ako je |A'| < |Najbolji| onda
            Najbolji := A'
            L := [A']
         inače ako je |A'| = |Najbolji| onda
            L := Dodaj (L, A')
   kraj
```

Slika 3.5. Algoritam *LVF*

U [26] se za maksimalni broj iteracija predlaže 77 * $|A|^5$ što je dobiveno iskustvenim putem kao kompromis između duljine trajanja algoritma i preciznosti rješenja (što algoritam dulje izvodi, rješenja su sve bolja). Ovisnost maksimalnog broja pokušaja o eksponencijalnoj vrijednosti broja atributa ($|A|$) u skladu je s empirički dobivenim zaključkom: što je veći broj atributa, to je teži problem dobivanja relevantnih atributa.

Nedostatak osnovnog algoritma LVF je nemogućnost rada s kontinuiranim vrijednostima. U slučaju pojave kontinuiranih vrijednosti, one se mogu diskretizirati primjenom odgovarajućeg algoritma ili se s kontinuiranim vrijednostima u posebnim slučajevima može postupati kao da su diskretne vrijednosti.

3.7.3. Metoda LVI

Algoritam LVI (*Las Vegas Incremental*) [14], [17] je filtarski algoritam koji se razlikuje od algoritma LVF po tome što za određivanje optimalnog podskupa atributa ne uzima cijeli skup uzoraka U, nego njegov podskup U_0. Optimalni podskup atributa A', koji se dobije na temelju U_0, provjerava se prema evaluacijskoj mjeri na cijelom skupu uzoraka U i ako je ispunjena minimalna vrijednost J_0, dobiveni podskup A' se smatra rješenjem.

Autori algoritma (H. Liu, R. Setiono) preporučaju da podskup uzoraka U_0 bude 10% cijelog skupa uzoraka U. Ako je U_0 inicijalno jako malen, u sljedećoj će iteraciji biti proširen s puno članova iz U_1 i postat će sličan samom skupu U, a ako je U_0 inicijalno velik, uštede računalnog vremena neće biti izražene. Algoritam je prikazan na slici 3.6.

```
Ulaz:
   U(A) – skup uzoraka U opisan skupom atributa A, |A| = n
   max – maksimalni broj iteracija
   J – evaluacijska mjera
```

```
   p - postotak
Izlaz:
   A' – pronađeno rješenje; podskup skupa A

Postupak:
   U0 = dio (U, p) // inicijalni podskup
   U1 = U \ U0     // testni skup
   J0 := J(A)      // najmanja dozvoljena vrijednost od J
   ponavljaj zauvijek
      A' := LVF (max, J, U0(A));
      ako je J(A') ≥ J0 onda zaustavi
      inače
         C := {članovi U1 s niskim doprinosom J korištenjem A'}
         U0 := U0 ∪ C
         U1 := U1 \ C
   kraj
```

Slika 3.6. Algoritam *LVI*

3.7.4. Algoritam SFG/SBG

Algoritmi *SFG* (eng. *Sequential Forward Generation*) i *SBG* (eng. *Sequential Backward Generation*) su oba slijedni algoritmi. U algoritmu *SFG* kreće se od A' koji je prazan skup, a kojem se u svakom koraku iteracije dodaje novi najbolji atribut iz skupa A. U algoritmu *SBG* se kreće od $A' = A$, a u svakom koraku iteracije uklanja se najlošiji atribut. *Najbolji* atribut za *SFG* je onaj čijim se dodavanjem evaluacijska funkcija J najviše poveća u odnosu na prethodni korak, a *najlošiji* atribut za SBG je onaj čijim se uklanjanjem evaluacijska funkcija J najviše poveća u odnosu na prethodni korak. Algoritam je prikazan na slici 3.7.

```
Ulaz:
   U(A) – skup uzoraka U opisan skupom atributa A
   J – evaluacijska mjera

Izlaz:
   A' – pronađeno rješenje; podskup skupa A

Postupak:
   A' = 0   // inicijalizacija za SFG
   //A' = A // inicijalizacija za SBG
   ponavljaj
      // SFG
      A0 := argmax{J(A' ∪ {A0}) | Ai ∈ A \ A'}
      A' := A' ∪ {A0}

      // SBG
      // A0 := argmax{J(A' \ {A0}) | Ai ∈ A'}
      // A' := A' \ {A0}

   dok više nema poboljšanja J u zadnjih j koraka
   // ILI za SFG: dok A' = A
   // ILI za SBG: dok A' = 0
```

Slika 3.7. Algoritam *SFG*/*SBG*

Algoritmi W-SFG i W-SBG (*W* za *wrapper*, odnosno za metodu omotača) koriste za određivanje točnosti vanjski algoritam učenja kao evaluacijsku mjeru.

3.7.5. Algoritam najbolji prvi

Algoritam traženja *najboljeg prvog* sljedećeg atributa (eng. *best first search*) [29], [52], [53] se sastoji se u pretraživanju prostora atributa kombinacijom tehnike pohlepnog odabira prvog sljedećeg atributa, ali se razlikuje od tog algoritma po tome što uključuje povratno pretraživanje (eng. *backtracking*). Ovaj algoritam koristi evaluacijsku mjeru *J* kojom se određuje koje je od generiranih rješenja najbolje. Algoritam može početi tako da je odabrani podskup atributa A' inicijalno prazan skup (kao na slici 3.8.), skup svih atributa ili bilo koji podskup skupa svih atributa. Pretraživanje se zaustavlja kada se generirani podskup ne poboljša u *k* zadanih koraka.

```
Ulaz:
   U(A) – skup uzoraka U opisan skupom atributa A
   J – evaluacijska mjera
   R – red koji sadrži generirane podskupove atributa
   k – broj koraka

Izlaz:
   A_najbolji – pronađeno rješenje; podskup skupa A

Postupak:
   A_najbolji = 0     // najbolji podskup
   A' = 0             // prazan skup atributa
   R = 0              // prazan red
   k' = 0             // broj koraka u kojima se nije promijenio A_najbolji

   Najbolji_prvi (U, J, R, k, A', A_najbolji)
      A_0 := argmax{J(A' ∪ {A_0}) | A_i ∈ A \ A'}
      A' := A' ∪ {A_0}
      Dodaj_u_red(A', R)
      ponavljaj zauvijek
         A'' = Skini_iz_reda(R)
         ako J(A'') > J(A_najbolji) onda
            A_najbolji = A''
         inače ako A_najbolji nije promijenjen u k koraka
            zaustavi
         Najbolji_prvi (U, J, R, k, A', A_najbolji)
      kraj
```

Slika 3.8. Algoritam traženja prvog najboljeg podskupa

4. Pregled osnovnih metoda dubinske analize podataka

Metode dubinske analize služe kako bi se izgradio model prema kojem će se za nepoznate podatke pokušati utvrditi nova informacija. Ako je cilj analize kategorizacija podataka po razredima, onda je nova informacija razred kojemu podaci pripadaju. Pri tome se algoritmi dijele u dvije osnovne skupine [7]: nadgledani algoritmi (eng. *supervised algorithms*) i nenadgledani algoritmi (eng. *unsupervised algorithms*).

Nadgledani algoritmi su oni u kojima se za izgradnju modela koriste podaci za koje su unaprijed poznati razredi kojima podaci pripadaju, a zatim se na temelju izgrađenog modela predviđa razred kojemu će pripadati nepoznati podaci. Ovoj skupini pripadaju metode razvrstavanja podataka (eng. *classification*) i metode regresije (eng. *regression*).

Nenadgledani algoritmi na temelju zadanih podataka formiraju razrede podataka, bez prethodnog saznanja o tome kojemu razredu bi podaci mogli pripadati. Ovoj skupini pripadaju metode grupiranja (eng. *clustering*) i asocijativnih pravila (eng. *association rules*).

4.1. Asocijativna pravila

Ova se metoda sastoji u otkrivanju i prepoznavanju *asocijativnih pravila* koji ukazuju na *učestalu* vezu između vrijednosti atributa i pojedinog atributa u zadanom skupu podataka. Ova se metoda često koristi u analizi potrošačke košarice (eng. *market basket analysis*) u transakcijskim sustavima, gdje se pokušava odgovoriti na pitanja oblika: *što će kupiti korisnik X, ako je kupio proizvode A i B?*
Općenito, asocijativno pravilo se zapisuje u obliku:

$$A \Rightarrow B \text{ ili } A_1 \wedge \cdots \wedge A_m \rightarrow B_1 \wedge \cdots \wedge B_n,\ A \cap B = \phi$$

gdje su A_i i B_j parovi *atribut-vrijednost atributa*.

S obzirom da neki transakcijski sustav koji prati ponašanje kupca može pronaći na tisuće, pa i milijune takvih pravila, definiraju se mjere koje će pokazati koliko je neko pravilo zanimljivo analitičaru. Takve mjere su podrška i pouzdanost pravila:

(i) *podrška pravila* (eng. *support*): $podrška(A \Rightarrow B) = P(A \cup B)$

Ako je podrška pravila *p%*, to znači da u skupu transakcija *p%* njih sadrži i *A* i *B*.

(ii) *pouzdanost pravila* (eng. *confidence*): $pouzdanost(A \Rightarrow B) = P(B \mid A)$,

Ako je pouzdanost pravila *r%*, to znači da u skupu transakcija *r%* onih transakcija koje sadrže *A*, sadrže i *B*.

Općenito, samo ona pravila koja ispunjavaju minimalno zadane prag podrške i prag pouzdanosti smatraju se zanimljivima. Takva se pravila nazivaju *jaka pravila*.

Primjer (asocijativnog pravila u analizi potrošačke košarice [5]):

$$dob(X,"30...39") \wedge prihod(X,"4K...8K") \Rightarrow kupuje(X,"računalo")$$

[*podrška* = 2%, *pouzdanost* = 60%]

Ovo se pravilo tumači na sljedeći način: 2% od ukupnog broja transakcija odnosi se na kupovinu računala od strane kupaca starosne dobi između 30 i 39 godina, čiji je prihod između 4000 i 8000. Pouzdanost od 60% znači da su kupci starosne dobi između 30 i 39 godina, čiji je prihod između 4000 i 8000, u 60% slučajeva kupili računalo.

4.1.1. Određivanje asocijativnih pravila

Neka je zadan skup stavki (eng. *itemset*) koji sadrži *k* stavki. Svaka stavka predstavlja vrijednost nekog atributa. Svaka se transakcija sastoji od skupa stavki. Primjer: skup stavki = {*"30...39"*, *"4K...8K"*, *"računalo"*} sadrži 3 stavke.

Učestalost (eng. *frequency*) pojavljivanja pojedine stavke je broj transakcija u kojima se stavka pojavljuje.

Stavka ispunjava *minimalnu brojnu podršku* (eng. *minimum support count*), ako je broj pojavljivanja stavke u transakcijama veći ili jednak zadanoj minimalnoj brojnoj podršci. Skup od *k* učestalih stavki se uobičajeno označava s L_k.

Algoritam pronalaženja jakih asocijativnih pravila:
1. *pronaći učestale podskupove stavki* (svaki se od ovih podskupova pojavljuje u cijelom skupu transakcija barem onoliko puta kolika je minimalna brojna podrška)
2. *generirati jaka asocijativna pravila iz skupa učestalih stavki* (ovako odabrana pravila ispunjavaju minimalnu brojnu podršku i minimalnu pouzdanost)

4.2. Regresija

Regresija (eng. *regression*) je metoda kojom se predviđa vrijednost numeričkih atributa. Na temelju poznatih vrijednosti atributa zadanih podataka određuju se parametri modela, a zatim se na temelju parametara modela određuju nepoznate vrijednosti atributa novih podataka.

Osnovni tipovi regresije [5], [6]:
1. linearna regresija (eng. *linear regression*)
2. višestruka regresija (eng. *multiple regression*)
3. logistička regresija (eng. *logistic regression*)
4. nelinearna regresija

4.2.1. Linearna regresija

Linearna regresija je najjednostavniji oblik regresije u kojoj se vrijednost zavisne slučajne varijable *Y* određuje kao linearna funkcija prediktivne varijable *X*:

$$Y = \alpha X + \beta ,$$

gdje se regresijski koeficijenti α i β određuje metodom najmanjih kvadrata.

Neka je zadan skup uzoraka *U*, gdje je svaki uzorak prikazan kao dvodimenzionalni vektor (x_i, y_i). Broj uzoraka je *u*.

Koeficijenti α i β određuju se kao:

$$\beta = \frac{\sum_{i=1}^{u}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{u}(x_i - \bar{x})^2}, \alpha = \bar{y} - \beta\bar{x}$$

Vrijednosti $\bar{x}$ i $\bar{y}$ su srednje vrijednosti: $\bar{x} = \frac{\sum_{i=1}^{u} x_i}{u}, \bar{y} = \frac{\sum_{i=1}^{u} y_i}{u}$.

4.2.2. Višestruka regresija

Višestruka regresija je proširenje linearne regresije. Za razliku od linearne regresije uključuje više od jedne prediktivne varijable. Zavisna varijabla *Y* se određuje kao linearna funkcija višedimenzionalnog vektora prediktivnih varijabli. Općeniti oblik višestruke regresije je:

$$Y = \alpha_1 X_1 + \cdots \alpha_n X_n + \beta,$$

gdje se parametri α_i i β određuju metodom najmanjih kvadrata.

4.2.3. Logistička regresija

Logistička regresija je proširenje linearne regresije uz ograničenje da interval vrijednosti koje može poprimiti zavisna varijabla *Y* može biti samo iz intervala [0, 1]. Logistička funkcija time modelira vjerojatnost kao linearnu funkciju prediktivne varijable, a općeniti oblik je:

$$Y = \frac{1}{1 + e^{-X}}.$$

4.2.4. Nelinearna regresija

Ako varijable u modelu ne pokazuju linearnu, nego polinomnu ovisnost, tada je polinomnu ovisnost potrebno svesti na linearan oblik i dalje rješavati kao linearnu regresiju.

Primjer:

Neka je zadana funkcija međuovisnosti varijabli:

$$Y = \alpha_1 X + \alpha_2 X^2 + \alpha_3 X^3 + \beta$$

Ovakva funkcija svodi se na linearni oblik, tako da se uvedu nove varijable:

$X_1 = X^1, X_2 = X^2, X_3 = X^3$, pa se zatim rješava jednadžba:

$$Y = \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \beta$$

4.3. Grupiranje

Grupiranje je metoda kojom se skup uzoraka (skup fizičkih ili apstraktnih objekata) svrstava u skupove međusobno što sličnijih podataka. Objekti unutar jedne grupacije (eng. *cluster*) moraju biti međusobno što sličniji i moraju se što više razlikovati od objekata izvan grupacije. Kako bi se utvrdila sličnost ili različitost između pojedinih objekata, potrebno je definirati mjere kojima će se mjeriti sličnost, odnosno različitost. Kao mjeru sličnosti uzima se udaljenost između pojedinih objekata.

Udaljenost mora biti tako definirana da za objekte koji su međusobno slični bude što bliža 0, a što veća za međusobno različite objekte.

Grupiranje objekata koristi se kod prepoznavanja uzoraka, procesiranja slika i marketinških analiza.

4.3.1. Mjere udaljenosti između pojedinih objekata

Neka je zadana *podatkovna matrica* (eng. *data matrix*) dimenzija *n* x *p*, gdje je *n* broj objekata ili uzoraka, a *p* broj varijabli ili atributa. Struktura podatkovne matrice odgovara strukturi relacijske tablice:

$$\begin{bmatrix} x_{11} \cdots x_{1p} \\ x_{21} \cdots x_{2p} \\ \vdots \\ x_{n1} \cdots x_{np} \end{bmatrix}$$

Definira se *matrica različitosti* (eng. *dissimilarity matrix*) dimenzija *n* x *n*, gdje je *n* broj objekata, a matrica za svaki element *d(i, j)* sadrži udaljenosti između *i*-tog i *j*-tog objekta izračunatu prema zadanoj mjeri udaljenosti. Vrijedi: *d(i, j) = d(j, i)* i *d(i, i) = 0*:

$$\begin{bmatrix} 0 & d(1,2) & \cdots & d(1,n) \\ d(2,1) & 0 & \cdots & d(2,n) \\ \cdots & & & \\ d(n,1) & d(n,2) & \cdots & 0 \end{bmatrix}$$

Svaka mjera udaljenosti koja se definira mora ispunjavati sljedeće uvjete [5]:

1. udaljenost mora biti nenegativan broj: $d(i, j) \geq 0$
2. udaljenost objekta od sebe samog mora biti 0: $d(i, i) = 0$
3. udaljenost mora biti simetrična: $d(i, j) = d(j, i)$
4. za udaljenost mora vrijediti nejednakost trokuta, tj. izravna udaljenost između objekta *i* i objekta *j* ne smije biti veća od udaljenosti od objekta *i* do objekta *j* preko objekta *h*: $d(i, j) \leq d(i, h) + d(h, j)$

Neka su objekti *i* i *j* prikazani kao *p*-dimenzionalni vektori:

$i = (x_{i1}, x_{i2}, ..., x_{ip})$ i $j = (x_{j1}, x_{j2}, ..., x_{jp})$

Primjeri mjera udaljenosti između objekta *i* i objekta *j*:

1. Euklidska udaljenost: $d(i, j) = \sqrt{\sum_{m=1}^{p} (x_{im} - x_{jm})^2}$
2. Euklidska težinska udaljenost: $d(i, j) = \sqrt{\sum_{m=1}^{p} w_m (x_{im} - x_{jm})^2}$,

   gdje je w_m težina *m*-te varijable (veća težina je pridijeljena važnijim varijablama)

3. Manhattan ili blokovska udaljenost: $d(i,j) = \sum_{m=1}^{p} \left| x_{im} - x_{jm} \right|$

4. Minkowskijeva udaljenost: $\sqrt[q]{\sum_{m=1}^{p} \left| x_{im} - x_{jm} \right|^{q}}$

Težinska udaljenost se može primijeniti i na Manhattan i Minkowskijevu udaljenost.

Ako varijable kojima su objekti opisani nisu numeričke nego binarne vrijednosti (0 ili 1), onda se udaljenost između objekta *i* i *j* određuje prema izrazu:

$$d(i,j) = \frac{r+s}{q+r+s+t},$$

gdje se parametri *q*, *r*, *s* i *t* očitaju iz tablice 4.1.
Primjer:
U tablici 4.1. vrijednost *q* predstavlja ukupan broj varijabli kojima je vrijednost 1 i za objekt *i* i za objekt *j*; vrijednost *r* predstavlja ukupan broj varijabli kojima je vrijednost 1 za objekt *i*, a 0 za objekt *j*, itd.

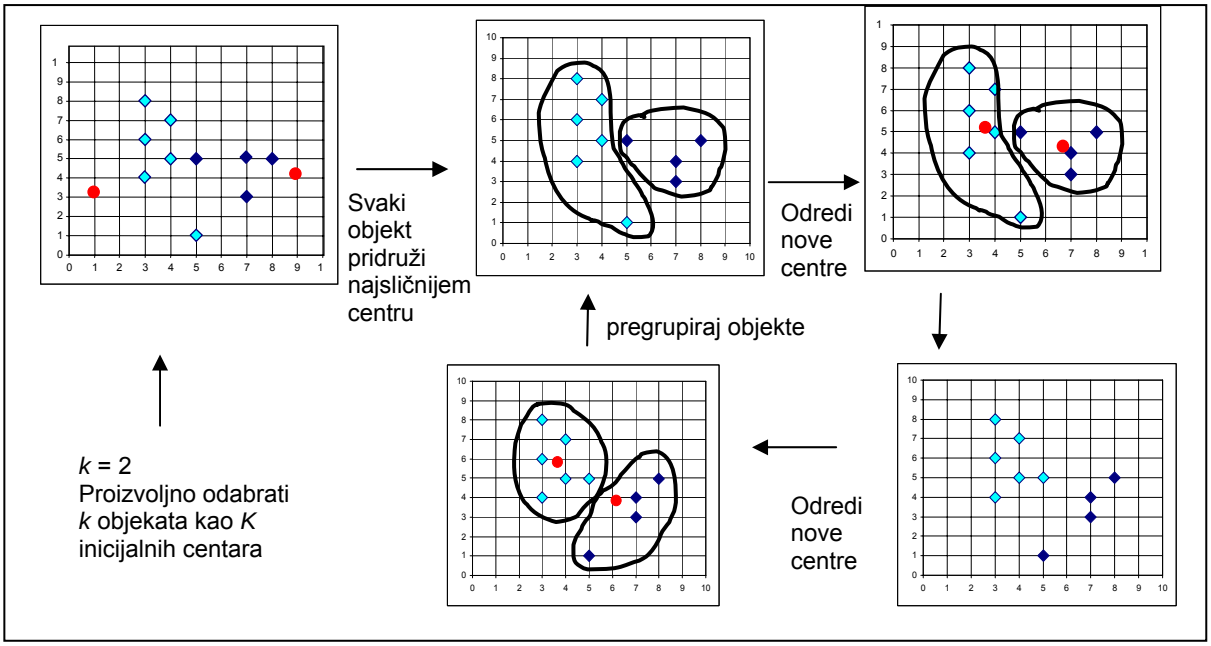
Tablica 4.1. Tablica za određivanje udaljenosti kod binarnih varijabli

		Objekt *j*		
		1	**0**	**Ukupno**
Objekt *i*	**1**	q	r	q + r
	0	s	t	s + t
	Ukupno	q + s	r + t	p

4.3.2. Metoda particioniranja

Neka je zadan skup od *n* objekata i *k* – broj grupacija u koje objekte treba grupirati. Grupacije se formiraju tako da bude ispunjen uvjet da su objekti unutar jedne grupacije međusobno što sličniji i da se što više razlikuju od objekata iz drugih grupacija.

Primjer: *Grupiranje prema k-srednjih vrijednosti*
Objekti ili uzorci se grupiraju oko *k* objekata koji predstavljaju srednju vrijednost unutar pojedine grupacije.
Princip rada prikazan je na slici 4.1.



Slika 4.1. Grupiranje oko *k* centara

Algoritam je prikazan na slici 4.2. [49]

```
Ulaz:
   k - broj grupacija
   U - skup uzoraka ili objekata
   n - broj objekata u skupu

Izlaz:
   Skup od k grupacija za koje je minimizirana pogreška mjere
   udaljenosti

Postupak:
   Proizvoljno odabrati k objekata kao k centara grupacija
   ponavljaj
      Pridruži svaki objekt onoj grupaciji kojoj je objekt najsličniji
      prema odabranoj mjeri udaljenosti (najmanja mu je udaljenost u
      odnosu na centar grupacije)
      Odredi novi centar grupacije kao srednju vrijednost svih objekata
      u grupaciji
   dok nema promjena u centrima grupacija
```

Slika 4.2. Algoritam grupiranja oko k srednjih vrijednosti

4.3.3. Nedostaci metoda grupiranja

Većina algoritama grupiranja pokazuje dobra svojstva za male skupove podataka, ali puno lošije rezultate za velike skupove podataka kakvi su danas tipično pohranjeni u bazama podataka. Objekti koji imaju na stotine ili tisuće atributa neprikladni su za grupiranje, jer većina algoritama grupiranja daje dobre rezultate samo za objekte koji imaju nekoliko atributa.

Mjere udaljenosti koje se često koriste, kao Euklidska ili blokovska udaljenost omogućuju pronalazak grupacija koje su otprilike iste veličine i gustoće i pravilnog

oblika, ali takve mjere udaljenosti nisu prikladne za pronalazak grupacije proizvoljnih oblika.

Nedostatak je i to što mnogi algoritmi grupiranja traže da korisnik unaprijed odredi koliko grupacija očekuje, što zahtijeva prethodno razumijevanje semantike podataka.

Neki algoritmi grupiranja osjetljivi su na šum ili ekstremne vrijednosti u podacima. Takvi algoritmi, zbog nemogućnosti prepoznavanja šuma, pogrešno formiraju grupe (grupe se preoblikuju kako bi obuhvatile i takve objekte, iako im oni ne pripadaju).

4.4. Razvrstavanje

Razvrstavanje ili klasifikacija je postupak pronalaska modela koji opisuje koncept podjele podataka po razredima kako bi se zatim na temelju tog modela predvidjela pripadnost objekata za koje je razred nepoznat.

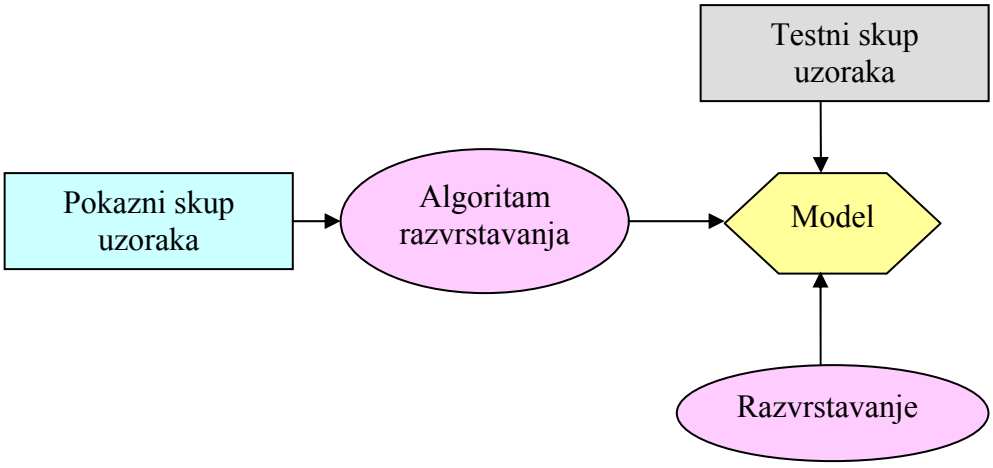
Kako bi se izgradio i testirao model razvrstavanja, podaci za koje je poznat razred podjele se u dva skupa:

1. *pokazni skup uzoraka* (eng. *training set*) - skup podataka koji se koristi za izgradnju modela
2. *testni skup uzoraka* (eng. *test set*) - skup podataka nad kojim se provjerava ispravnost modela; obično čini 20%-30% cijelog skupa podataka.

Nedostatak ovakve podjele skupa podataka u dva dijela je u tome što se podaci koji se koriste kao testni skup ne koriste za izgradnju modela. Međutim, nužno je izdvojiti dio podataka u testni skup kako bi se nad testnim skupom podataka ispitala točnost izgrađenog modela.

Točnost modela je postotak testnih uzoraka koji su točno klasificirani izgrađenim modelom. Kada bi se točnost modela određivala na samom pokaznom uzorku podataka, onda bi dobivena točnost bila preoptimistična (eng. *overfitting*), odnosno moglo bi se dogoditi da izgrađeni model uključuje neke anomalije koje su svojstvene samo pokaznom skupu podataka.

Ako se pokaže da je točnost razvrstavanja podataka iz testnog skupa podataka na izgrađenom modelu zadovoljavajuća, onda se dobiveni model može koristiti za razvrstavanje podataka kojima je razred nepoznat (slika 4.3.).



Slika 4.3. Proces razvrstavanja

U slučajevima kada je skup podataka s kojima treba izgraditi model malen, tada je bolje koristiti metodu *N-unakrsnih provjera* (eng. *N-fold cross-validation*), gdje se skup podataka podjeli u *N* disjunktnih podskupova približno jednake veličine (*N* je najčešće 10). Zatim se izgradi *N* modela, gdje se za izgradnju modela koriste svi podskupovi osim jednog, a točnost modela se zatim ispita na izostavljenom podskupu. Pogreška cijelog modela je prosječna pogreška za *N* ponavljanja. Za razvrstavanje podataka s nepoznatim razredom može se koristiti onaj model koji je imao najmanju pogrešku.

Metode razvrstavanja mogu se usporediti prema sljedećim kriterijima:

1. *točnost predviđanja* – sposobnost modela da točno predvidi razred za nepoznati uzorak
2. *brzina* – računalno vrijeme potrebno za izgradnju i testiranje modela
3. *robusnost* - sposobnost modela da točno predvidi razred za nepoznate uzorke koji sadrže šum ili atributi imaju nepoznate vrijednosti
4. *skalabilnost* – sposobnost efikasne izgradnje modela za velike količine podataka
5. *razumljivost* – razumljivost izgrađenog modela

Primjeri algoritama koji se koriste za razvrstavanje podataka:

1. stabla odlučivanja (eng. *decision trees*) [48], [5]
2. naivni Bayesov klasifikator (eng. *naive Bayesian classifier*) [60], [5]
3. prenošenje informacije unatrag (eng. *backpropagation*) - neuronske mreže
4. metoda *k*-najbližih susjeda [60], [5]
5. algoritam VFI (*Voting Feature Interval*) [8]
6. algoritam SVM (*Support Vector Machines*) [59]

Navedene algoritme razvrstavanja možemo podijeliti u dvije skupine:

1. *lijeni algoritmi učenja* (eng. *lazy learner*) - klasifikator samo pohranjuje pokazne uzorke i kao takav ne postoji sve do trenutka kada treba razvrstati novi nepoznati uzorak (npr. metoda *k*-najbližih susjeda)

2. *agilni algoritmi učenja* (eng. *eager learner*) - prvo se izgradi model na temelju pokaznih uzoraka, a zatim se novi uzorci razvrstavaju korištenjem tog modela

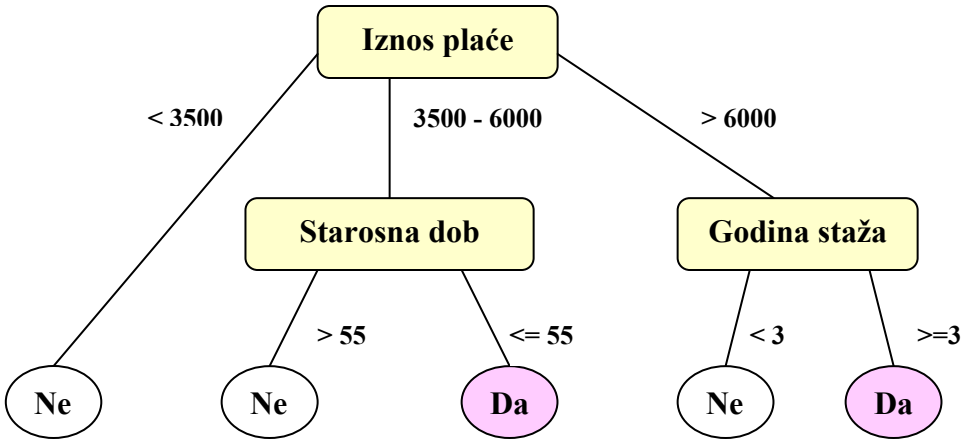
Lijeni algoritmi učenja su brži za stvaranje, ali puno sporiji kod razvrstavanja novih uzoraka, jer puno dulje traje postupak pronalaženja razreda u koji treba razvrstati novi uzorak.

4.4.1. Stabla odlučivanja

Stablo odlučivanja je dijagram toka predstavljen stablastom strukturom. U svakom *unutarnjem čvoru* ispituje se uvjet nad nekim atributom, a svaka *grana* stabla predstavlja jedan ishod ispitivanja. *List* stabla predstavlja razred ili distribuciju razreda. Metoda je opisana prema [5], [48].

Tipičan primjer stabla odlučivanja prikazan je na slici 4.3. gdje se određuje kreditna sposobnost građanina. Razredi u koje se mogu svrstati uzorci imaju oznake *Ne* i *Da*. Pripadnost uzorka razredu *Ne* znači da osoba koja se opisuje uzorkom nije kreditno sposobna, a pripadnost uzorka razredu *Da*, znači da je osoba kreditno sposobna. Da bi se neki uzorak razvrstao u jedan od ta dva razreda, potrebno je vrijednosti pojedinih atributa koji opisuju uzorak testirati kroz zadano stablo odlučivanja. Testiranje počinje u korijenu stabla (čvor *Iznos plaće*), a završava u listu *Ne* ili *Da*.

Važno je napomenuti da uzorak može biti opisan s više atributa, nego što ih se koristi za opisivanje stabla odlučivanja. Atributi koji su izostavljeni iz stabla odlučivanja su irelevantni ili slabo relevantni za određivanje pripadnosti uzorka pojedinom razredu (npr. ako se u primjeru sa slike 4.4. za opis uzorka koriste podaci kao što su ime i prezime osobe, matični broj osobe, boja kose i slično, a ti su atributi izostavljeni iz stabla odlučivanja, znači da su ti atributi irelevantni za određivanje kreditne sposobnosti).



Slika 4.4. Stablo odlučivanja kreditne sposobnosti građanina

Osnovni algoritam za stvaranje stabla odlučivanja prikazan je na slici 4.5., a predstavlja varijantu algoritma ID3 [48]. To je pohlepni algoritam koji djeluje po principu podijeli, pa vladaj.

```
Algoritam: Stvori_stablo_odlučivanja (U, A).
Ulaz:
   U - skup pokaznih uzoraka
   A - lista atributa s parovima atribut-diskretna vrijednost atributa

Izlaz: Stablo odlučivanja

Postupak:
   stvori čvor N
   ako su svi uzorci istog razreda R onda
      vrati N kao list s oznakom R

   ako je A prazna onda
      vrati N kao list s oznakom razreda koji je najčešći razred
      uzoraka iz U

   odaberi A0 ∈ A takav da ima najveću informacijsku dobit

   označi čvor N s A0

   za svaku vrijednost ai koju poprima A0
      stvori granu iz čvora N uz uvjet A0 = ai
      neka je Ui ⊂ U za koji je A0 = ai //particija
      ako je Ui prazan skup onda
         dodaj list s oznakom razreda najčešćeg u skupu U
      inače
         dodaj čvor koji vrati Stvori_stablo_odlučivanja(Ui, A \ A0)
   kraj
```

Slika 4.5. Osnovni algoritam za stvaranje stabla odlučivanja

Mjera *informacijske dobiti* koja se koristi u algoritmu služi za odabir najboljeg atributa u svakom čvoru stabla. Ova se mjera još zove i *mjera odabira atributa* ili *mjera dobrote podjele čvora*. Za odabir atributa u pojedinom čvoru uzima se onaj atribut koji ima najveću informacijsku dobit, odnosno onaj atribut koji ima najveću redukciju entropije. Atribut s najvećom informacijskom dobiti je onaj koji minimizira informaciju potrebnu da bi se uzorci razvrstali u particije (najmanja slučajnost pri podjeli uzoraka u particije).

4.4.2. Određivanje informacijske dobiti

Neka je U skup uzoraka za koje je poznat razred pripadanja. Neka je u ukupan broj uzoraka u skupu. Neka su razredi označeni kao R_i, za $i = 1,..., m$, gdje je m broj razreda. Uzorci koji pripadaju razredu R_i čine particiju skupa uzoraka U_i. Broj uzoraka u skupu U_i je u_i. Vjerojatnost da proizvoljan uzorak pripada razredu R_i je $\frac{u_i}{u}$.

Očekivana informacija potrebna za razvrstavanje uzoraka je:

$$I(u_1,..,u_m) = -\sum_{i=1}^{m} \frac{u_i}{u} \log_2 \frac{u_i}{u}$$

Neka atribut A poprima vrijednost iz skupa $\{a_1, ..., a_v\}$. Na temelju skupa vrijednosti koje atribut poprima, skup uzoraka se particionira na skup $\{U_1, .., U_v\}$, gdje je U_j skup

uzoraka za koje je vrijednost atributa A jednaka a_j. Broj uzoraka u_{ij} je broj uzoraka iz podskupa U_j koji pripadaju razredu R_i.

Entropija koja se dobije particioniranjem u podskupove prema vrijednostima atributa A je:

$$E(A) = \sum \frac{u_{1j} + \cdots + u_{mj}}{u} I(u_{ij}, \ldots, u_{mj}),$$

gdje $\frac{u_{1j} + \cdots + u_{mj}}{u}$ predstavlja težinu j-tog podskupa koja se dobije tako da se podijeli broj uzoraka koji su u particiji U_j u odnosu na ukupan broj uzoraka u. Što je entropija manja, to je veća čistoća particije.

Za podskup U_j očekivana informacija potrebna za razvrstavanje je:

$$I(u_{1j}, .., u_{mj}) = -\sum_{i=1}^{m} \frac{u_{ij}}{|U_j|} \log_2 \frac{u_{ij}}{|U_j|}$$

Informacijska dobit za atribut A je:

$$Dobit(A) = I(u_1, \ldots, u_m) - E(A)$$

4.4.2.1 Podrezivanje stabla

Nakon što se izgradi stablo, mnoge grane mogu reflektirati anomalije u pokaznom skupu uzoraka koji potječu od šuma u podacima ili ekstremnih vrijednosti u pojedinim uzorcima. Skraćivanje ili podrezivanje stabla (eng. *pruning tree*) služi kako bi se riješio problem pretjerane adaptiranosti stabla podacima iz pokaznog skupa uzoraka, a sastoji se u uklanjanju nepouzdanih grana u stablu. Podrezano stablo rezultira bržom i točnijom klasifikacijom novih uzoraka. Dvije su metode podrezivanja stabala: skraćivanje unaprijed ili predpodrezivanje (eng. *prepruning*) i naknadno skraćivanje ili podrezivanje (eng. *postpruning*) stabla.

Skraćivanje stabla unaprijed sastoji se u zaustavljanju daljnjih grananja stabla u određenom trenutku. U tom trenutku svaki čvor na toj razini postaje list, a razred kojemu će pripasti uzorci može biti razred kojemu pripada najviše uzoraka ili se može napraviti vjerojatnosna distribucija po razredima ovisno o razredima kojima pripadaju pojedini uzorci u tom čvoru. Trenutak zaustavljanja može biti određen nekom statističkom mjerom (npr. informacijska dobit) kada više niti jedan atribut ne donosi unaprijed zadanu minimalnu dobit. Problem je odabrati odgovarajuću minimalnu dobit – niska minimalna dobit može proizvesti presloženo i suviše adaptirano stablo, dok previsoka dobit može dati loše rezultate zbog prevelike jednostavnosti dobivenog stabla.

Druga je metoda *naknadno skraćivanje stabla*, a sastoji se u uklanjanju grana stabla u trenutku kada je stablo u potpunosti oblikovano. Čvor na razini koja je posljednja ostavljena nakon uklanjanja grana postaje list stabla, a razred kojemu će pripadati uzorci u tom čvoru postaje najučestaliji razred. Grane koje je potrebno odrezati određuju se tako da se odredi pogreška u odnosu na slučaj kada je ta grana zadržana u stablu. Ako uklanjanje grane povećava pogrešku u odnosu na slučaj kada je grana ostavljena u stablu, onda se grana ostavlja u stablu. Obično se generira niz od nekoliko naknadno podrezanih stabala, a zatim se među njima odabire ono koje dovodi do najmanje pogreške. Druga metoda koja se koristi je skraćivanje ovisno o minimalnom broju bitova koje je potrebno upotrijebiti kako bi se stvorilo stablo.

4.4.2.2 Poboljšanja osnovnog algoritma za stvaranje stabla odlučivanja

Poboljšanja osnovnog algoritma ID.3 uključena su u algoritam *C4.5* [48]. Algoritam *J48* [39] je implementacija C4.5 u programskom alatu *Weka*.

Osnovni algoritam opisan na slici 4.5. pretpostavlja da su svi atributi kategorički ili diskretni. Algoritam se može modificirati tako da atributi mogu imati intervale diskretnih ili kontinuiranih vrijednosti. Primjer: za neku numeričku vrijednost V test koji se provodi nad atributom A može iz čvora generirati dvije grane: jednu za koju je $a_i \leq V$ i drugu za koju je $a_i > V$, gdje je a_i vrijednost atributa A za i-ti uzorak. Ako je za atribut A zadano v vrijednosti, onda iz tog čvora proizlazi v-1 grana.

Predložene su i mnoge metode kako postupati s vrijednostima atributa koje nedostaju, npr. ako za nove uzorke nedostaje vrijednost nekog atributa, onda se taj uzorak može tretirati kao da mu je vrijednost atributa koja nedostaje najčešća vrijednost tog atributa ili se informacijska dobit za taj atribut može umanjiti s obzirom na broj uzoraka kojima je nepoznata vrijednost tog atributa.

4.4.3. Naivni Bayesov klasifikator

Bayesovi klasifikatori su statistički klasifikatori. Temelje se na predviđanju vjerojatnosti da neki uzorak pripada nekom razredu, a pri tome se koristi Bayesov teorem. Dobra strana ove metode je brzina, čak i za velike skupove podataka. Metoda je opisana prema [5] i [60].

Najveći nedostatak ove metode je u temeljnoj pretpostavci da je djelovanje nekog atributa na pripadnost uzorka pojedinom razredu neovisno o djelovanju drugih atributa, odnosno da su atributi međusobno neovisni, što u praksi nije uvijek istina. Ova pretpostavka znatno pojednostavljuje računalnu obradu, te se stoga zove *naivnom*, što je uključeno i u naziv klasifikatora.

U teoriji Bayesov klasifikator u odnosu na druge klasifikatore ima minimalnu pogrešku, međutim u praksi to često nije tako, jer pretpostavka da su atributi međusobno neovisni nije ispunjena. Istraživanja su pokazala da je u nekim domenama Bayesov klasifikator sumjerljiv s neuronskim mrežama i stablima odlučivanja.

4.4.3.1 Bayesov teorem

Neka je X uzorak podataka za koji je razred nepoznat. Neka je H pretpostavka da uzorak X pripada razredu R. Za problem razvrstavanja želimo odrediti $P(H|X)$, tj. želimo odrediti vjerojatnost da je pretpostavka H ispravna za zadani X. Bayesov teorem je:

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

$P(H|X)$ je aposteriori vjerojatnost da vrijedi H, ako vrijedi X. $P(H)$ je apriori vjerojatnost neovisna o X. $P(X|H)$ je aposteriori vjerojatnost da vrijedi X ako vrijedi H.

Primjer:
Neka je X gumeno i žuto, a neka je pretpostavka H da je X lopta. Tada je $P(H|X)$ vjerojatnost da vrijedi H uz uvjet da znamo da je X gumeno i žuto. $P(H)$ je vjerojatnost da H vrijedi. P($X|H$) je vjerojatnost da je X gumeno i žuto, ako znamo da je X lopta.

4.4.3.2 Algoritam naivnog Bayesovog klasifikatora

Neka je svaki uzorak iz skupa U podataka predstavljen kao n-dimenzionalni vektor: $X = (x_1, ..., x_n)$, gdje x_i vrijednost atributa A_i za zadani uzorak. Neka je zadano m razreda: $R_1, ..., R_m$. Broj uzoraka u skupu U je u, a u pojedinom razredu u_i.

Ako za uzorak X nije poznat razred, tada će klasifikator predvidjeti da X pripada onom razredu koji ima najveću aposteriornu vjerojatnost $P(R_i|X)$, tj. X će biti razvrstan u razred R_i ako i samo ako vrijedi: $P(R_i|X) > P(R_j|X)$ uz uvjet $j = 1, ..., m$ i $i \neq j$. Razred R_i za koji je $P(R_i|X)$ maksimalna zove se *maksimalna aposteriori hipoteza*.

Prema Bayesovom teoremu vrijedi:

$$P(R_i \mid X) = \frac{P(X \mid R_i)P(R_i)}{P(X)}.$$

S obzirom da je $P(X)$ konstantan za sve razrede, potrebno je maksimizirati $P(X \mid R_i)P(R_i)$.

Ako unaprijed nisu poznate vjerojatnosti pojedinih razreda, onda je uobičajeno pretpostaviti da je vjerojatnost pripadanja pojedinom razredu jednaka, tj. da vrijedi: $P(R_1) = ... = P(R_m)$, pa je stoga potrebno maksimizirati $P(X \mid R_i)$, a ako ne vrijedi da je vjerojatnost pripadanja pojedinom razredu jednaka, onda je potrebno maksimizirati jednakost $P(X \mid R_i)P(R_i)$.

Pripadnost pojedinom razredu može se odrediti kao:

$$P(R_i) = \frac{u_i}{u}.$$

Uz *naivnu* pretpostavku o neovisnosti atributa, određuje se $P(X \mid R_i)$ kao:

$$P(X \mid R_i) = \prod_{k=1}^{n} P(x_k \mid R_i),$$

a vjerojatnosti $P(x_k \mid R_i)$ mogu se odrediti na temelju podataka iz skupa pokaznih uzoraka:

(i) Ako je atribut A_k *kategorička* varijabla, onda je:

$$P(x_k \mid R_i) = \frac{u_{ik}}{u_i},$$

gdje je u_{ik} broj uzoraka iz pokaznog skupa uzoraka kojima je x_k vrijednost atributa A_k, a pripadaju razredu R_i.

(ii) Ako je A_k varijabla s *kontinuiranim* vrijednostima, onda se pretpostavlja da atribut ima Gaussovu distribuciju, tj. da vrijedi:

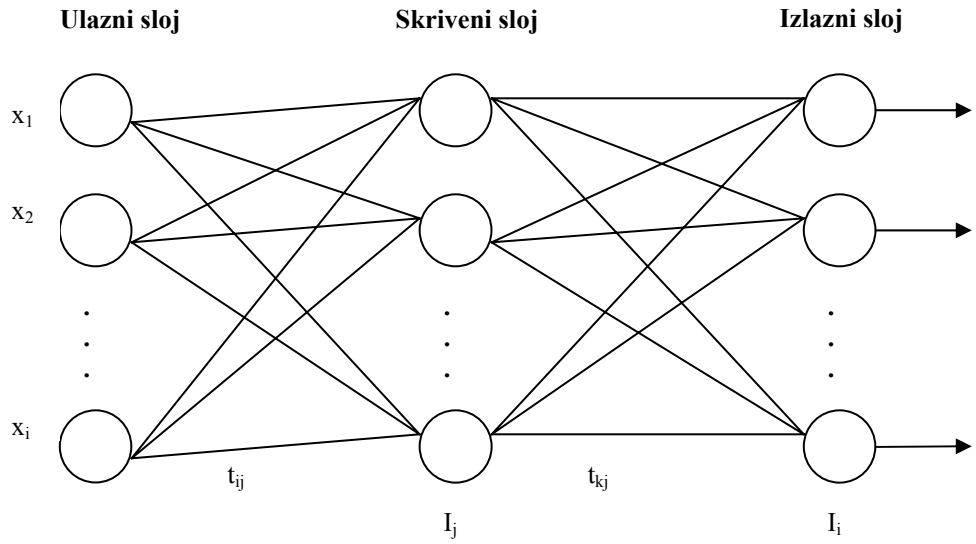
$$P(x_k \mid R_i) = \frac{1}{\sqrt{2\pi}\sigma_{R_i}} e^{-\frac{(x_k - \mu_{R_i})}{2\sigma_{R_i}}},$$

gdje je μ_{R_i} srednja vrijednost i σ_{R_i} standardna devijacija atributa A_k za razred R_i.

4.4.3.3 Neuronske mreže i algoritam prenošenja informacije unatrag

Neuronska mreža je skup međusobno povezanih ulazno-izlaznih jedinica gdje svaka veza nosi određenu težinu [5]. Za vrijeme postupka učenja težine se podešavaju kako bi što bolje predvidjele razred za zadane uzorke podataka. Najveći nedostatak neuronskih mreža je složena interpretacija dobivenih pravila koja uključuju proračunate težine. Pozitivna im je strana otpornost na šum u podacima i uzorke s ekstremnim vrijednostima, a isto se tako pokazalo da su u nekim primjenama sposobne dobro razvrstati uzorke na kakvima nisu bili trenirane [5].

Prenošenje informacije unatrag (eng. *backpropagation*) je algoritam učenja temeljen na neuronskim mrežama. Algoritam prenošenja informacije unatrag je učenje utemeljeno na višeslojnim neuronskim mrežama gdje je izlaz svakog sloja, osim posljednjeg ili izlaznog sloja, ulaz za onaj sljedeći sloj (eng. *multilayer feed-forward*) (slika 4.6).



Slika 4.6. Višeslojna neuronska mreža

Vrijednosti pojedinih atributi nekog uzorka simultano se unose kroz *ulazni sloj*. Izlaz iz ovog sloja se sastoji od više izlaza koji imaju svoje težine, a ujedno su ulaz za sljedeći sloj koji se zove *skriveni sloj*. Broj skrivenih slojeva (slojeva između ulaznog i izlaznog sloja) je proizvoljan, ali se u praksi najčešće koristi jedan sloj. Težinski izlazi iz zadnjeg skrivenog sloja predstavljaju ulaz u posljednji – *izlazni sloj*.

Neuronska mreža koja sadrži jedan skriveni sloj zove se dvoslojna neuronska mreža, onda koja sadrži dva skrivena sloja zove se troslojna neuronska mreža, itd. Ovakva struktura je u potpunosti povezana zato što izlaz iz bilo kojeg sloja može biti dio ulaza u skriveni ili izlazni sloj.

Ulaz:
U - skup uzoraka
l - mjera učenja
Mreža - višeslojna neuronska mreža

Izlaz:
Neuronska mreža istrenirana za razvrstavanje uzoraka.

Postupak:
inicijalizirati sve težine t_{ij} i odstupanja θ_j u mreži *Mreža*
ponavljaj dok uvjet završetka nije ispunjen
za svaki uzorak *X* iz *U*
// prijenos ulaza prema naprijed
za svaki skriveni ili izlazni sloj *j*
$U_j = \sum_i t_{ij} I_i + \theta_j$ // određivanje ulaza *j*-tog sloja
$I_j = \frac{1}{1+e^{-U_j}}$ // određivanje izlaza *j*-tog sloja

// vraćanje informacije o pogrešci
za svaku jedinicu *j* izlaznog sloja
$Gr_j = I_j(1 - I_j)(S_j - I_j)$ // proračun pogreške

za svaku jedinicu *j* u skrivenom sloju (od zadnje prema prvoj)
$Gr_j = I_j(1 - I_j)\sum_k Gr_k t_{jk}$ // k - jedinice iz (j+1) sloja

za svaku težinu t_{ij} u *Mreži*
$\Delta t_{ij} = (l) Gr_j I_i$ // prirast težine
$t_{ij} = t_{ij} + \Delta t_{ij}$ // nova težina

za svako odstupanje θ_j u *Mreži*
// odrediti prirast i povećanje odstupanja
$\Delta\theta_j = (l) Gr_j$
$\theta_j = \theta_j + \Delta\theta_j$

Slika 4.7. Algoritam prenošenja informacije unatrag

Na početku se sve težine t_{ij} koje se koriste u neuronskoj mreži inicijaliziraju na slučajne vrijednosti iz nekog manjeg intervala (npr. od -1.0 do 1.0 ili od -0.5 do 0.5). Svaka jedinica unutar jednog sloja neuronske mreže ima svoje odstupanje (eng. *bias*) koje se također inicijalizira na male slučajne vrijednosti.

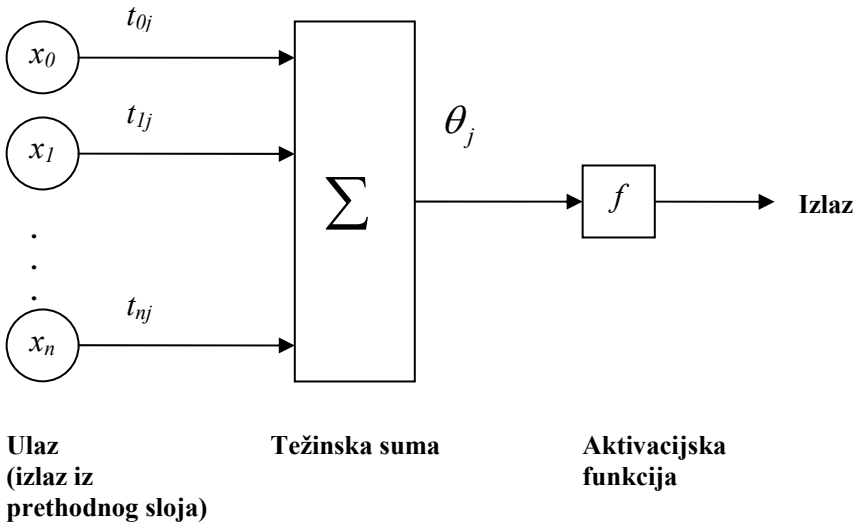
Vrijednosti atributa nekog uzorka ulazni su podaci za jedinice ulaznog sloja (na slici 4.6. postoji *i* ulaznih jedinica).

Ulazne vrijednosti za skrivene slojeve i izlazni sloj se određuju kao linearna kombinacija svojih ulaza (slika 4.7.). Ako je s *j* označena jedinica promatranog sloja, a s *i* označene jedinice prethodnog sloja čiji izlazi čine ulaz u jedinicu *j*, onda vrijedi: $U_j = \sum_i t_{ij} I_i + \theta_j$, gdje je t_{ij} težina veze između jedinice *i* iz prethodnog sloja i

jedinice j u promatranom sloju. Odstupanje θ_j je prag koji služi kako bi varirao aktivnost jedinice j.

Nakon što se izračuna ulaz, onda se prema aktivacijskoj funkciji određuje i izlaz iz te jedinice. Kao *aktivacijska funkcija* koristi se logistička funkcija, tj. $I_j = \frac{1}{1+e^{-U_j}}$. Ova funkcija sažima izlaznu vrijednost na vrijednost iz intervala od 0 do 1.

Nakon proračuna izlazne vrijednosti iz izlaznog sloja, računa se pogreška algoritma za zadani uzorak i izračunata pogreška se zatim prenosi iz zadnjeg (izlaznog) sloja prema slojevima unatrag sve do prvog skrivenog sloja. Pogreška j-te jedinice u izlaznom sloju se određuje kao: $Gr_j = I_j(1-I_j)(S_j - I_j)$, gdje je S_j stvarni izlaz koji je poznat za pokazne uzorke nad kojima se mreža trenira.



Slika 4.8. Prikaz jedinice *j* skrivenog ili izlaznog sloja

U gornjem se primjeru težine i odstupanja ažuriraju nakon svakog novog pokaznog uzorka koji se provede kroz sustav. Ovakav pristup se naziva *ažuriranje po uzorcima* (eng. *case updating*). Drugi je pristup ažuriranje težina i odstupanja nakon što su svi uzorci provedeni kroz sustav. Ovakav se pristup naziva *epohalno ažuriranje* (eng. *epoch updating*), a jedna se iteracija po svim pokaznim uzorcima naziva *epoha*. U teoriji, matematički oblik prenošenja informacije unatrag temelji se na epohalnom pristupu, dok se u praksi više koristi ažuriranje po uzorcima, jer daje točnije rezultate [5].

Trenutak u kojima završava treniranje neuronske mreže može se odabrati na neki od sljedećih načina:

(i) ako su prirasti Δt_{ij} manji od unaprijed zadanog praga

(ii) postotak pogrešno razvrstanih uzoraka u prethodnoj epohi je manji od unaprijed zadanog praga

(iii) završen je unaprijed zadani broj epoha

4.4.4. Metoda *k*-najbližih susjeda

Metoda *k*-najbližih susjeda [60], [5] se temelji na učenju po principu analogije. Svaki je pokazni uzorak predstavljen *n*-dimenzionalnim vektorom, gdje je *n* broj dimenzija koje su numerički atributi. Kada se kroz trenirani sustav provede nepoznati uzorak, tada sustav traži *k* uzoraka koji su najbliži ili najsličniji zadanom uzorku. Odabrani najsličniji uzorci se nazivaju *k-najbližih susjeda* (eng. *k-nearest neighbors*). Nepoznati uzorak se razvrstava u onaj razred kojemu pripada većina od njegovih *k*-najbližih susjeda.

Za slučaj kada se ovaj algoritam koristi za predviđanje, a ne razvrstavanje, onda se numerička vrijednost koju algoritam treba predvidjeti dobiva kao prosječna vrijednost *k* najbližih susjeda zadanog uzorka.

Sličnost ili udaljenost se definira korištenjem neke standardne mjere udaljenosti, npr. Euklidske udaljenosti.

Primjer:

Neka su zadana sva uzorka *X* i *Y* koji su predstavljeni kao točke u *n*-dimenzionalnom prostoru. Euklidska udaljenost između ta dva uzorka je:

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

Što je udaljenost $d(X,Y)$ manja, to se uzorci smatraju sličnijima.

4.4.5. Algoritam VFI

Algoritam VFI (eng. *Voting Feature Interval*) [8] se temelji na razvrstavanju vrijednosti koje pojedini atribut može poprimiti u neki od intervala vrijednosti predviđenih za taj atribut. Intervali vrijednosti (eng. *feature interval*) koje može poprimiti pojedini atribut izgrađuju se na temelju pokaznog skupa uzoraka. Svaki interval pojedinog atributa se odnosi na jedan od razreda.

Postupak razvrstavanja novog uzorka (uzorka čiji je razred nepoznat), sastoji se u razvrstavanju vrijednosti svakog atributa novog uzorka u neki od intervala predviđenih za taj atribut. S obzirom da se svaki interval nekog atributa odnosi na neki razred, tako svaki atribut *glasuje* za pripadnost novog uzorka nekom od razreda. Ako se vrijednost nekog atributa novog uzorka nalazi na granici intervala dvaju razreda, onda se glas dijeli između tih razreda. Novi uzorak se razvrstava u onaj razred koji prikupi najviše *glasova*.

Ovaj algoritam sličan je naivnom Bayesovom algoritmu, po tome što se svaki atribut promatra zasebno i svaki ravnopravno sudjeluje u donošenju odluke u koji razred treba pripasti novi uzorak. Prednosti ovog algoritma pred naivnim Bayesovim algoritmom su veća brzina, te otpornost algoritma na međuzavisnost atributa. U praktičnom dijelu rada ovaj se algoritam pokazao kao izvrstan: pokazao je skalabilnost na velik broj atributa, te otpornost na međuzavisnost atributa.

4.4.5.1 Faza treniranja klasifikatora

Faza treniranja klasifikatora počinje s pronalaženjem najmanje i najveće vrijednosti koju poprimaju vrijednosti svakog atributa za uzorke iz svakog razreda. Općenito: ako je *r* broj razreda kojima atributi mogu pripadati, onda je *2r* broj vrijednosti (*r* najmanjih i *r* najvećih vrijednosti) koje je potrebno pronaći za svaki atribut. Primjer: ako svi uzorci mogu pripadati jednom od dva razreda, onda je za svaki

od ta dva razreda potrebno naći najmanju i najveću vrijednost za svaki atribut koji opisuje uzorke, što čini ukupno četiri vrijednosti koje je potrebno pronaći za svaki atribut. Ako je atribut kategorički, tada svaka promatrana vrijednost predstavlja zaseban interval.

Nakon što se za svaki razred odredi najmanja i najveća vrijednost za svaki atribut, intervali vrijednosti se sortiraju. Svaki je interval dovoljno predstaviti samo donjom granicom intervala, jer je za numeričke vrijednosti atributa gornja granica jednog intervala ujedno donja granica sljedećeg intervala, a za nominalne vrijednosti atributa donja granica je i gornja granica intervala.

Sljedeći je korak formiranje liste vektora za svaki atribut A_k, gdje je svaki vektor u listi oblika *<donja_granica*, *brojUzoraka_1*, ..., *brojUzoraka_r>*. Vrijednost *donja_granica* je donja granica intervala atributa A_k, a vrijednosti *brojUzoraka_j* predstavljaju broj uzoraka koji pripadaju *j*-tom razredu ($j = 1$, .., r) za koje je vrijednost atributa A_k unutar tog intervala.

Na slici 4.9. prikazana je faza treniranja klasifikatora. Procedura `nađi_granice_intervala` pronalazi u skupu uzoraka donju i gornju granicu vrijednosti atributa A_k koju mogu poprimiti uzorci iz razreda R_1 do R_r. Procedura `prebroji_uzorke` broji uzorke iz razreda R_j koji se nalaze unutar intervala *i* atributa A_k. Ako je vrijednost atributa A_k za neki uzorak na granici *i*-tog i susjednog intervala, onda se brojač za takav uzorak u *i*-tom intervalu ne uveća za 1, nego se uveća za 0.5 za *i*-ti i njemu susjedni interval. Izlaz algoritma je lista *brojac_razred*`_interval` čiji je svaki element *broj_uzoraka_u_intervalu*`[`A_k`, i, `R_j`]` koji sadrži broj uzoraka iz razreda R_j čija je vrijednost atributa A_k iz intervala *i*.

```
Ulaz:
   U - skup uzoraka
   A – skup n atributa {A1, ..., An} koji opisuje skup uzoraka
   R - lista razreda {R1, ..., Rr}

Izlaz:
   brojac_razred_interval

Postupak:
   za svaki atribut Ak
      za svaki razred Rj
         Granica[Ak] = Granica[Ak] ∪ nađi_granice_intervala(U, Ak, Rj)

      sortiraj(Granica[Ak])

      // za svaki par iz Granica[Ak] formiraj interval
      za svaki interval i
         za svaki razred Rj
            // odredi broj uzoraka u razredu Rj koji su u intervalu i
            broj_uzoraka_u_intervalu[Ak, i, Rj] =
                        prebroji_uzorke (Ak, i, Rj)
```

Slika 4.9. Algoritam VFI - faza treniranja

4.4.5.2 Faza razvrstavanja klasifikatora

Na slici 4.10. prikazana je faza razvrstavanja algoritma VFI. Za svaki uzorak prikupljaju se *glasovi* (eng. *votes*) na temelju kojih se određuje kojemu razredu će uzorak pripadati.

Neka uzorak U_0 za atribut A_k ima vrijednost U_0 (A_k).

(i) Ako je vrijednost $U_0(A_k)$ nepoznata (eng. *missing value*), tada atribut A_k svakom razredu R_j, gdje je $j = 1, ..., r$ daje 0 glasova.

(ii) Ako je $u(A_k)$ određena vrijednost koja se nalazi unutar intervala i, tada se glas koji daje atribut A_k određuje kao:

$$glas\_atributa[A_k, R_j] = broj\_uzoraka\_u\_intervalu[A_k, i, R_j] / broj\_uzoraka[R_j]$$

Vrijednost $broj\_uzoraka\_u\_intervalu[A_k, i, R_j]$ je broj uzoraka čija je vrijednost atributa A_k u intervalu i, a koji su u razredu R_j.

(i) Ako je A_k numerički atribut, a vrijednost $u(A_k)$ na granici dva intervala, tada je glas atributa A_k prosjek koji se dobije za interval i i interval $i + 1$.

(ii) Ako atribut A_k ima nominalne vrijednosti, tada je svaka nominalna vrijednost unutar jednog intervala, pa nema podjele glasova između susjednih intervala.

Vrijednost $glas\_atributa[A_k, R_j]$ je glas atributa A_k za razred R_j. Ta se vrijednost normira tako da suma glasova atributa A_k za sve razrede zajedno bude realna vrijednost manja ili jednaka 1.

Glasovi svih atributa se pohranjuju u vektor glasova:

$$<glas(A_k, R_1), ..., glas(A_k, R_r)>$$

gdje je $glas(A_k, R_j)$ realan broj koji predstavlja glas koji atribut A_k daje za razred R_j (na slici 4.10. označen je s $glas\_atributa[A_k, R_j]$).

Glasovi svih atributa se pribrajaju za ukupne glasove koji su prikazani vektorom:

$$<glas(R_1), ..., glas(R_r)>$$

gdje je $glas(R_j)$ je zbroj glasova koje je razred R_j (na slici 4.10. označen je s $glas[R_j]$).

Uzorak se svrstava u onaj razred koji je dobio najviše glasova. Alternativno, može se za svaki uzorak odrediti vjerojatnost da pripada pojedinom razredu, a vjerojatnost je definirana kao: $\frac{glas(R_j)}{\sum_j glas(R_j)}$.

```
Ulaz:
   U0 – uzorak koji treba razvrstati
   A – skup n atributa {A1, ..., An} koji opisuje skup uzoraka
   R - lista razreda {R1, ..., Rr}

Izlaz:
   Razred kojemu uzorak U0 pripada.

Postupak:
   za svaki razred Rj
      glas[Rj] = 0

   za svaki atribut Ak
      za svaki razred Rj
         glas_atributa[Ak, Rj] = 0 // glas atributa Ak za razred Rj
      ako je U0(Ak) poznata vrijednost
         i = interval(Ak, U0(Ak)) // interval u kojem je U0(Ak)
         glas_atributa[Ak, Rj] = broj_uzoraka_u_intervalu (Ak, i, Rj) /
                                 broj_uzoraka[Rj]
         normiraj_vektore_glas_atributa(Ak) // suma glasova = 1
      za svaki razred Rj
         glas[Rj] = glas[Rj] + glas_atributa[Ak, Rj]

   vrati Rj za koji je glas[Rj] maksimalan
```

Slika 4.10. Faza razvrstavanja - algoritam VFI

Primjer [8]:

Neka je zadano sedam uzoraka u skupu pokaznih uzoraka. Svaki je uzorak opisan s dva atributa: A_0 i A_1 i oznakom razreda. Neka 3 uzorka pripadaju razredu A i 4 uzorka razredu B. U tablici 4.2 prikazane su vrijednosti atributa i razreda za svaki od uzoraka iz pokaznog skupa uzoraka, te vrijednosti testnog uzorka za koji treba odrediti razred:

Tablica 4.2. Tablica koja prikazuje pokazni skup uzoraka i testni uzorak

	A_0	A_1	**razred**
uzorak_1	1	4	*A*
uzorak_2	2	1	*A*
uzorak_3	2	7	*A*
uzorak_4	3	4	*B*
uzorak_5	4	6	*B*
uzorak_6	6	4	*B*
uzorak_7	6	8	*B*

Faza treniranja klasifikatora

Za atribut A_0:

- za razred *A* vrijednost A_0 se nalazi u intervalu: [1, 2]
- za razred *B* vrijednost A_0 se nalazi u intervalu: [3, 6]

$Granica[A_0] = \{1, 2, 3, 6\}$
$Interval[A_0] = \{<-\infty, 1], [1, 2], [2, 3], [3, 6], [6, +\infty>\}$

$broj\_uzoraka\_u\_intervalu[A_0, <-\infty, 1], A] = 0.5$
$broj\_uzoraka\_u\_intervalu[A_0, <-\infty, 1], B] = 0$
$broj\_uzoraka\_u\_intervalu[A_0, [1, 2], A] = 3 * 0.5 = 1.5$
$broj\_uzoraka\_u\_intervalu[A_0, [1, 2], B] = 0$
$broj\_uzoraka\_u\_intervalu[A_0, [2, 3], A] = 2 * 0.5 = 1$
$broj\_uzoraka\_u\_intervalu[A_0, [2, 3], B] = 0.5$
$broj\_uzoraka\_u\_intervalu[A_0, [3, 6], A] = 0$
$broj\_uzoraka\_u\_intervalu[A_0, [3, 6], B] = 1 + 3 * 0.5 = 2.5$
$broj\_uzoraka\_u\_intervalu[A_0, [6, +\infty>, A] = 0$
$broj\_uzoraka\_u\_intervalu[A_0, [6, +\infty>, B] = 2 * 0.5 = 1$

Za atribut A_1:

- za razred *A* vrijednost A_1 se nalazi u intervalu: [1, 7]
- za razred *B* vrijednost A_1 se nalazi u intervalu: [4, 8]

$Granica[A_1] = \{1, 4, 7, 8\}$
$Interval[A_1] = \{<-\infty, 1], [1, 4], [4, 7], [7, 8], [8, +\infty>\}$
$broj\_uzoraka\_u\_intervalu[A_1, <-\infty, 1], A] = 0.5$

broj_uzoraka_u_intervalu[A_1, $<-\infty$, 1], *B*] = 0
broj_uzoraka_u_intervalu[A_1, [1, 4], *A*] = 2 * 0.5 = 1
broj_uzoraka_u_intervalu[A_1, [1, 4], *B*] = 2 * 0.5 = 1
broj_uzoraka_u_intervalu[A_1, [4, 7], *A*] = 2 * 0.5 = 1
broj_uzoraka_u_intervalu[A_1, [4, 7], *B*] = 2 * 0.5 + 1 = 2
broj_uzoraka_u_intervalu[A_1, [7, 8], *A*] = 0.5
broj_uzoraka_u_intervalu[A_1, [7, 8], *B*] = 0.5 = 0.5
broj_uzoraka_u_intervalu[A_1, [8, $+\infty>$, *A*] = 0
broj_uzoraka_u_intervalu[A_1, [8, $+\infty>$, *B*] = 0.5

Faza razvrstavanja nepoznatog uzorka
Neka je za testni uzorak $A_0 = 5$ i $A_1 = 2$. Potrebno je odrediti razred kojemu uzorak pripada. Na temelju intervala određenih u fazi treniranja klasifikatora za testni uzorak uočeno je da vrijednost atributa A_0 pripada intervalu [3, 6], a vrijednost atributa A_1 pripada intervalu [1, 4].
Prvo se za svaki atribut odredi kojemu bi razredu dao glas:
glas_atributa[A_0, *A*] = *broj_uzoraka_u_intervalu* (A_0, *[3,6]*, *A*) / *broj_uzoraka*[*A*]
= 0 / 3 = 0
glas_atributa[A_0, *B*] = *broj_uzoraka_u_intervalu* (A_0, *[3,6]*, *B*) / *broj_uzoraka*[*B*]
= 2.5 / 4 = 0.625
glas_atributa[A_1, *A*] = *broj_uzoraka_u_intervalu* (A_1, *[1,4]*, *A*) / *broj_uzoraka*[*A*]
= 1 / 3 = 0.333
glas_atributa[A_1, *B*] = *broj_uzoraka_u_intervalu* (A_1, *[1,4]*, *B*) / *broj_uzoraka*[*B*]
= 1 / 4 = 0.25

Normirani vektori za atribut A_0 (suma glasova mora biti 1):
glas_atributa[A_0, *A*] = 0
glas_atributa[A_0, *B*] = 1

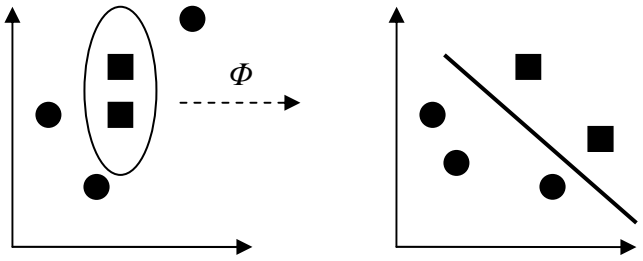
Normirani vektori za atribut A_1 (suma glasova mora biti 1):
glas_atributa[A_1, *A*] = 0.57
glas_atributa[A_1, *B*] = 0.43

Određivanje sume glasova:
glas[*A*] = 0.57
glas[*B*] = 1.43
S obzirom da je razred *B* dobio više glasova, zaključak je da testni uzorak pripada razredu *B*.

4.4.6. Algoritam SVM

Algoritam *SVM* (eng. *Support Vector Machines*) [59], [9] se sastoji u preslikavanju skupa pokaznih uzoraka iz ulaznog prostora uzoraka (eng. *input space*) $\Re^N$ u drugi višedimenzionalni prostor *F* (eng. *feature space*). Preslikavanje se obavlja korištenjem funkcije Φ, čime svaki pokazni uzorak $\mathbf{x}$ (*N*-dimenzionalni vektor) iz ulaznog prostora postaje $\Phi(\mathbf{x})$ iz prostora *F*.

Nakon što se vektor uzorka preslika u novi prostor, u novom prostoru se određuje kojem razredu vektor pripada. U novom je prostoru određena *hiperravnina razdvajanja* koja predstavlja *granicu odlučivanja* za ulazni prostor podataka [9], odnosno razdvaja mapirane uzorke u dva razreda. *Optimalna hiperravnina* je ona s maksimalnom granicom razdvajanja između uzoraka dvaju razreda. Prema položaju vektora $\Phi(\mathbf{x})$ u odnosu na *hiperravninu razdvajanja* unutar prostora F određuje se kojem razredu pripada $\Phi(\mathbf{x})$ (prikazano na slici 4.10.).



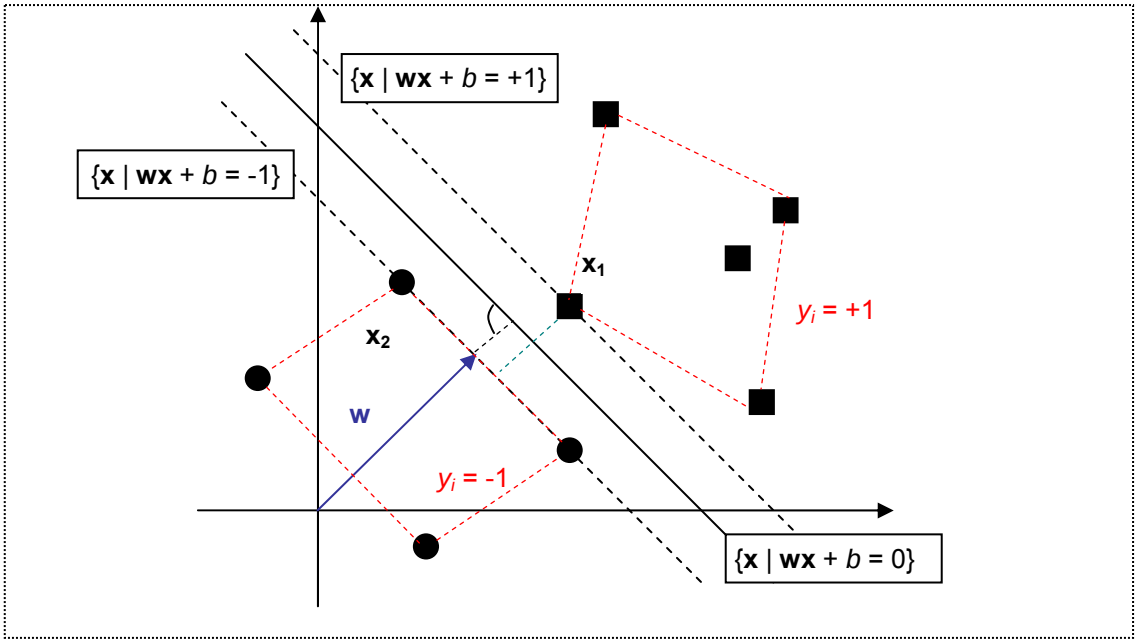
Slika 4.11. Ideja SVM algoritma: uzorke koji su svrstani u dva razreda preslikava se u prostor dimenzija *F* u kojemu se formira hiperravnina razdvajanja

Neka je zadan skup pokaznih uzoraka $\{\mathbf{x_1}, ..., \mathbf{x_l}\}$ koji ima l elemenata. Svaki uzorak ima N dimenzija. Razred kojemu pripada i-ti uzorak označen je s y_i i može biti ±1, tj. vrijedi:

$$(\mathbf{x_1}, y_1),...(\mathbf{x_1}, y_l) \in \Re^N \times \{\pm 1\}.$$

Hiperravnina razdvajanja je okomita na najkraću liniju spajanja *konveksnih ljusaka*[1] (eng. *convex hull*) dvaju razreda ±1 (slika 4.12.).

[1] *Konveksna ljuska* je najmanji konveksni skup koji sadrži cijeli skup S, gdje je S zadan u cijelom prostoru. Može se vizualizirati kao balon koji obavija sve elemente skupa S i nalazi se najbliže moguće skupu S[10].



Slika 4.12. Određivanje hiperravnine razdvajanja

Preslikavanje u prostor F se obavlja nelinearnim mapiranjem:

$$\Phi : \Re^N \rightarrow F .$$

U prostoru F potrebno je konstruirati funkciju f koja razvrstava nove uzorke (čiji je razred nepoznat) tako da vrijedi $f(\mathbf{x}) = y = \pm 1$, gdje je $\mathbf{x}$ N-dimenzionalni vektor u prostoru F, a y je razred uzorka, a može biti ± 1.

Funkcija odlučivanja f se definira kao: $f(\mathbf{x}) = sign((\mathbf{wx}) + b) = \pm 1$,

gdje je $(\mathbf{wx}) + b = 0, \mathbf{w} \in \Re^n, b \in R$ razred hiperravnina.

Vektor $\mathbf{w}$ se određuje kao: $\mathbf{w} = \sum_i \mathbf{v_i x_i}$, gdje su vektori $\mathbf{x_i}$ oni koji leže na granicama konveksnih ljusaka razreda (slika 4.12.), a zovu se *vektori podrške* (eng. *support vectors*). Ti su vektori dobiveni preslikavanjem nekih pokaznih uzoraka, a nose sve relevantne informacije o problemu razvrstavanja.

Prema tome, funkcija odlučivanja temelji se na vektorima podrške:

$$f(\mathbf{x}) = sign(\sum_i \mathbf{v_i}(\mathbf{x x_i}) + b) = \pm 1 .$$

Parametri $\mathbf{v_i}$ dobiju se rješavanjem kvadratnih jednadžbi.

Sada se situacija preslika u prostor F: svaki vektor podrške $\mathbf{x_i}$ zamijeni se s $\Phi(\mathbf{x_i})$, a vektor $\mathbf{x}$ se zamijeni s $\Phi(\mathbf{x})$.

Arhitektura SVM metode:

1. ulazni vektor $\mathbf{x}$ i vektori podrške $\mathbf{x_i}$ mapiraju se nelinearnom funkcijom Φ u višedimenzionalni prostor F
2. u prostoru F računaju se skalarni produkti $k(\mathbf{x}, \mathbf{x_i}) = \Phi(\mathbf{x})\Phi(\mathbf{x_i})$, gdje je k *funkcija jezgre*
3. određuju se težine $\mathbf{v_i}$ rješavanjem kvadratne jednadžbe
4. izlazna vrijednost, odnosno razred uzorka se određuje s:

$$f(\mathbf{x}) = sign(\sum_i \mathbf{v_i} k(\mathbf{x x_i}) + b)$$

Primjeri *funkcije jezgre k* (eng. *kernel function*):

1. polinomna funkcija: $k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\mathbf{y})^d$, gdje je *d* broj dimenzija prostora
2. funkcija *RBF* (eng. *radial basis function*): $k(\mathbf{x},\mathbf{y}) = exp\left(-\|\mathbf{x}-\mathbf{y}\|^2 / \left(2\sigma^2\right)\right)$
3. sigmoidalna funkcija (uz parametre κ i θ) $k(\mathbf{x},\mathbf{y}) = tanh\left(\kappa(\mathbf{x}\mathbf{y}) + \theta\right)$

5. Evaluacija rezultata

Evaluacija rezultata dobivenih dubinskom analizom podataka prvenstveno se temelji na evaluaciji eksperimentalnih rezultata [46]. Evaluacije klasifikatora odnosi se na mjerenje njegove efikasnosti, tj. odnosi se na sposobnost klasifikatora da pravilno razvrsta što veći broj uzoraka iz testnog skupa podataka.

Matrica konfuzije (eng. *confusion matrix*) ([36], [37], [38]) sadrži informaciju o djelovanju klasifikatora nad testnim skupom podataka, a uspoređuju se rezultati dobiveni razvrstavanjem uzoraka u odnosu na stvarne razrede kojima uzorci pripadaju.

Neka razred uzorka označava da li uzorak djeluje na promatrani sustav ili ne. Ako je oznaka razreda *promjena*, onda promatrani uzorak djeluje na ispitivani sustav, a ako je oznaka razreda *nema promjene*, onda uzorak ne djeluje na ispitivani sustav. Oznaka *promjena* smatrat će se pozitivnim ishodom, a oznaka *nema promjene* negativnim ishodom.

Podaci se prikazuje u matričnom obliku (tablica 5.1.).

Tablica 5.1. Matrica konfuzije

		Predviđanje	
		nema promjene	*promjena*
Razred uzorka	*nema promjene*	**a**	**b**
	promjena	**c**	**d**

Definiraju se sljedeće vrijednosti:

1. ***a*** je broj *ispravno* predviđenih *negativnih* ishoda
2. ***b*** je broj *pogrešno* predviđenih *pozitivnih* ishoda
3. ***c*** je broj *pogrešno* predviđenih *negativnih* ishoda
4. ***d*** je broj *ispravno* predviđenih *pozitivnih* ishoda

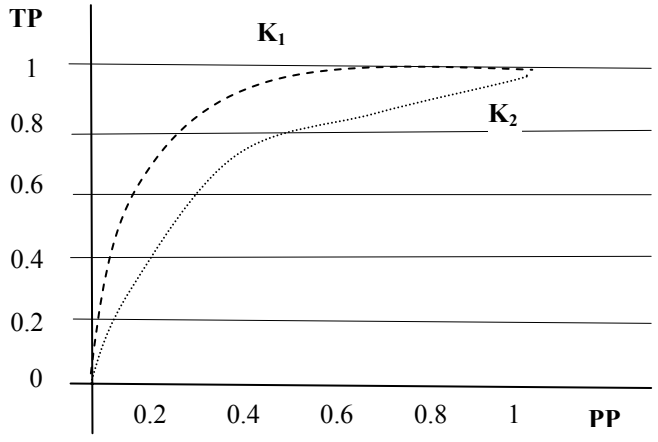
Mjere koje se koriste za matricu koja prikazuje rezultate predviđanja za podatke koji pripadaju dvama razredima:

1. *točnost* (eng. *accuracy*) je omjer uzoraka kojima je razred točno predviđen i ukupnog broja uzoraka:
$$\frac{a+d}{a+b+c+d}$$
2. *opoziv* (eng. *recall*) ili mjera *točno predviđenih pozitivnih uzoraka* (*TP*) (eng. *true positive rate*):
$$\frac{d}{c+d}$$
3. *mjera pogrešno predviđenih pozitivnih uzoraka* (PP) (eng. *false positive rate*) je omjer uzoraka koji su pogrešno svrstani u pozitivan razred i ukupnog broja negativnih uzoraka:
$$\frac{b}{a+b}$$

4. *mjera točno predviđenih negativnih uzoraka* (eng. *true negative rate*):
$$\frac{a}{a+b}$$
5. *mjera pogrešno predviđenih negativnih uzoraka* (eng. *false negative rate*) je omjer uzoraka koji su pogrešno svrstani u negativan razred i ukupnog broja pozitivnih uzoraka:
$$\frac{c}{c+d}$$
6. *preciznost* (eng. *precision*) je omjer točno predviđenih pozitivnih uzoraka i ukupnog broja uzoraka za koje je predviđen pozitivan razred:
$$\frac{d}{b+d}$$

Točnost koja se dobije s izrazom (1) nije uvijek prikladna mjera efikasnosti predviđanja sustava. Problem je za sljedeće slučajeve: ako je velika većina slučajeva negativna (npr. ako 990 uzoraka od ukupno 1000 uzoraka pripada razredu *nema promjene*), tada mnogi klasifikatori rade tako da svaki uzorak s nepoznatim razredom pokušaju razvrstati kao negativan slučaj, pa će tako svaki pozitivni slučaj klasifikator razvrstati kao negativan, a točnost će biti 99% [38].

Drugi način evaluacije klasifikatora je korištenjem ROC krivulje (*Receiver Operating Characteristic*). ROC krivulja je graf gdje je na *x*-osi prikazana *mjera pogrešno predviđenih pozitivnih uzoraka*, a na *y*-osi prikazana *mjera točno predviđenih pozitivnih uzoraka*. Vrijednosti na obje osi su iz intervala [0, 1]. ROC krivulja sadrži sve informacije koje su predstavljene matricom konfuzije s obzirom da je mjera pogrešno predviđenih negativnih slučajeva komplement mjere točno predviđenih pozitivnih slučajeva, a mjera točno predviđenih negativnih slučajeva je komplement mjere pogrešno predviđenih pozitivnih slučajeva.
Primjeri ROC krivulja prikazana su na slici 5.1.



Slika 5.1. ROC krivulje

ROC krivulja mjeri odnos između točnih pozitivnih ishoda i pogrešnih pozitivnih ishoda za različite parametre sustava. Na slici 5.1. točka (0, 1) je savršeni klasifikator, zato što je mjera pogrešno predviđenih pozitivnih ishoda 0, a mjera točno predviđenih pozitivnih

ishoda 1, odnosno svi pozitivni ishodi su točno predviđeni. Točka (0, 0) predstavlja klasifikator koji sve slučajeve predviđa kao negativne, a (1, 1) je klasifikator koji sve slučajeve predviđa kao pozitivne. Točka (1, 0) predstavlja klasifikator koji sve slučajeve pogrešno razvrsta. S krivulje je moguće očitati povećanje mjere točno predviđenih pozitivnih slučajeva uz istovremeno povećanje pogrešno predviđenih pozitivnih slučajeva.
Krivulja se dobije tako da se svaka točka (PP, TP) dobije variranjem parametra sustava. Klasifikator koji nema parametara predstavljen je samo jednom točkom. Slika 5.1. prikazuje krivulje K_1 i K_2, dobivene za dva različita klasifikatora. Prema slici, klasifikator predstavljen krivuljom K_1 daje točnije rezultate.

6. Priprema podataka za analizu

U radu su korišteni eksperimentalno prikupljeni podaci koji potječu od projekta *Saccharomyces Genome Deletion Project* [30]. Kao izvor podataka još su korištene i baze podataka MEDLINE [31], *Saccharomyces Genome Database* [32] i MIPS [33]. Podaci su rezultati mjerenja djelovanja pojedinih gena kvasca na promatrani sustav organizma.

Djelovanje se pojedinog gena promatra mjerenjem djelovanja *soja kvasca* na promatrani biološki sustav. Soj kvasca je u odnosu na divlji tipa kvasca karakteriziran *nedostatkom jednog gena*. Ako se uklanjanjem gena pokaže promjena u promatranom sustavu, a istovremeno nema promjene u kontrolnom sustavu, smatra se da je taj gen zaslužan za rad promatranog biološkog sustava. Kontrolni sustav je neki drugi sustav u organizmu, a kreće se od pretpostavke da gen koji djeluje i na kontrolni sustav i na promatrani sustav djeluje na cijeli organizam [45].

Gen je kod u molekuli DNK koji kodira za protein. Genski se kod prepisuje u molekulu mRNK, a zatim se ta molekula pročita na ribosomu i prema tom se kodu sintetizira protein. Pretpostavka koja će se koristiti u ovome radu je da svaki gen *kodira za jedan protein*, odnosno da je djelovanje jednog gena vezano uz jedan protein, što se u biologiji smatra da je u najvećem broju slučajeva točno.

Podaci korišteni u praktičnom dijelu rada podijeljeni su u dva skupa: skup pokaznih uzoraka i skup testnih uzoraka. Svi uzorci opisani su mnoštvom atributa i mogu imati jednu od tri oznake razreda, ali će u radu biti razmatrana dva odvojena slučaja u kojima uzorci mogu pripadati jednom od dva razreda. Izgradit će se model razvrstavanja na temelju skupa pokaznih uzoraka, te će se provjeriti valjanost izgrađenog modela na testnom skupu uzoraka.

Razred kojemu pripada pojedini uzorak može imati jednu od sljedećih oznaka, gdje se oznaka razreda odnosi se na relativnu razinu djelovanja biološkog sustava koji se promatra:

1. oznaka *nc* (dolazi od eng. *no change*) – označava uzorak koji predstavlja soj kvasca koji ne pokazuje promjene u promatranom biološkom sustavu
2. oznaka *control* – označava uzorak koji predstavlja soj kvasca koji pokazuje utjecaj i na promatrani biološki sustav i na kontrolni biološki sustav
3. oznaka *change* - označava uzorak koji predstavlja soj kvasca koji pokazuje utjecaj na promatrani biološki sustav, ali ne pokazuje utjecaj na kontrolni biološki sustav

U teoriji postoje četiri odvojena slučaja (kao u tablici 5.1.). Međutim u praksi se eksperimenti obavljaju na sljedeći način: neka je podskup gena *H* identificiran tako da nedostatak gena iz tog podskupa znatno utječe na pad/rast pokazatelja promatranog biološkog sustava. Nakon toga se kontrolni sustav mjeri samo u odnosu na gene iz H. Zbog toga ne postoje informacije po kojima bi se razlučili prvi i drugi slučaj [34].

Tablica 6.1. Oznaka razreda temeljena na promjenama u promatranom i kontrolnom sustavu

Promjena u promatranom sustavu	Promjena u kontrolnom sustavu	Razred
0	0	NC
0	1	NC
1	0	CHANGE
1	1	CONTROL

U radu će se promatrati dva slučaja - djelovanje širokog skupa pozitivnog razreda i djelovanje uskog skupa pozitivnog razreda:

1. *široki skup pozitivnog razreda* čine svi uzorci koji inicijalno pripadaju razredu *change* ili *control* (promatra se djelovanje gena bilo na promatrani sustav, bilo na promatrani i kontrolni sustav istovremeno) i ti uzorci će svi imati oznaku razreda *change*
2. *uski skup pozitivnog razreda* čine svi uzorci koji inicijalno pripadaju isključivo razredu *change*, dok su uzorci koji pripadaju razredu *control* promatrani kao negativni (iz razreda *nc*) (promatra se djelovanje gena isključivo na promatrani sustav), pa će uzorci koji inicijalno pripadaju razredima *nc* i *change* imati oznaku razreda *nc*

6.1. Testni i pokazni skup podataka

Pokazni skup podataka sastoji se od 3018 uzorka, a *testni skup podataka* se sastoji od 1489 uzoraka. Oba skupa uzoraka imaju jednaku distribuciju razreda uzoraka. Na temelju pokaznog skupa uzoraka, izgradit će se model koji će se zatim testirati na testnom skupu uzoraka ispitivanjem da li izgrađeni model dobro predviđa razred pojedinog testnog uzorka. Korištenjem parametara ROC (*Receiver Operating Characteristic*) analize odredit će se koji je od primijenjenih postupaka klasifikacije dao najbolje rezultate.

Model će biti izgrađen korištenjem podataka o lokalitetu i funkcijama proteina, razredima proteina, međudjelovanju proteina, te korištenjem ključnih riječi iz sažetaka članaka koji se odnose na eksperimentalne radove vezane uz pojedine gene. S obzirom da su podaci prikupljeni eksperimentalnim putem oni uključuju i šum. Za neke gene u pokaznom i testnom skupu podataka nedostaje dosta podataka, ali su i ti uzorci uključeni u analizu kako bi se što zornije prikazalo stanje podataka kakvo je u stvarnosti.

6.2. Datoteke koje sadrže podatke

U ovom poglavlju prikazana je struktura datoteka koje čine izvor podataka [4].

6.2.1. Datoteka *train-class.txt*

Ova datoteka sadrži oznake razreda za 3018 pokaznih uzoraka. Svaki redak u datoteci predstavlja jedan uzorak.

Primjer retka u datoteci:

YMR142C change

Prvo polje u retku je identifikator gena (*YMR142C*), a drugo polje je oznaka razreda (*change*).

6.2.2. Datoteka *gene-abstracts.txt*

Ova datoteka sadrži pokazivače na sažetke iz znanstvenih radova koji su povezani uz pojedine sojeve kvasca, odnosno gene koji su identifikatori pojedinih sojeva kvasca.

Primjer retka u datoteci:

YML034W 10734188

Svaki redak u datoteci sadrži identifikator pojedinog gena (*YML034W*) i identifikator sažetka (*10734188*).

6.2.3. Datoteke koje sadrže sažetke

Datoteke koje sadrže tekst sažetaka referenciraju se na oznake sažetaka iz datoteke *gene-abstracts.txt*. Tekst svakog sažetka se nalazi u posebnoj datoteci, a imena datoteka su identifikatori sažetaka korišteni u datoteci *gene-abstracts.txt*. Sažeci su priskrbljeni iz baze podataka MEDLINE [31].

Veza između pojedinih gena i sažetaka je napravljena na jedan od dva načina: za neke gene u bazi podataka *Saccharomyces Genome Database* [32] postoje ručno postavljeni pokazivači na članke iz baze podataka MEDLINE. Za ove gene priskrbljeni su sažeci na koje pokazuju ovi pokazivači. Za ostale gene, sažeci su dobiveni temeljem upita nad bazom podataka [32] tako da su se za željeni gen i termin *Saccharomyces cerevisiae* dohvatili svi sažeci koji postoje u bazi.

6.2.4. Datoteka *interaction.txt*

Ova datoteka sadrži parove gena koji kodiraju za proteine koji fizički međudjeluju. Međudjelovanja proteina su simetrična, a redoslijed kojim su proteini navedeni u retku nije bitan. Ovi podaci su prikupljeni iz baze podataka MIPS [33].

Datoteka ne sadrži sva moguća međudjelovanja, a sadrži i neka međudjelovanja koja nisu točna. Međudjelovanje nije tranzitivno, tj. ako međudjeluju proteini *A* i *B*, te *B* i *C*, ne znači da *A* međudjeluje s *C*, ali može se zaključiti da su *A* i *C* povezani.

Primjer retka u datoteci:

YNL331C YNL331C

Svaki redak datoteke sadrži dva identifikatora gena (*YNL331C* i *YNL331C*) koji međudjeluju. U ovom primjeru riječ je o *homodimeru*, tj. proteinu koji djeluje sam na sebe.

6.2.5. Datoteka *localization.txt*

Ova datoteka sadrži podatke o lokalitetu pojedinih proteina unutar jezgre. Vrijednost lokaliteta koja se navodi uz pojedini gen može biti na bilo kojem nivou hijerarhije lokaliteta u jezgri. Ako lokalitet nije najviši u hijerarhiji, onda se navodi put do najvišeg lokaliteta u hijerarhiji gdje se nivoi lokaliteta u hijerarhiji odvajaju znakom "|". Ovi podaci su prikupljeni iz baze podataka MIPS [33].

Primjer retka u datoteci:

YGR072W cytoplasm

Prvo polje u datoteci predstavlja identifikator gena (*YGR072W*) koji kodira za protein, a drugo polje predstavlja lokalitet proteina u jezgri (*cytoplasm*). Moguće vrijednosti lokaliteta navedene su u datoteci *localization-hierarchy.txt*.

6.2.6. Datoteka *localization-hierarchy.txt*

Ova datoteka predstavlja hijerarhiju lokaliteta koji se koriste u datoteci *localization.txt*. Svaki redak predstavlja jedan čvor u hijerarhiji, gdje su čvorovi prikazani u odnosu roditelj-dijete.

Primjer hijerarhije:

peroxisome
    peroxisomal membrane

U primjeru je *peroxisome* nadređeni lokalitet od *peroxisomal membrane*.

6.2.7. Datoteka *function.txt*

Ova datoteka sadrži podatke o funkcijama koje obavljaju proteini. Neki proteini imaju više funkcija. Funkcija koja se navodi uz pojedini gen može biti na bilo kojem nivou hijerarhije funkcija. Ako funkcija nije najviša u hijerarhiji, onda se navodi put do najviše funkcije u hijerarhiji gdje se nivoi funkcija u hijerarhiji odvajaju znakom "|". Ovi podaci su prikupljeni iz baze podataka MIPS [33].

Primjer retka u datoteci:

YHR051W respiration|ENERGY

Prvo polje u datoteci predstavlja identifikator gena koji kodira za gen (YHR051W), a drugo polje je funkcija koju obavlja protein (*respiration|ENERGY*). Vrijednosti koje se mogu pojaviti kao drugo polje nalaze se u hijerarhiji funkcija koje obavljaju proteini, a navedeni su u datoteci *function-hierarchy.txt*.

6.2.8. Datoteka *function-hierarchy.txt*

Ova datoteka predstavlja hijerarhiju funkcija koje se koriste u *function.txt*. Svaki redak predstavlja jedan čvor u hijerarhiji, gdje su čvorovi prikazani u odnosu roditelj-dijete.

Primjer hijerarhije:

METABOLISM
    amino acid metabolism
        amino acid biosynthesis

U primjeru funkcija *METABOLISM* je nadređena funkciji *amino acid metabolism*, a funkcija *amino acid metabolism* je nadređena funkciji *amino acid biosynthesis*.

6.2.9. Datoteka *pc.txt*

Ova datoteka sadrži razrede kojima pripadaju proteini. Vrijednosti koje se mogu pojaviti kao razred kojemu pripada protein nalaze se u hijerarhiji opisanoj u datoteci *pc-hierarchy.txt*. Vrijednosti su prikupljene iz baze podataka MIPS [33].

Primjer retka:

YHR205W AGC group|Protein Kinases

Prvo polje datoteke je identifikator gena koji kodira za protein (*YHR205W*), a drugo polje je razred kojemu pripada protein (*AGC group|Protein Kinases*).

6.2.10. Datoteka *pc-hierarchy.txt*

Datoteka opisuje hijerarhiju vrijednosti koje se koriste u *pc.txt*. Svaki redak predstavlja jedan čvor u hijerarhiji razreda proteina. Čvorovi su povezani po principu roditelj-dijete.

Primjer hijerarhije:

Cyclins
    CLNs
    CLBs

U primjeru *Cyclins* je razred proteina nadređen razredima proteina *CLBs* i *CLNs*.

6.2.11. Datoteka *gene-aliases.txt*

Datoteka sadrži sinonime (aliase) za nazive svih gena iz podatkovnog skupa. Ovi sinonimi se koriste u analizi tekstova sažetaka. Neki geni koriste samo jedan sinonim, a ponekad se isti sinonim odnosi na nekoliko gena koji pripadaju istoj obitelji gena.

Primjer retka:

YHR124W NDT80

U primjeru gen *YHR124W* ima sinonim *NDT80*.

6.3. Struktura podataka

Alat korišten za dubinsku analizu podataka je programski alat WEKA [39]. Zbog zahtjeva na strukturu ulaznih datoteka za taj programski paket, podatke je trebalo pripremiti na odgovarajući način. Zahtijevani format ulazne tekstualne datoteke je ARFF format (*Attribute-Relation File Format*) [40]. Ovakav format datoteke uključuje navođenje popisa atributa (navode se nazivi i tipovi atributa), te navođenje tipa i naziva razreda (uobičajeno nakon popisa atributa). Tip atributa i tip razreda može biti numerički, znakovni ili nominalni tip, gdje je za nominalni tip atributa potrebno navesti koje vrijednosti atribut može poprimiti.

Iza popisa atributa navodi se u matričnom obliku popis uzoraka s vrijednostima atributa. Jedan se redak datoteke odnosi na jedan uzorak, a u svakom retku vrijednosti pojedinih atributa su odvojene zarezom. Redoslijed vrijednosti atributa za svaki uzorak mora odgovarati redoslijedu navedenom u popisu atributa.

Svaki se uzorak u ARFF datoteci odnosi na jedan gen. Atributi uzoraka su sva obilježja koja se odnose na gene, a određeni su prema podacima opisanima u poglavlju 6.2.

6.3.1. Atributi koji opisuju uzorke (gene)

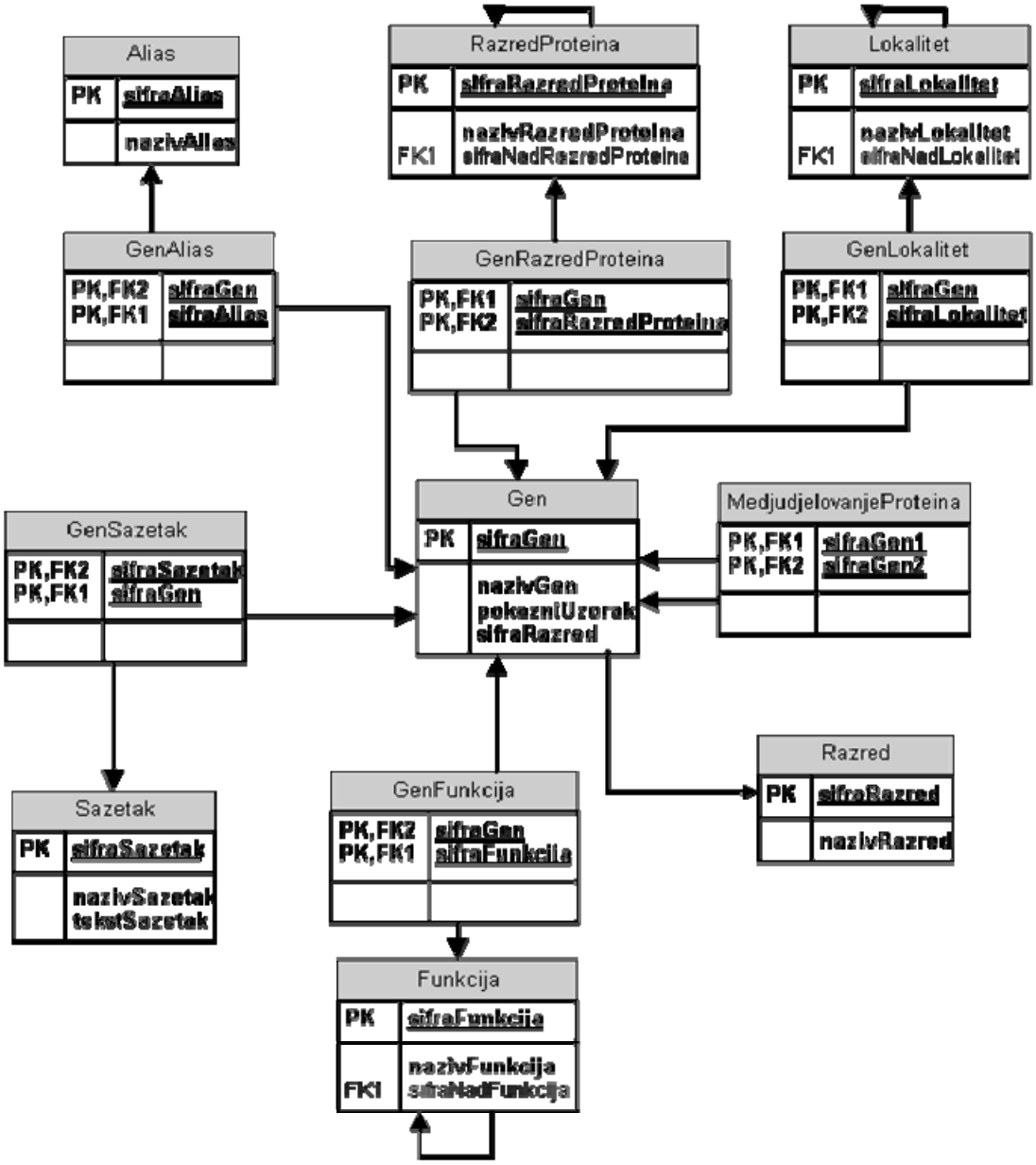
Pretpostavka koja je korištena jest da jedan gen kodira za jedan protein. Time su sva svojstva koja opisuju funkcije, lokalitete, razrede proteina ili međudjelovanje proteina vezana uz gen koji kodira za taj protein. Na temelju prikupljenih podataka formirane su sljedeće skupine atributa koje opisuju uzorke (gene):

1. atributi koji predstavljaju lokalitete proteina unutar stanice (prema podacima iz datoteke *localization.txt*)
2. atributi koji predstavljaju razrede proteina kojima pripadaju pojedini proteini (prema podacima iz datoteke *pc.txt*)

3. atributi koji predstavljaju funkcije koje proteini obavljaju (prema podacima iz datoteke *function.txt*)
4. atributi koji predstavljaju gene koji kodiraju za proteine koji međudjeluju s proteinima drugih gena (prema podacima iz datoteke *interaction.txt*)
5. atributi koji predstavljaju ključne riječi koje se pojavljuju u sažecima članaka o genima (prema podacima iz datoteke *gene-abstracts.txt*)

Prema opisanim skupinama atributa formirana je normalizirana baza podataka *Kvasac*. Tablice *Funkcija*, *Lokalitet* i *RazredProteina* sadrže popis svih funkcija, lokaliteta i razreda proteina, a tablice *GenFunkcija*, *GenLokalitet* i *GenRazredProteina* prate vezu pojedinog gena i neke funkcije, lokaliteta ili razreda proteina. Tablica *Sazetak* sadrži sažetke koji se odnose na gene, a veza između pojedinog sažetka i gena prikazana je u tablici *GenSazetak*. Međudjelovanje proteina opisano je u tablici *MedjudjelovanjeProteina*. Baza podataka napunjena je podacima iz datoteka opisanih u poglavlju 6.2.

Struktura normalizirane baze podataka prikazana je na slici 6.1. Kao sustav za upravljanje bazom podataka korišten je *Microsoft SQL Server 2000*.



Slika 6.1. Struktura baze podataka *Kvasac*

Međutim, kako normalizirana struktura baze podataka nije prikladna za dubinsku analizu podataka korištenjem programskog alata WEKA, potrebno je formirati odgovarajuću ulaznu matričnu strukturu podataka što je opisano u poglavljima 6.3.2-6.3.6. Radi pojednostavljenja, za nazive atributa zadržani su engleski nazivi.

6.3.2. Lokaliteti proteina

Svaki protein može biti smješten unutar stanice na jednom ili više lokaliteta, a svaki lokalitet može imati svoj nadređeni lokalitet.

Neka gen G_1 kodira za protein koji je smješten na lokalitetu L_1, čiji je nadređeni lokalitet L_2 (L_2 je lokalitet iz vrha hijerarhije lokaliteta L_1). U tablici *GenLokalitet* (na

slici 6.1.) postoji zapis ($sifraG_1$, $sifraL_1$), gdje je $sifraG_1$ šifra gena G_1, a $sifraL_1$ šifra lokaliteta L_1.

U ulaznoj datoteci za WEKA atribut koji predstavlja lokalitet L_1 nosi naziv $L\_L_1\_L_2$, a sastoji se od prefiksa $L\_$ što znači da atribut pripada skupini atributa koji se odnose na lokalitete; $L_1\_L_2$ znači da predstavlja lokalitet naziva L_1 čiji je lokalitet iz vrha hijerarhije naziva L_2. Naziv atributa sadrži lokalitet iz vrha hijerarhije, jer je moguće da postoji više različitih, ali istoimenih lokaliteta na nižim nivoima hijerarhije koje se onda razlikuje prema hijerarhiji lokaliteta kojoj pripadaju. Lokalitet iz vrha hijerarhije ima naziv oblika L_L_2.

Primjer:
Lokalitet *mitochondrial inner membrane* ima nadređeni lokalitet *mitochondria*. Za taj se lokalitet formira atribut naziva:

L_mitochondrial inner membrane_mitochondria

Atributi koji predstavljaju lokalitete u ulaznoj datoteci za WEKA su nominalni i mogu poprimiti vrijednosti iz skupa {0, 1}. Ako je vrijednost atributa $L\_L_1\_L_2$ jednaka 1 to znači da je protein gena (koji je predstavljen uzorkom) smješten na tom lokalitetu, a inače je 0. Vrijednost 1 bit će i za sve one druge atribute koji predstavljaju lokalitete u hijerarhiji iznad lokaliteta L_1 sve do i uključivo L_2.

6.3.3. Funkcije proteina

Svaki protein može imati jednu ili više funkcija, a svaka funkcija može imati nadređenu funkciju.

Neka gen G_1 kodira za protein koji ima funkciju F_1, čija je nadređena funkcija F_2 (F_2 je funkcija iz vrha hijerarhije funkcije F_1). U tablici *GenFunkcija* (na slici 6.1.) postoji zapis ($sifraG_1$, $sifraF_1$), gdje je $sifraG_1$ šifra gena G_1, a $sifraF_1$ šifra funkcije F_1.

U ulaznoj datoteci za WEKA atribut koji predstavlja funkciju F_1 nosi naziv $F\_F_1\_F_2$, a sastoji se od prefiksa $F\_$ što znači da atribut pripada skupini atributa koji se odnose na funkcije proteina; $F_1\_F_2$ znači da predstavlja funkciju naziva F_1 kojemu je nadređena funkcija iz vrha hijerarhije naziva F_2. Naziv atributa sadrži funkciju iz vrha hijerarhije, jer je moguće da postoji više različitih, ali istoimenih funkcija koje se onda razlikuje prema hijerarhiji funkcija kojoj pripadaju. Funkcija iz vrha hijerarhije ima naziv oblika F_F_2.

Primjer:
Funkcija *transcriptional control* je u hijerarhiji funkcija *TRANSCRIPTION*. Za tu se funkciju formira atribut naziva:

F_transcriptional control_TRANSCRIPTION

Atributi koji predstavljaju funkcije u ulaznoj datoteci za WEKA su nominalni i mogu poprimiti vrijednosti iz skupa {0, 1}. Ako je vrijednost atributa $F\_F_1\_F_2$ jednaka 1 to znači da protein gena (koji je predstavljen uzorkom) ima tu funkciju, a inače je 0. Vrijednost 1 bit će i za sve one druge atribute koji predstavljaju funkcije u hijerarhiji iznad funkcije F_1 sve do i uključivo F_2.

6.3.4. Razredi proteina

Jedan protein može biti iz jednog ili dva razreda, a svaki razred proteina može pripadati nadređenom razredu.

Neka gen G_1 kodira za protein iz razreda R_1, čiji je nadređeni razred proteina R_2 (R_2 je razred iz vrha hijerarhije razreda R_1). U tablici *GenRazredProteina* (na slici 6.1.) postoji zapis ($sifraG_1$, $sifraR_1$), gdje je $sifraG_1$ šifra gena G_1, a $sifraR_1$ šifra razreda proteina R_1.

U ulaznoj datoteci za WEKA atribut koji predstavlja razred R_1 nosi naziv $R\_R_1\_R_2$, a sastoji se od prefiksa *R_* što znači da atribut pripada skupini atributa koji se odnose na razrede proteina; $R_1\_R_2$ znači da predstavlja razred naziva R_1 kojemu je nadređeni razred iz vrha hijerarhije naziva R_2. Naziv atributa sadrži razred iz vrha hijerarhije, jer je moguće da postoji više različitih, ali istoimenih razreda koji se onda razlikuju prema hijerarhiji razreda kojoj pripadaju. Razred proteina iz vrha hijerarhije ima naziv oblika R_R_2.

Primjer:
Razred proteina *alpha* ima nadređeni razred iz vrha hijerarhije *Tubulins*. Za taj se razred formira atribut naziva:

R_alpha_Tubulins

Atributi koji predstavljaju razrede proteina u ulaznoj datoteci za WEKA su nominalni i mogu poprimiti vrijednosti iz skupa {0, 1}. Ako je vrijednost atributa $R\_R_1\_R_2$ jednaka 1 to znači da protein gena (koji je predstavljen uzorkom) pripada tom razredu, a inače je 0. Vrijednost 1 bit će i za sve one druge atribute koji predstavljaju razrede proteina u hijerarhiji iznad razreda R_1 sve do i uključivo R_2.

6.3.5. Međudjelovanje proteina

U podacima o međudjelovanju postoje zapisi o 647 gena čiji proteini međudjeluju s drugim genima. Atributi koji predstavljaju međudjelovanje sadrže u svojem nazivu ime gena koji međudjeluje s drugim genima (ukupno su formirana 647 takva atributa). S obzirom da je broj gena koji interagiraju puno manji od ukupnog broja gena koji predstavljaju uzorke pri testiranju, pretpostavka je da nedostatak podataka za ostale gene ne znači da oni ne sudjeluju u interakciji s drugim genima, nego da su ti podaci nepoznati ili nedostupni. Za gene za koje nisu poznati podaci o interakciji s drugim genima, za sve vrijednosti atributa koji predstavljaju međudjelovanje imaju vrijednost *nepoznato*.

Svaki atribut koji predstavlja međudjelovanje je nominalan atribut i može poprimiti vrijednost iz skupa {0, 1, 2}. Ove vrijednosti mogu poprimiti samo oni geni koji interagiraju s barem jednim genom. Vrijednost 0 predstavlja oznaku da protein gena na koji se uzorak odnosi ne interagira s proteinom gena po kojem je nazvan atribut. Vrijednost 1 predstavlja oznaku da protein gena na koji se uzorak odnosi izravno interagira s proteinom gena po kojem je nazvan atribut, a vrijednost 2 predstavlja oznaku da protein gena na koji se uzorak odnosi posredno interagira s proteinom gena po kojem je nazvan atribut (npr. ako međudjeluju proteini *A* i *B*, te *B* i *C*, onda postoji i neka veza između *A* i *C,* iako tranzitivnost ne vrijedi).

6.3.6. Ključne riječi iz sažetaka

Sve riječi koje se pojavljuju u sažecima, a nisu brojevi, interpunkcijski ili specijalni znakovi kandidati su za ključne riječi koje bi mogle opisivati djelovanje gena. Prebrojane su sve riječi koje se pojavljuju u sažecima s tim da su izostavljene najčešće riječi engleskog jezika prema [43], [44] (s obzirom da su svi sažeci pisani engleskim jezikom), te riječi koje su česte u biologiji. Lista učestalih riječi broji oko 2700 riječi (oko 2000 riječi iz svakodnevnog govora, a ostalo su riječi koje su česte u biologiji, te interpunkcijski i drugi posebni znakovi).

Od ukupnog broja dostupnih sažetaka pretraženo je njih 9825, koji se odnose na gene koji su zastupljeni u uzorcima. Svaki sažetak je tekst duljine do 4100 znakova.

Algoritam traženja riječi iz sažetaka prikazan je na slici 6.2.

```
Ulaz:
   Sažetak – skup sažetaka; svaki sažetak je tekst
   UčestaleRiječi – lista učestalih engleskih riječi

Izlaz:
   GenRiječ – skup listi riječi;
      jedan element je lista čiji su elementi <riječ,brojPojavljivanja>
   NeučestaleRiječi – lista neučestalih engleskih riječi koje se
                      pojavljuju u svim sažecima;
                      element liste je <riječ, brojPojavljivanja>

Postupak:
   za svaki sažetak ∈ Sažetak
      za svaku riječ ∈ sažetak
         ako riječ ∉ UčestaleRiječi
            ako riječ ∈ NeučestaleRiječi
               povećaj broj pojavljivanja te riječi za 1
            inače
               dodaj riječ u NeučestaleRiječi i brojač pojavljivanja
                  postavi na 1

         za svaki gen na koji se odnosi sažetak
            ako postoji zapis u GenRiječ za gen i riječ
               povećaj broj pojavljivanja riječi riječ za gen za 1
            inače
               dodaj riječ u listu riječi za gen u GenRiječ i postavi
                  broj pojavljivanja riječi riječ za gen na 1
```

Slika 6.2. Algoritam brojanja ključnih riječi u sažecima

Korištenjem algoritma prikazanog na slici 6.2. generirane su datoteke *GenRijec.txt* (koja sadrži oko 240000 zapisa) i *NeucestaleRijeci.txt* (koja sadrži oko 63000 zapisa). Trajanje izvođenja je oko 5 minuta.

Datoteka *GenRijec.txt* formirana je prema listi `GenRiječ`, a sadrži zapise oblika:

šifraGen#riječ_iz_sažetka#broj pojavljivanja riječi za gen u svim sažecima

Primjer:

5506#plasma#2

što znači da se za gen sa šifrom *5506* u sažecima riječ *plasma* pojavljuje *2* puta.

Datoteka *NeucestaleRijeci.txt* sadrži zapise oblika:

riječ_iz_sažetka#broj pojavljivanja riječi u svim sažecima za sve gene

Primjer:

zymocin#9

što znači da se u svim sažecima riječ *zymocin* pojavila *9* puta.

Za atribute iz kategorije ključnih riječi, uzeto je 1390 riječi koje se pojavljuju između 50 i 1000 puta (slično kao u [41]). Od oko 63000 riječi 419 njih se pojavljuje 100 ili više puta (od kojih se svega nekoliko riječi pojavljuje više od 1000 puta), a 1397 riječi se pojavljuje 50 ili više puta. Time su izbačene riječi koje se pojavljuju prečesto ili prerijetko, a njihov broj bi dodatno povećao ionako velik broj atributa koji opisuju uzorke.

Atribut koji će predstavljati ključnu riječ u nazivu atributa za ulaznu datoteku za WEKA sadržavat će prefiks *S_* (koji upućuje da se radi o atributa koji dolazi kao riječ iz sažetka) i ključnu riječ. Npr. za riječ *topoisomerase* bit će formiran atribut *S_topoisomerase*.

Vrijednost atributa koji predstavljaju ključne riječi može biti iz skupa {0, 1}. Ako se za neki gen riječ iz naziva atributa pojavila barem jedanput u sažecima vrijednost atributa bit će 1, a inače će biti 0.

U ovom se radu koristi nominalni prikaz vrijednosti atributa za sve atribute, pa tako i za one koji predstavljaju tekst (binarni prikaz). To je najjednostavniji način prikaza, te svi algoritmi klasifikacije mogu raditi s nominalnim vrijednostima. Osim jednostavnosti, takav je pristup lakši i brži, nego ako atributi koriste numeričke vrijednosti što je slučaj u ostalim načinima koji se koriste za prikaz atributa koji predstavljaju tekst (numerički prikaz, *tf-idf* prikaz i *Hadamard* prikaz [54]).

6.4. Nenormalizirane tablice koje sadrže sve atribute koje opisuju gene

Prema podacima iz normaliziranih tablica (slika 6.1.) izgrađene su nenormalizirane tablice koje sadrže kategorije atributa opisane u poglavljima 6.3.2. do 6.3.6. Ove tablice objedinjuju podatke o genima, a predstavljaju matrični prikaz kakav se očekuje za ulaz za Weka-u. Reci tablica su geni, a stupci su atributi. Takvom strukturom je omogućeno jednostavno prebacivanje sadržaja tih tablica u datoteke kakve su potrebne kao ulaz za Weka-u.

Ukupan broj atributa koji se pojavljuju u ovim tablicama je 2527, pa nije bilo moguće objediniti sve atribute u jednu tablicu zbog ograničenja *SQL Servera 2000* da broj atributa u jednoj tablici može biti najviše 1024. Zbog ograničenja SUBP-a, brzine i logičke jednostavnosti (tematska podjela) podaci su razdijeljeni u pet tablica (*F_L_RP*, *MEDJUDJELOVANJE*, *GenRijec101do1000*, *GenRijec50do100_1* i *GenRijec50do100_2*).

6.4.1. Tablica F_L_RP

Tablica *F_L_RP* je nenormalizirana tablica koja sadrži šifru i naziv gena (odgovaraju atributima *sifraGen* i *nazivGen* iz tablice *Gen* sa slike 6.1.), te kao ostale atribute sve funkcije proteina iz tablice *Funkcija* (259 atributa), lokalitete proteina iz tablice *Lokalitet* (42 atributa) i razrede proteina iz tablice *RazredProteina* (188 atributa) (struktura tablice prikazana je na slici 6.3.). Prefiks atributa koji predstavljaju funkcije proteina je *F_*, prefiks atributa koji predstavljaju lokalitete proteina je *L_*, a prefiks atributa koji predstavljaju razreda proteina *R_*. Svaki atribut može imati vrijednost 0 ili 1, kako je objašnjeno u poglavljima 6.3.2. do 6.3.4. Ako za neki atribut nije poznata vrijednost funkcije (ili lokalitet ili razred proteina), onda se automatski vrijednosti svih atributa koji predstavljaju funkcije (ili lokalitete ili razrede proteina) postavi na 0.

Imena atributa su zbog ograničenja skraćena na 128 znakova (to znači da imena nekih atributa sadrže kraći naziv nadređene funkcije/lokaliteta/razreda proteina iz pripadajuće hijerarhije). Ova tablica ima ukupno 489 atributa.

Tablica 6.2. Tablica *F_L_RP*

		Funkcije proteina			***Lokaliteti proteina***			***Razredi proteina***		
sifraGen	**nazivGen**	**F_1**	**..**	**F_259**	**L_1**	**...**	**L_42**	**R_1**	**...**	**R_188**
1000	YAL012W	0	0	1	0	0	1	0	0	0
1001	YAL016W	1	0	0	0	0	0	0	0	0

6.4.2. Tablica MEDJUDJELOVANJE

Tablica *MEDJUDJELOVANJE* je nenormalizirana tablica koja sadrži kao atribute atribut *nazivGenM*, te atribute čiji je naziv ime gena čiji proteini djeluju s drugim genima (647 atributa) (tablica 6.3.). Svaki redak predstavlja međudjelovanje proteina jednog gena (njegov je naziv *nazivGenM*) s proteinima gena koji predstavljaju ostali atributi. S obzirom da za većinu gena nisu poznati podaci o međudjelovanju, uveden je poseban redak za opis takvih gena. Taj zapis za vrijednosti svih atributa sadrži oznaku *?*, koja u ARFF datotekama predstavlja oznaku za nepoznat podatak.

Tablica 6.3. Tablica *MEDJUDJELOVANJE*

nazivGenM	**Gen1**	**...**	**GenN**
YAL012W	1		0
?	?		?

6.4.2.1 Algoritam formiranja tablice međudjelovanja

Tablica *MEDJUDJELOVANJE* se puni podacima iz matrice međudjelovanja proteina MP čije je oblikovanje prikazano na slici 6.4.

Pretpostavka: ako protein gena *A* interagira s proteinom gena *B*, a protein gena *B* interagira s proteinom gena *C*, tada postoji i neki oblik interakcije između proteina gena *A* i gena *C*. Ta veza nije tranzitivna, ali djelovanje postoji, pa će takva djelovanja također biti uzeta u obzir.

Neka je gen *A* predstavljen retkom tablice *MEDJUDJELOVANJE* (n-torka *t*). Ako protein gena *A* interagira s proteinom gena *B*, koji je predstavljen stupcem (atributom) tablice, onda će vrijednost *t*(*B*) biti jednaka 1. Ako protein gena *A* interagira s proteinom gena *B*, a protein gena *B* interagira s proteinom gena *C*, gdje je *C* također atribut tablice *MEDJUDJELOVANJE*, onda će vrijednost *t*(*C*) biti 2. Ako ne postoji niti

izravno, niti posredno međudjelovanje proteina atributa *A* i proteina gena *D*, onda će vrijednost *t*(*D*) biti 0.

Algoritam za formiranje tablice međudjelovanja prikazan je na slici 6.3. Algoritam je sljedeći: na temelju zadanih međudjelovanja između proteina pojedinih gena, odredit će se koji geni posredno djeluju s drugim genima. Vrijednosti međudjelovanja između pojedinih gena pohranjuje se u matricu *MP* koja je dimenzija *N* x *N* (*N* broj gena koji interagiraju). Vrijednost polja *MP(i, j)* može biti 0, 1 ili 2. Ako je vrijednost polja 1, to znači da proteini *i*-tog i *j*-tog gena međudjeluju; ako je vrijednost polja 2, onda to znači da proteini *i*-tog i *j*-tog gena posredno djeluju, a vrijednost 0 znači da proteini *i*-tog i *j*-tog gena ne djeluje niti izravno, niti posredno.

Za *i*-ti gen se formira lista `Lista(i)` koja sadrži podatke o međudjelovanju tog gena s drugim genima. Svaki element liste je uređena trojka `(j, oznakaDjelovanja, oznakaListaPregledana)` gdje je:

- `j` – indeks gena s kojim gen `i` interagira
- `oznakaDjelovanja` - oznaka da li gen `j` djeluje izravno (1) ili posredno (2) na gen `i`
- `oznakaListaPregledana` – oznaka da li je za gen `j` pregledana njegova lista; ako `Lista(j)` nije pregledana, onda se u listu `Lista(i)` dodaju elementi liste `Lista(j)` koji već nisu u `Lista(i)` s oznakom `oznakaDjelovanja` = 2.

Liste svih gena zajedno čine polje listi `Lista` koje ima `N` elemenata.

Inicijalno za svaki gen `i` pripadajuća `Lista(i)` sadrži samo one gene `j` s kojima gen `i` izravno interagira, tj. inicijalno lista sadrži samo trojke oblika: `(j, 1, 0)`, gdje je `oznakaDjelovanja` = 1 i `oznakaListaPregledana` = 0.

Za svaki gen `i` pregledava se njegova lista, koja predstavlja sve gene `j` s kojima gen izravno ili posredno međudjeluje. Ako lista gena `j` nije još pregledana, onda se svi elementi iz te `Lista(j)`, koji nisu u `Lista(i)`, dodaju u `Lista(i)`, kao oni geni s kojima `i`-ti gen posredno interagira. Pregledavanje liste `i`-tog gena se zaustavlja kada u listi ne postoji više niti jedan gen čija lista nije pregledana, odnosno, kada više ne postoji niti jedan element za koji je `oznakaListaPregledana` jednaka 0.

Algoritam se zaustavlja kada se pregledaju liste svih gena.

```
Ulaz:
   ListaGen – lista naziva gena koji interagiraju
   N – broj gena koji interagiraju s barem jednim genom

Izlaz:
   MP – matrica međudjelovanja proteina dimenzija N x N
   Lista – N listi gdje svaka lista sadrži elemente u obliku
           uređene trojke: (j, oznakaDjelovanja, oznakaListaPregledana)
Postupak:
   // inicijalizacija matrice međudjelovanja
   za svaki i = 0 do N - 1 // po genima
      za svaki j = i do N – 1 // po genima
         ako gen i i gen j interagiraju
            MP(i, j) := MP(j, i) := 1
         inače
            MP(i, j) := MP(j, i) := 0
```

```
// inicijalizacija listi gena Lista
za svaki i = 0 do N - 1 // po genima
   Lista(i) := {(j, 1, 0) | MP(i, j) = 1}

// ažuriranje listi gena
za svaki gen i = 0 do N - 1 // po genima

   // sređivanje liste i-tog gena
   za svaki gen j iz Lista(i) za koji je
      oznakaListaPregledana = 0 i oznakaListaDjelovanja > 0
      dohvati Lista(j)

      ListaNovi := Lista(j) \ Lista(i)

      za svaki gen k iz ListaNovi
         oznakaDjelovanja := 2
         oznakaListaPregledana := 0

      Lista(i) := Lista(i) U ListaNovi

   // kad se pregleda Lista(i)
   oznakaListaPregledana := 1

   // ažuriranje matrice MP s podacima o posrednom djelovanju
   za svaki gen j iz Lista(i) za koji je oznakaDjelovanja := 2
      MP(i, j) := MP(j, i) := 2
```

Slika 6.3. Algoritam punjenja matrice međudjelovanja proteina

6.4.3. Tablice koje kao nazive atributa sadrže ključne riječi iz sažetaka

Tablice *GenRijec101do1000*, *GenRijec50do100_1* i *GenRijec50do100_2* sadrže kao atribute šifru gena, te ključne riječi koje su se u sažecima pojavile između 50 i 1000 puta. Zbog ograničenja SQL Serveru 2000 da je broj stupaca u jednoj tablici najviše 1024, a ključnih riječi koje se pojavljuju između 50 i 1000 puta ima 1389, atributi nisu u jednoj, nego su podijeljeni u 3 tablice. U tablici *GenRijec101do1000* nalaze se kao atributi ključne riječi koje se pojavljuju između 101 i 1000 puta u svim sažecima (403 riječi). U tablici *GenRijec50do100_1* se nalaze ključne riječi koje se pojavljuju između 50 i 100 puta i nalaze se abecedno do uključivo slova M (567 riječi) i u tablici *GenRijec50do100_2* se nalaze ključne riječi koje se pojavljuju između 50 i 100 puta i nalaze se abecedno od slova N do Z (420 riječi).

Svaka od tablica koje sadrže ključne riječi može se predočiti kao tablica 6.3. Za svaki gen koji je predstavljen n-torkom t i svaku ključnu riječ koja je predstavljena atributom A, vrijednost $t(A)$ je broj pojavljivanja te ključne riječi u sažecima koja se odnose na taj gen.

Tablica 6.4. Tablice koje kao nazive atributa sadrže ključne riječi iz sažetaka

sifraGen	Riječ1	...	RiječN
1000	34		0
1001	0		540

6.4.4. Formiranje ARFF datoteka

Format ARFF datoteke uključuje navođenje popisa atributa (navode se nazivi i tipovi atributa), te navođenje tipa i naziva razreda (uobičajeno se navodi nakon popisa atributa), što je detaljnije objašnjeno u poglavlju 6.3. Atributi koji se koriste u analizi su šifra i naziv gena, te svi atributi koji predstavljaju funkcije, lokalitete i razrede proteina, zatim atributi koji predstavljaju ključne riječi, te kao posljednji atribut navodi se razred. Atributi su dobiveni iz tablica *F_L_RP*, *MEDJUDJELOVANJE* i tablica koje se odnose na ključne riječi iz sažetaka, a ukupni broj atributa (bez šifre gena i razreda kojemu uzorak pripada) je 2527. Matrični ulaz kakav se očekuje kao ulazna datoteka za WEKA može se pojednostavljeno predočiti kao na slici 6.5.

Tablica 6.5. Matrični prikaz skupa uzoraka

		Funkcije proteina	***Lokaliteti proteina***	***Razredi proteina***	***Ključne riječi***	
sifraGen	**nazivGen**	**F_1 ...**	**L_1 ...**	**R_1 ...**	**Riječ1...**	**Razred**
1000	YAL012W	0	0	1	1	0
1001	YAL016W	1	1	0	0	1

Na temelju zadanog skupa uzoraka formiraju se dvije datoteke: datoteka s pokaznim uzorcima (*train.arff*) i datoteka s testnim uzorcima (*test.arff*), gdje svaka datoteka sadrži šifru gena, listu atributa iz gornjih tablica, te razred kojemu uzorak pripada.

7. Eksperimentalni rezultati

Zadani se skup uzoraka sastoji od 4507 uzoraka, gdje svaki uzorak predstavlja jedan gen opisan s 2527 atributa (struktura tih atributa opisana je u poglavlju 6.3.), te je za svaki uzorak poznat razred kojemu uzorak pripada. Cilj je na temelju tih podataka izgraditi model kojim će se za nepoznate uzorke (gene) moći predvidjeti kojemu razredu pripadaju. Model će se izgraditi korištenjem algoritama razvrstavanja. Kao što je uobičajeno kod algoritama razvrstavanja, otprilike dvije trećine cijelog skupa uzoraka koristit će se za izgradnju modela (pokazni skup uzoraka), a preostali dio skupa uzoraka koristit će se za provjeru modela (testni skup uzoraka). Odabran je pokazni skup od 3018 uzoraka i testni skup od 1489 uzoraka.

Svi su atributi koji opisuju uzorke, osim šifre i naziva gena, nominalni atributi. Atributi koji označavaju funkciju, lokalitet, razred proteina ili ključnu riječ iz sažetka mogu imati vrijednosti iz skupa {0, 1}. Za atribute koji označavaju međudjelovanje proteina vrijednosti atributa može biti iz skupa {0, 1, 2}. Posebno je analizirano kada vrijednost atributa koji označava međudjelovanje ima vrijednost iz skupa {0, 1, 2}, a posebno ako se vrijednost 2 zamijeni s 0 ili 1, tj. ako se posredno djelovanje između proteina zamijeni s oznakom nedjelovanja ili oznakom izravnog djelovanja.

Promatrat će se pripadnost uzoraka pozitivnom i negativnom razredu, gdje oznaka *change* predstavlja pozitivni razred (značenje: gen djeluje na ispitivani biološki sustav), dok oznaka *nc* predstavlja negativni razred (značenje: gen ne djeluje na ispitivani biološki sustav). Kao što je navedeno u 6. poglavlju, svi uzorci inicijalno imaju oznaku razreda iz skupa {*change*, *nc*, *control*}, ali je oznaka *control* zamijenjena s oznakom *nc* ili *change*, ovisno da li se promatra široko ili usko djelovanje gena:

1. *široki skup pozitivnog razreda* čine svi uzorci koji inicijalno pripadaju razredu *change* ili *control* (promatra se djelovanje gena bilo na promatrani sustav, bilo na promatrani i kontrolni sustav istovremeno)
    a. pokazni skup uzoraka: razredu *change* pripadaju 84 uzorka, a razredu *nc* pripadaju 2934 uzoraka
    b. testni skup uzoraka: razredu *change* pripadaju 43 uzorka, a razredu *nc* pripadaju 1446 uzoraka
2. *uski skup pozitivnog razreda* čine svi uzorci koji inicijalno pripadaju isključivo razredu *change*, dok su uzorci koji pripadaju razredu *control* promatrani kao negativni (iz razreda *nc*) (promatra se djelovanje gena isključivo na promatrani sustav)
    a. pokazni skup uzoraka: razredu *change* pripada 38 uzoraka, a razredu *nc* pripada 2980 uzoraka
    b. testni skup uzoraka: razredu *change* pripada 19 uzoraka, a razredu *nc* pripada 1470 uzoraka

7.1. Problemi koji karakteriziraju skup uzoraka

Kao prvi problem uočljiv je vrlo velik skup atributa kojim je opisan skup uzoraka (oko 2500 atributa). Zbog toga je nužno smanjiti skup atributa na dobro odabran podskup nad kojim bi se testirali algoritmi razvrstavanja. Općenito vrijedi da algoritmi razvrstavanja daju bolje rezultate (veću točnost) ako rade s dobro odabranim podskupom

atributa u odnosu na cijeli skup atributa [14], [15], [16], [17], [29]. Algoritmi smanjenja broja atributa uklanjaju i međusobno ovisne, odnosno redundantne, atribute. Osim veće točnosti pri razvrstavanju, smanjeni broj atributa pridonosi puno bržem dobivanju rezultata, što se posebno odnosi na algoritme kojima vrijeme razvrstavanja eksponencijalno ovisi o broju atributa.

S obzirom da su podaci prikupljeni eksperimentalno, kao drugi problem javlja se nepotpunost podataka, tj. za mnoge uzorke vrijednosti atributa su nepoznate (eng. *missing values*), te šum u podacima. Vrijednosti atributa koje nisu poznate zamijenjene su posebnom oznakom za nepoznat podatak. Pri odabiru podskupa početnog skupa atributa, nepoznate vrijednosti određene su prema distribuciji poznatih vrijednosti za te atribute.

Kao najveći problem koji karakterizira skup uzoraka pokazat će se nerazmjer između broju uzoraka koji pripadaju pozitivnom razredu u odnosu na broj uzoraka koji pripadaju negativnom razredu. U širokom skupu pozitivnog razreda 2.7% uzoraka pokaznog skupa pripadaju pozitivnom razredu, dok je u uskom skupu pozitivnog razreda svega 1.3% uzoraka pokaznog skupa u pozitivnom razredu. Taj nerazmjer je utjecao da svi korišteni algoritmi razvrstavanja, osim algoritma VFI [8], čak i nakon što bi se ulazni skup atributa smanjio na nekoliko stotina ili nekoliko desetaka atributa, *sve* testne uzorke razvrstaju u negativan razred. Razlog tome je što se time postiže najveća točnost klasifikatora, dok je točnost predviđanja pozitivnog razreda za sve algoritme 0% ili tek nešto malo više od toga, iako je zapravo predviđanje pozitivnog razreda ključan podatak. Zbog toga su korišteni algoritmi razvrstavanja (osim algoritma VFI) morali biti prilagođeni za rad s ovako specifičnim skupom podataka kako bi dali prihvatljive rezultate.

7.2. Postupak razvrstavanja

Odabir relevantnog podskupa atributa, te testiranje algoritama razvrstavanja obavljeni su korištenjem programskog alata Weka [39].

7.2.1. Odabir relevantnog podskupa skupa atributa

S obzirom na problem velikog broj atributa kojim je opisan skup uzoraka, prije samog postupka razvrstavanja potrebno je provesti odabir relevantnog podskupa skupa atributa.

Dva su osnovna pristupa u smanjivanju dimenzijske složenosti: metoda filtra i metoda omotača. Metoda omotača provjerava prikladnost odabranog podskupa atributa korištenjem algoritma razvrstavanja. Metoda filtra odabire podskup atributa neovisno o postupku razvrstavanja, tj. za evaluaciju odabranog podskupa koriti druge metode. Ove su metode detaljnije objašnjenje u 3. poglavlju. Pokazalo se da je za skup podataka kakav je karakterističan za biološke podatke, a to je skup uzoraka s vrlo velikim brojem atributa, filtarski pristup jedino prihvatljivo rješenje, dok je pristup odabira atributa korištenjem metode omotača vrlo spor ili potpuno neprihvatljiv (pojašnjeno u poglavlju 3.7.).

Kako je u konkretnom slučaju vrlo velik prostor svih podskupova atributa (oko 2^{2500} mogućih podskupova), čak i mnoge filtarske metode koje koriste heuristički ili slučajni način pretraživanja nisu prihvatljive. Zbog toga je za zadani skup uzoraka

odabir podskupa atributa razdvojen u dva koraka (slično kao u [42]), gdje je u oba koraka korišten filtarski pristup u odabiru podskupa atributa.

Neka je s A označen početni skup svih atributa koji opisuju uzorke (ukupno 2527 atributa, bez šifre i naziva gena). Korištena je metoda odabira relevantnog podskupa skupa atributa u dva koraka:

(i) odabir podskupa A_1 – podskup čine oni atributi čija je informacijska dobit (određena na način opisan u poglavlju 4.4.2.) veća od zadanog praga
(ii) odabir podskupa A_2 – podskup čine oni atributi iz podskupa A_1 koji su odabrani algoritmom *odabir prvog najboljeg* (eng. *best first search*)

U koraku (i) skup atributa A (koji sadrži više od 2500 atributa) svodi se na nekoliko stotina atributa koji čine podskup A_1, a odabrani su oni atributi čija je informacijska dobit ≥ 0.001. Za atribute čije vrijednosti nedostaju u pojedinim uzorcima (zapisima), nepoznate su vrijednosti određene prema distribuciji poznatih vrijednosti za te atribute.

Smanjeni podskup A_1 se u koraku (ii) pretražuje metodom odabira prvog najboljeg (eng. *best first search*) čime se formira novi podskup A_2 koji ima nekoliko desetaka atributa.

Nakon što se odabere podskup relevantnih atributa A_2, na temelju pokaznog skupa uzoraka s atributima samo iz skupa A_2 izgradit će se model razvrstavanja, a zatim će se na izgrađenom modelu provjeriti rezultati dobiveni za razvrstavanje testnog skupa uzoraka s atributima samo iz skupa A_2.

Za usporedbu rezultata dobivenih podskupom A_2 koristit će se i podskup A_3 koji je sačinjen od 50 atributa s najvećom informacijskom dobiti.

7.2.2. Utjecaj vrijednosti atributa koji se odnose na međudjelovanje proteina na rezultat razvrstavanja uzoraka

Utjecaj međudjelovanja proteina na rezultate razvrstavanja promatran je kroz tri slučaja:

(i) isključeno posredno međudjelovanje proteina
(ii) posredno međudjelovanje proteina ravnopravno s izravnim međudjelovanjem proteina
(iii) posredno međudjelovanje proteina je uključeno u analizu, ali se promatra kao zasebna kategorija (odvojeno od izravnog međudjelovanja)

Slučaj (i) je karakteriziran oznakama 0 i 1 kao mogućim vrijednostima atributa koji označava međudjelovanje. Oznaka 1 označava izravno međudjelovanje dvaju proteina za koje kodiraju dva gena (gen predstavljen retkom i gen predstavljen stupcem, odnosno atributom), a oznaka 0 označava slučaj kada proteini dvaju gena međusobno ne djeluju ili djeluju posredno.

Slučaj (ii) je karakteriziran oznakama 0 i 1 kao mogućim vrijednostima atributa koji označava međudjelovanje. Oznaka 1 označava izravno ili posredno međudjelovanje dvaju proteina za koje kodiraju dva gena (gen predstavljen retkom i gen predstavljen stupcem, odnosno atributom), a oznaka 0 označava slučaj kada proteini dvaju gena međusobno ne djeluju.

Slučaj (iii) je karakteriziran oznakama 0, 1 i 2 kao mogućim vrijednostima atributa koji označava međudjelovanje. Oznaka 1 označava izravno međudjelovanje dvaju proteina za koje kodiraju dva gena (gen predstavljen retkom i gen predstavljen stupcem, odnosno atributom), oznaka 2 označava posredno međudjelovanje dvaju

proteina za koje kodiraju dva gena, a oznaka 0 označava slučaj kada proteini dvaju gena ne djeluju.

Slučajevi (i) do (iii) promatrani su zasebno za široki pozitivni razred uzoraka, a zasebno za uski pozitivni razred uzoraka. Pokazalo se da se podjednaki rezultati dobiju za slučajeve (ii) i (iii), a ta dva slučaja daju tek nešto bolje rezultate nego slučaj (i). Iz toga se može zaključiti da je korisno uključiti i posredno međudjelovanje kao oblik međudjelovanja proteina za koje geni kodiraju. Mala razlika između rezultata dobivenih u sva tri slučaja proizlazi vjerojatno iz nedostatka podataka o međudjelovanju proteina (skup uzoraka čini više od 4500 uzoraka, a uključeni su podaci za oko 650 kombinacija međudjelovanja, dok je za očekivati da u stvarnosti postoji više međudjelovanja). Zbog navedenih razloga, te jednostavnosti, daljnji rezultati bit će prikazani za slučaj (ii), tj. slučaj kada su kao posredno i izravno međudjelovanje ravnopravno uključeni.

7.2.3. Testirani algoritmi razvrstavanja

Metode razvrstavanja koje su korištene u testiranju su (metode su opisane u 4. poglavlju):

(i) algoritam VFI (*Voting Feature Interval*) [8]
(ii) naivni Bayesov algoritam [60], [5]
(iii) algoritam stabla odlučivanja C4.5 [48] (Wekina implementacija J48 [39])
(iv) algoritam SVM [59] (korištena linearna implementacija SMO [50])
(v) metoda *k*-najbližih susjeda [60], [5]

Testiranjem navedenih metoda nad opisanim skupom podataka (bez da su metode prethodno posebno prilagođene za rad sa skupom uzoraka u kojem izrazito prevladava jedan razred), pokazalo se da jedino algoritam VFI daje dobre rezultate. Za algoritam VFI rezultat točno predviđenih pozitivnih uzoraka je između 58% (za uski skup pozitivnog razreda) i gotovo 70% (za široki skup pozitivnog razreda). Rezultati dobiveni za druge metode (bez prilagodbe za rad sa skupom uzoraka u kojem izrazito prevladava jedan razred) su 0% ili tek nešto više od 0% točno predviđenih pozitivnih uzoraka.

7.2.4. Rezultati razvrstavanja za algoritam VFI

Rezultati razvrstavanja za algoritam VFI prikazani su u tablici 7.1. Prikazani su rezultati za podskupove A_1, A_2, A_3 čija je struktura objašnjena u poglavlju 7.2.1. Lako je uočljivo da manji, dobro odabran podskup od nekoliko desetaka atributa (A_2 ili A_3) daje bolje rezultate nego veći podskup od više stotina atributa (A_1) što je u skladu s [14], [15], [16], [17], [29]. Isto tako, viša točnost previđenih pozitivnih uzoraka prisutna je kod šireg skup uzoraka u odnosu na uski skup uzoraka, što je očekivano s obzirom na veći broj uzoraka koji pripadaju pozitivnom razredu kod širokog skupa uzoraka u odnosu na uski skup uzoraka.

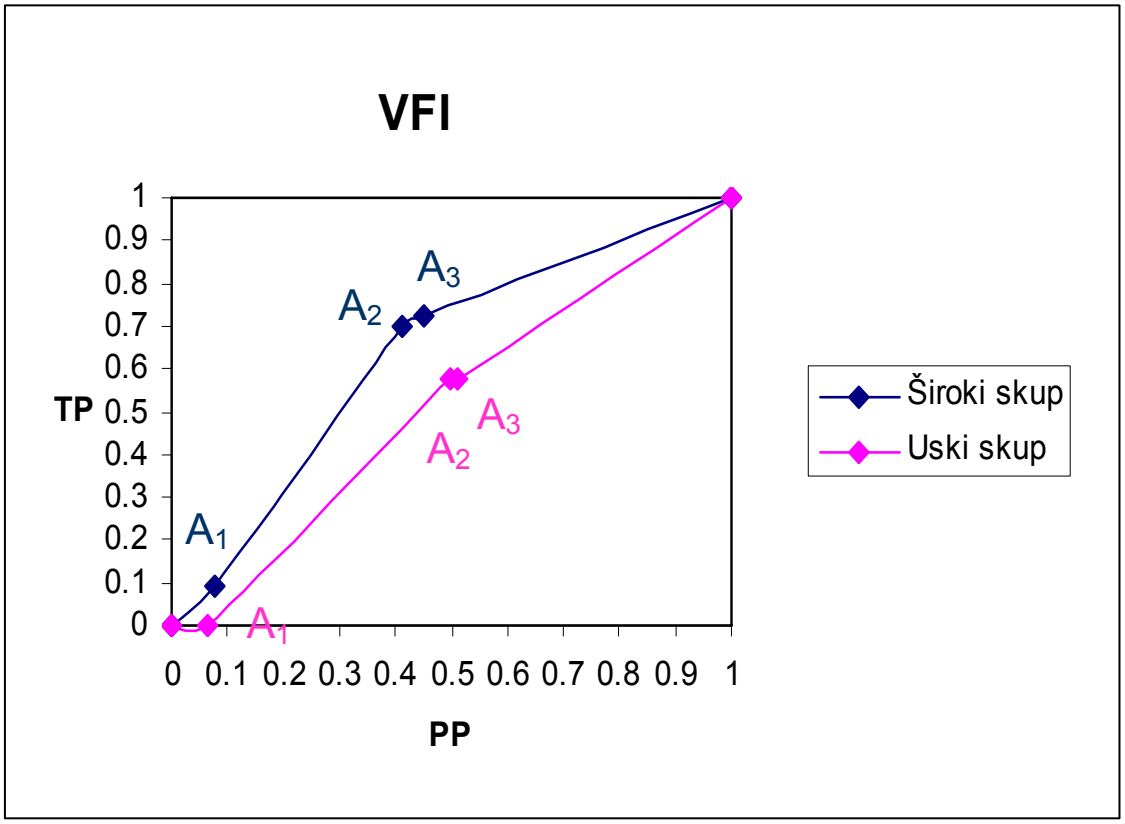
Rezultati su grafički prikazani na slici 7.1. Najbolji su rezultati dobiveni za podskup A_2 gdje je ukupna točnost za široki skup 59%, a za uski skup 51%. Točnost predviđanja (TP) za podskup A_2 je gotovo 70% u slučaju širokog skupa, a 58% za uski skup.

Također je uočljivo da podskupovi A_1 i A_2 imaju manje atributa za uski skup uzoraka u odnosu na široki skup uzoraka, što se može objasniti time da manje atributa djeluje na pozitivne uzorke u slučaju uskog skupa uzoraka, s obzirom da se promatra

isključivo djelovanje gena na kontrolnom sustavu bez djelovanja na cijelom biološkom sustavu.

Tablica 7.1. Rezultati razvrstavanja za metodu VFI

	Široki skup uzoraka			Uski skup uzoraka		
	Broj atributa	TP	PP	Broj atributa	TP	PP
A_1	859	0.093	0.076	698	0	0.064
A_2	54	0.698	0.412	44	0.579	0.497
A_3	50	0.721	0.45	50	0.579	0.511



Slika 7.1. Grafički prikaz rezultata razvrstavanja za metodu VFI

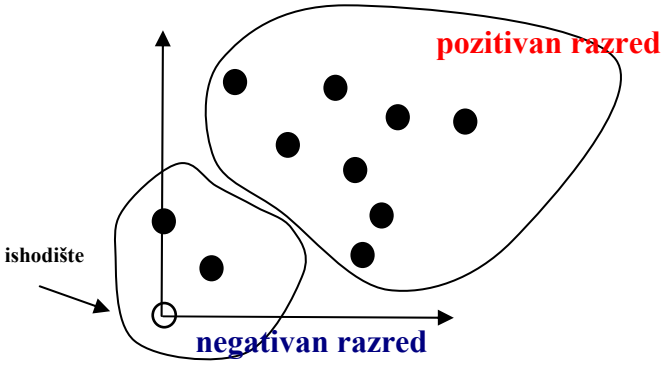
7.2.5. Prilagodbe algoritama razvrstavanja za rad sa skupom uzoraka u kojem izrazito prevladava jedan razred

Osnovni problem koji se javlja kod svih algoritama, osim algoritma VFI, kao što je već spomenuto, je neprilagođenost radu gdje ulazni skup uzoraka čine uzorci koji u velikoj većini pripadaju jednom razredu. Za te algoritme vrijedi da klasifikator, kako bi postigao što veću ukupnu točnost razvrstavanja, sve nove podatke razvrstava u većinski razred, a ono što je doista cilj razvrstavanja – pravilno razvrstati nove uzorke i u manjinski (u ovom slučaju pozitivan) razred, se ne ostvaruje. U konkretnom slučaju, novi uzorci se u svim algoritmima, osim za algoritam VFI, razvrstavaju u većinski, tj. negativan razred, a predviđanje pozitivnog razreda je za te algoritme 0%! Zbog toga je nužno sve algoritme (osim algoritma VFI) prilagoditi radu s takvim specifičnim ulaznim skupom podataka.

Osnovna ideja koja se koristi kod svih prilagodbi je *treniranje klasifikatora samo s manjinskim (pozitivnim) razredom*, nakon čega se bez treniranja negativnim uzorcima proglasi da je neki kriterij ili granica razdvajanja ono što će u fazi razvrstavanja odijeliti pozitivne od negativnih uzoraka. Korištenjem relaksacijskih parametara (eng. *relaxation parameters*) postavlja se i (po potrebi pomiče) granica koja će odijeliti pozitivne i negativne uzorke, a s obzirom da se granica postavlja u fazi treniranja (koja se provodi samo nad pozitivnim uzorcima), ona će u fazi treniranja dio uzoraka koji pripadaju pozitivnom razredu uključiti u negativan razred. Nakon postavljene granice, a ovisno o odabranim relaksacijskim parametrima, manji ili veći dio pozitivnih uzoraka bit će uključen u negativan razred (na slici 7.2. prikazan je princip za algoritam *1-razred* SVM [55]).

7.2.5.1 Prilagodba algoritma SVM za rad s ulaznim podacima koji većinski pripadaju jednom razredu: algoritam *1-razred* SVM

Prilagodba algoritma SVM [59] za rad s ulaznim podacima koji većinski pripadaju jednom razredu *1-razred* SVM (eng. *one-class classification problem*) predložena je u [55]. Osnovna ideja je sljedeća: prvo se klasifikator trenira samo uzorcima koji pripadaju pozitivnom razredu, a jedini uzorak koji je korišten kao predstavnik negativnog razreda je ishodište (vrijednosti svih atributa tog uzorka su 0). Zatim se korištenjem relaksacijskih parametara postavljaju granice razdvajanja pozitivnog i negativnog razreda, a zatim se po potrebi pomiču tako da nakon pomicanja negativan razred obuhvaća i neke uzorke koji pripadaju pozitivnom razredu. Princip je prikazan na slici 7.2. [54]: nakon promjene relaksacijskih parametara, granice negativnog razreda uključuju ishodište i još dvije točke (tj. dva uzorka), a sve ostale točke (tj. uzorci) pripadaju pozitivnom razredu.



Slika 7.2. *1-razred* SVM

7.2.5.2 Prilagodba algoritma *k*-najbližih susjeda za rad s ulaznim podacima koji većinski pripadaju jednom razredu: *1-razred k*-najbližih susjeda

Prilagodba algoritma *k*-najbližih susjeda [60] za rad samo s jednim (pozitivnim razredom) (eng. *NN-PC*, *Nearest Neighbour - Positive Class*) predložena je u [56]. Faza treniranja provodi se samo nad uzorcima koji pripadaju pozitivnom razredu. Pri tome se odredi konstanta *d*, koja će biti maksimalna udaljenost koja može biti između testnog

uzorka i nekog uzorka koji pripada pozitivnom razredu. Ako je testni uzorak udaljen za više od *d* od svih uzoraka koji pripadaju pozitivnom razredu, tada se uzorak razvrstava u negativan razred, a inače se razvrstava u pozitivan razred.

Konstanta *d* se definira kao maksimalna udaljenost između svih udaljenosti d_i, gdje se udaljenost d_i definira kao najmanja udaljenost između uzorka u_i i bilo kojeg drugog uzorka u_j iz skupa uzoraka:

$$d = \text{Max} \{ d_i \mid \text{za } u_i \text{ Min d}(u_i, u_j)\}$$

Mjera udaljenosti može biti Euklidova mjera udaljenosti ili neka druga. U radu je korištena Euklidova mjera udaljenosti.

7.2.5.3 Prilagodba naivnog Bayesovog algoritma za rad s ulaznim podacima koji većinski pripadaju jednom razredu: *1-razred* naivni Bayesov algoritam

Prilagodba naivnog Bayesovog algoritma [60] za rad samo s jednim (pozitivnim razredom) (eng. *NB-PC*, *Naive Bayes - Positive Class*) predložena je u [56]. Faza treniranja provodi se samo nad uzorcima koji pripadaju pozitivnom razredu. Pri tome se u fazi treniranja određuje prag *p*, koji će predstavljati minimalnu vjerojatnost između svih vjerojatnosti p_i određenih za svaki uzorak u_i iz skupa uzoraka. Vjerojatnost p_i je umnožak vjerojatnosti pojedinih atributa uzorka u_i, gdje je vjerojatnost pojedine vrijednosti atributa v_k normalizirana prema najučestalijoj vrijednosti tog atributa:

$$p = \text{Min } \{p_i \mid \prod p\ (A_j = v_k)\}$$

U fazi testiranja pozitivnim se uzorkom smatra onaj uzorak čiji je umnožak vjerojatnosti atributa veći od praga *p*, a inače se uzorak smatra negativnim.

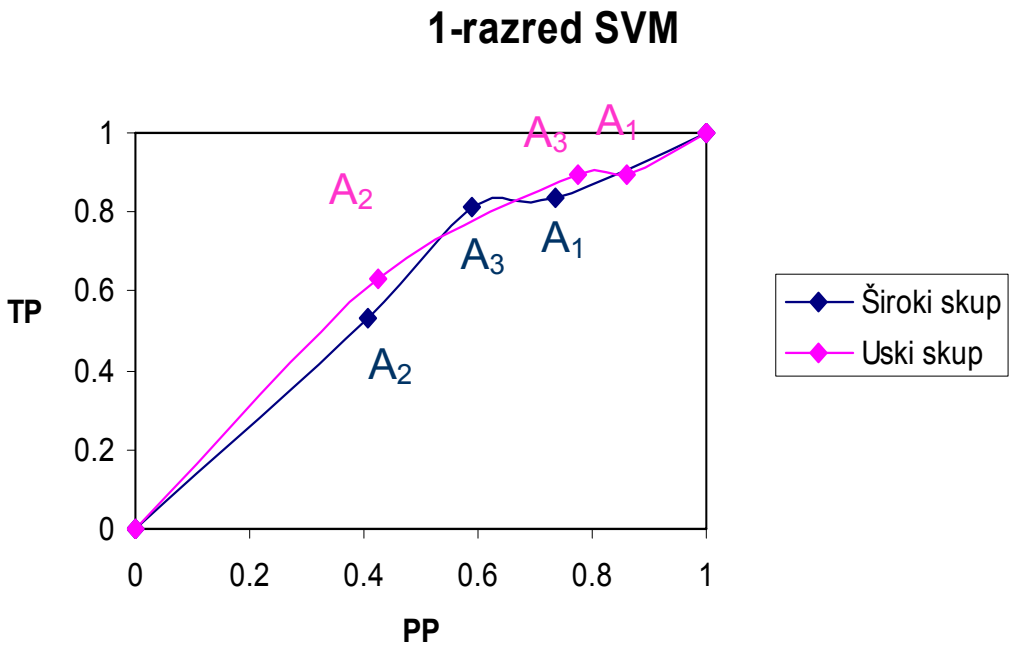
7.2.6. Rezultati razvrstavanja za algoritam *1-razred* SVM

Prikazani rezultati dobiveni su korištenjem programskog paketa WLSVM [58], koji je implementacija LibSVM [57] za Weka-u [39]. LibSVM je implementacija algoritma SMO [50] (linearna implementacija algoritma SVM) koji uključuje podršku za *1-razred* SVM. Kod testiranja su korištene funkcije jezgre (linearna funkcija, RBF funkcija, sigmoidalna funkcija) dostupne za WLSVM [58] s različitim parametrima, a pokazalo se da se najbolji rezultati dobiju za funkciju jezgre RBG (eng. *radial basis function*) i parametar $\gamma = 20$.

Rezultati razvrstavanja za metodu *1-razred* SVM prikazani su u tablici 7.2. U rezultatima je uočljivo, da jednako kao i u slučaju algoritma VFI, manji, dobro odabran podskup od nekoliko desetaka atributa (A_2 ili A_3) daje bolje rezultate nego veći podskup od više stotina atributa (A_1). Najbolji rezultati dobiveni su za podskup A_2 za funkciju jezgre RBF i parametar $\gamma = 20$. Ukupna točnost za široki skup je 59%, a za uski skup je 58%. Točnost predviđanja pozitivnog razreda (TP) za široki skup uzoraka je 53%, a za uski skup uzoraka je 63% (prikazano na slici 7.3.).

Tablica 7.2. Rezultati razvrstavanja za metodu *1-razred* SVM

	Široki skup			Uski skup		
	Broj atributa	**TP**	**PP**	**Broj atributa**	**TP**	**PP**
A_1	859	0.837	0.734	698	0.895	0.862
A_2	54	0.535	0.408	44	0.632	0.424
A_3	50	0.814	0.59	50	0.895	0.776



Slika 7.3. Rezultati razvrstavanja *1-razred* SVM

7.2.7. Rezultati razvrstavanja za algoritam *1-razred* naivni Bayes i algoritam *1-razred k*-najbližih susjeda

Algoritam *1-razred* naivni Bayes [56] se pokazao neprikladnim za korišteni skup uzoraka, te rezultati nisu ni prikazani. Predloženi prag prema kojem bi bili odijeljeni pozitivni i negativni uzorci za korišteni skup uzoraka daje vrlo loše rezultate, tj. sve testne uzorke svrstava u pozitivan razred. Prihvatljivi rezultati dobiju se nakon povećanja praga za nekoliko redova veličina, ali nakon toga već mala povećanja praga daju loš rezultat prema kojem se svi uzorci razvrstavaju u negativan razred. Zbog malog raspona vrijednosti praga koje daju prihvatljive rezultate što se dobije empiričkim putem, ova je metoda odbačena kao neprikladna za testirani skup podataka.

Algoritam *1-razred k*-najbližih susjeda [56] daje dobre rezultate za male modifikacije maksimalne udaljenosti *d*. Umjesto predložene vrijednosti prema [56], dobre rezultate razvrstavanja (sumjerljive vrijednostima koje daje algoritam *1-razred* SVM) dobiju se za vrijednosti *d*/2, *d*/3. Nedostatak ove metode je nemogućnost rada sa skupom atributa većim od nekoliko desetaka, pa metoda nije testirana za skup A_1, nego samo za skupove A_2 ili A_3.

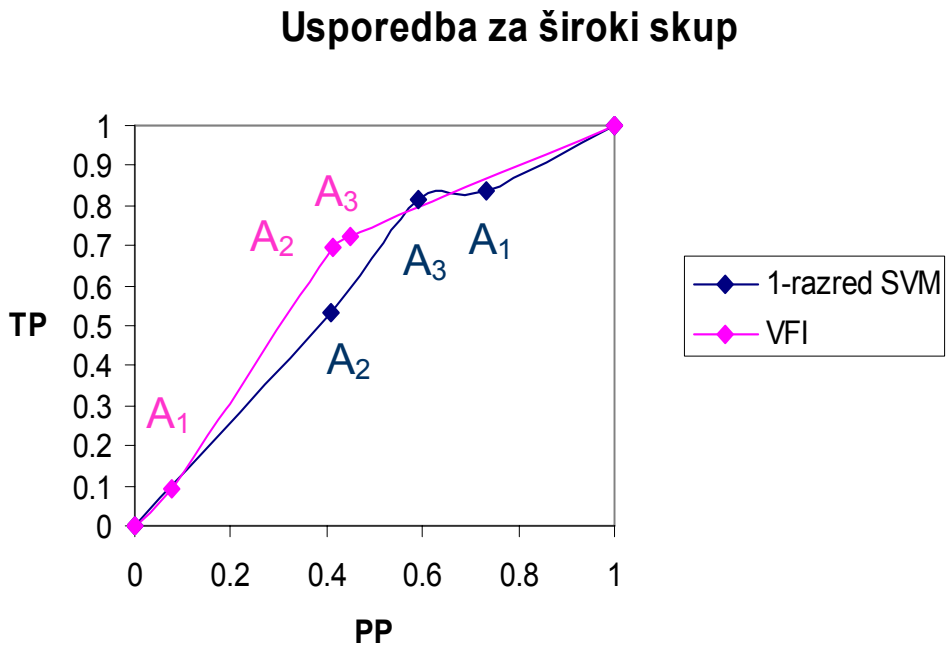
7.2.8. Usporedba rezultata za algoritam *1-razred* SVM i algoritam VFI

S obzirom da su najbolje rezultate dali algoritam VFI i algoritam *1-razred* SVM, ovdje je prikazana usporedba rezultata za široki (slika 7.4.) i uski skup podataka (slika 7.5.).

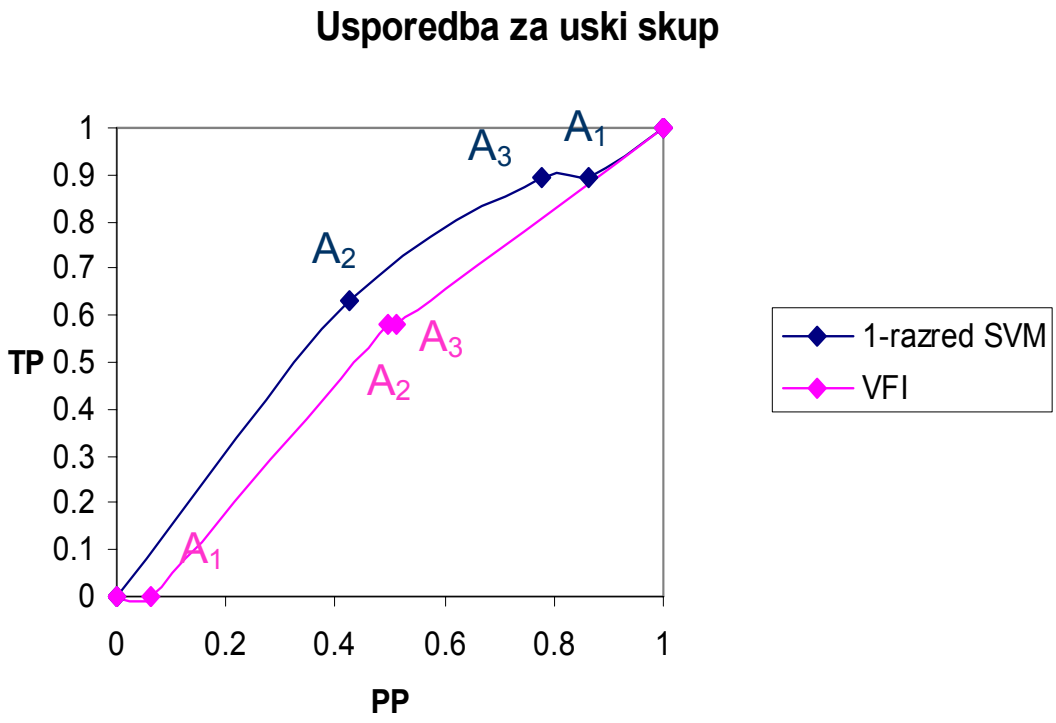
Za oba se algoritma najbolji rezultat dobije za podskup A_2, koji je na grafu najbliži točki (1, 0) (slika 7.4. i slika 7.5.). Točka (1, 0) predstavlja idealan klasifikator (sve pozitivne uzorke razvrstava kao pozitivne, a sve negativne uzorke razvrstava kao negativne, što je detaljnije objašnjeno u 5. poglavlju).

Uočljivo je da za zadani skup podataka algoritam VFI daje bolje rezultate za široki skup podataka (slika 7.4.). S druge strane, algoritam *1-razred* SVM daje bolje rezultate za uski skup podataka (slika 7.5.).

Također, lako je uočiti da se za ova dva algoritma pozicija podskupa A_1 na grafu razlikuje. Za algoritam VFI podskup A_1 je blizu ishodišta, a za algoritam *1-razred* SVM blizu točke (1, 1). To je zato što algoritam VFI u fazi treniranja radi i s pozitivnim i negativnim uzorcima (prevladavaju negativni uzorci), a kako podskup A_1 nije dovoljno dobar podskup da bi dobro odijelio pozitivne i negativne uzorke, većina testnih uzoraka se razvrstava u negativan razred, što je na krivulji točka (0, 0). S druge strane, algoritam *1-razred* SVM u fazi treniranja radi isključivo s pozitivnim uzorcima, pa s obzirom da je riječ o podskupu koji nije dovoljno dobar da odijeli pozitivne i negativne uzorke, većina testnih uzoraka razvrstava se u većinski, tj. pozitivan razred, što na krivulji odgovara točki (1, 1).



Slika 7.4. Usporedba rezultata razvrstavanja za široki skup podataka – algoritam VFI i algoritam *1-razred* SVM



Slika 7.5. Usporedba rezultata razvrstavanja za uski skup podataka - algoritam VFI i algoritam *1-razred* SVM

U konačnoj usporedbi između algoritma VFI i algoritma *1-razred* SVM, ali i ostalih metoda razvrstavanja prilagođenih za treniranje samo uzorcima jednog razreda, za konkretan slučaj ulaznih podataka, prednost je na strani algoritam VFI jer nije ovisan o parametrima, dok su rezultati drugih metoda, pa tako i algoritma *1-razred* SVM uvelike ovisni o parametrima.

Također je uočljivo da rezultati dobiveni za podskupove A_2 i A_3 daju puno bolje rezultate od podskupa A_1. Međutim, dok su za algoritam VFI rezultati dobiveni za podskupove A_2 i A_3 slični, iako malo bolji za A_2, za metodu *1-razred* SVM ti se rezultati značajnije razlikuju u korist podskupa A_2. Time je pokazana opravdanost odabira podskupa atributa u dva koraka kako je opisano u poglavlju 7.2.1. u odnosu na odabir podskupa atributa isključivo prema informacijskoj dobiti.

7.3. Komentar rezultata dobivenih algoritmima razvrstavanja

Obavljenim se testiranjem pokazalo da se smanjivanjem broja atributa s oko 2500 na pedesetak (podskupovi A_2 i A_3) značajno ubrzava proces izgradnje modela razvrstavanja, a što je još važnije, postiže se i puno bolja točnost rezultata razvrstavanja uzoraka. Osim toga, smanjenjem broja atributa uklonjeni su i oni atributi čije su vrijednosti nepoznate u većem broju uzoraka (zbog čega je teško odrediti utječu li doista na promatrani sustav), te međusobne ovisni atributi.

Korišteni su sljedeći algoritmi razvrstavanja (bez prethodne prilagodbe za rad s jednim većinskim razredom): (i) naivni Bayesov algoritam [60], (ii) algoritam J48 [39]

(implementacija algoritma stabla odlučivanja C4.5 [48] u programskom alatu Weka), (iii) algoritam SMO [50] (eng. *Sequential Minimal Optimization*) (linearna varijanta algoritma SVM [59]), (iv) algoritam *k*-najbližih susjeda [60], (v) algoritam VFI [8] (eng. *Voting Feature Interval*). Svi su navedeni algoritmi razvrstavanja, osim algoritma VFI, dali vrlo loše rezultate. Točnije, svi su algoritmi, osim VFI, sve ili gotovo sve testne uzorke razvrstali u negativan razred (neovisno o tome da li se promatrao široki ili uski pozitivni razred). Praktično to znači da je za sve gene predviđeno da pripadaju negativnom razredu, odnosno da ne utječu na promatrani biološki sustav. Točnost predviđanja tih algoritama za pozitivan razred je 0% ili tek malo više od toga, što je gledano sa strane domenskog eksperta, beskorisno, jer je potrebno izgraditi model koji može predvidjeti koji geni *utječu* na promatrani sustav.

Jedini algoritam koji je bez dodatnih prilagodbi dao dobre rezultate je algoritam VFI, čija je točnost predviđanja pozitivnog razreda: 58% za uski skup podataka i 70% za široki skup podataka, što je prihvatljiv rezultat za ovakav skup podataka.

S obzirom na specifičnost ulaznih podataka u kojima je prisutan većinski negativan razred, ostale je algoritme (koji su inicijalno dali loše rezultate) bilo potrebno prilagoditi za rad s takvim podacima. Nakon prilagodbe testirani su algoritmi: (i) *1-razred* SVM [55] (ii) *1-razred* naivni Bayesov algoritam [56] (iii) *1-razred k*-najbližih susjeda [56]. Algoritam *1-razred* SVM je za odabir najprikladnijih parametara (funkcija jezgre RBF i parametar $\gamma = 20$) dao rezultate sumjerljive rezultatima algoritma VFI, ali je nedostatak ovog algoritma velika ovisnost o parametrima. Jednako je zapažanje i za algoritam *1-razred k*-najbližih susjeda, čiji je dodatni nedostatak nemogućnost rada sa skupom atributa većim od nekoliko desetaka. Za *1-razred* naivni Bayesov algoritam također vrijedi da je uvelike ovisan o parametrima, te se za zadani skup ulaznih podataka dobri rezultati dobiju samo za mali raspon empirički podešenih parametara.

8. Zaključak

Metode prikazane u ovom radu odnose se na primjenu dubinske analize podataka nad biološkim podacima, ali se mogu primijeniti i u analizi podataka iz drugih područja, posebno onih u kojima su uzorci karakterizirani vrlo velikim brojem atributa, te većinskom pripadnosti uzoraka jednom razredu. Pokazano je da je temeljita priprema podataka, u konkretnom slučaju s posebnim naglaskom na smanjenje dimenzijske složenosti, iako vremenski najzahtjevniji dio u cjelokupnom procesu otkrivanja znanja, nužna za dubinsku analizu podataka, kako bi se brže i jednostavnije dobili kvalitetni rezultati u samoj analizi.

Uočena je neprikladnost većine klasičnih metoda razvrstavanja za rad s velikim skupovima podataka, te nužnost prilagodbe većina metoda razvrstavanja za rad s uzorcima koji većinski pripadaju jednom razredu. Od testiranih algoritama razvrstavanja, jedino je algoritam VFI dao dobre rezultate bez posebne prilagodbe za rad s uzorcima koji većinski pripadaju jednom razredu. Od ostalih algoritama, koji su morali biti prilagođeni za rad s takvim specifičnim skupom podataka, najbolje je rezultate dao algoritam 1-razred SVM, iako je njegov nedostatak, kao i nedostatak ostalih algoritama koji su morali biti prilagođeni, ovisnost o parametrima. Daljnja bi istraživanja trebalo usmjeriti prema razvoju novih algoritama razvrstavanja za rad s ovakvim specifičnim skupom ili prilagodbi postojećih algoritama razvrstavanja kako bi se smanjila ovisnost o parametrima.

Iskustvo stečeno u ovom radu pokazalo je važnost ostvarivanja suradnje s poznavateljima domenske problematike, čime se olakšava upoznavanje strukture i značenja podataka koje je potrebno analizirati, a ubrzava i usmjerava proces pripreme podataka. Na kraju, rezultati dobiveni analizom dobivaju smisao samo ako imaju značenje krajnjem korisniku (poznavatelju domenske problematike), odnosno ako mu pružaju novu informaciju, te ovisno o problematici, ubrzavaju istraživanja usmjeravajući ih u određenom smjeru. U konkretnom slučaju analize bioloških podataka, usmjeravanje istraživanja donosi znatnu materijalnu i vremensku uštedu u vrlo skupocjenim, opsežnim i dugotrajnim biološkim istraživanjima.

9. Literatura

1. Hand D., *Data Mining: Statistics and More*, The American Statistician, 52(2): 112-118., 1998.
2. Fayyad U. , Chaudhuri S., Bradley P., *Data Mining and its Role in Database Systems*
3. *Data Mining: Dictionary Entry and Meaning*, http://www.hyperdictionary.com, 2003
4. Craven M., *KDD Cup 2002 Training and Test Data* http://www.biostat.wisc.edu/~craven/kddcup/train.html, 2002
5. Han J., Kamber M., *Data Mining Concepts And Techniques*, Morgan Kaufmann Publishers, 2001.
6. De Veaux R., *Conference Tutorial*, *European Conference od Advanced Statistics*, 2003.
7. Goharian N., Grossman D., *Introduction to Data* Mining, 2003.
8. Demiröz G., Güvenir H. A., *Classification by Voting Feature Intervals*, ECML-97, 1997.
9. Hearst M. A., *Trends controversies: Support vector machines*. IEEE Intelligent System, 13(4):18-28, 1998. BibTeX entry
10. *Convex Hull: Dictionary Entry*, http://computing-dictionary.thefreedictionary.com, 2004
11. Weiss S. M., Indurkhya N., *Decision-Rule Solutions for Data Mining with Missing Values*. IBERAMIA-SBIA 2000: 1-10, 2000.
12. Quinlan J. R., *Unknown Attribute Values in Induction,* Proceedings of 6th International Workshop on Machine Learning, 164-168, 1989.
13. Grzymala-Busse J. W., Hu M., *A Comparison of Several Approaches to Missing Attribute Values in Data Mining*. In Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing RSCTC'2000.
14. Liu H., Setiono R.; *Incremental Feature Selection*, Applied Intelligence,Vol. 9, Nr. 3: 217-230, 1998.
15. Lau M., Schultz M., *A Feature Selection Method for Gene Expression Data with Thousands of feature,* http://zoo.cs.yale.edu/classes/cs490/0203b/manfred.lau/cs490%20feature%20selection%20paper.pdf
16. Portinale L., Saitta L., *Feature Selection*. D14.1, IST Project MiningMart, IST-11993, 2002.
17. Molina L. C., Belanche L., Nebot A., *Feature Selection Algorithms: A Survey and Experimental Evaluation*, 2002 IEEE International Conference on Data Mining (ICDM'02): 306**,** 2002
18. Duin R. P. W., Jain A. K., Mao J., *Statistical Pattern Recognition: A review.* IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1):4-37, 2000.
19. Dash M., Liu H., *Feature Selection for Classification*, Intelligent Data Analysis 1: 131–156, 1997.
20. Hall M. A., Smith L., A., *Feature Selection for Machine Learning Comparing Correlation based Filter Approach to the Wrapper*, FLAIRS-99 Program, 1999

21. Narendra P. M., Fukunaga K., *A Branch and Bound Algorithm for Feature Selection.* IEEE Transactions on Computers, C-26(9):917–922, 1977.
22. Koller D., Sahami M., *Toward Optimal Feature Selection.* Proceedings of International Conference on Machine Learning, 1996.
23. Yang J., Honavar V,. *Feature Subset Selection Using a Genetic Algorithm*. Proceedings of the Genetic Programming Conference: 380-385. Stanford, CA, 1997.
24. Schlimmer J. C., *Efficiently Inducing Determinations: A Complete and Systematic Search Algorithm That Uses Optimal Pruning*. *Proceedings of Tenth International Conference on Machine Learnin*g, 284–290, 1993.
25. Kira K., Rendell L. A., *The Feature Selection Problem: Traditional Methods and a New Algorithm*. Proceedings of Ninth National Conference on Artificial Intelligence: 129–134, 1992.
26. Liu H., Setiono R., *A Probabilistic Approach to Feature Selection - a Filter Solution*. Proceedings of International Conference on Machine Learning: 319–327, 1996.
27. Liu H., Setiono R., *Feature Selection and Classification - a Probabilistic Wrapper Approach.* Proceedings of Ninth International Conference on Industrial and Engineering Applications of AI and ES: 284–292, 1996.
28. Xing E., Jordan M., Karp R., *Feature Selection for High-dimensional Genomic Microarray Data*. International Conference on Machine Learning, 2001.
29. Kohavi R., John G. H.; *Wrappers for Feature Subset Selection*, Artificial Intelligence 97(12): 273—324, 1997.
30. *Saccharomyces Genome Deletion Project*, http:// www.sequence.stanford.edu/group/yeast_deletion_project
31. baza podataka MEDLINE, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi
32. baza podataka *Saccharomyces Genome Database*, http://genome-www.stanford.edu/Saccharomyces/
33. baza podataka *MIPS Comprehensive Yeast Genome Database,* http://mips.gsf.de/proj/yeast/CYGD/db/index.html
34. Craven M., *KDD cup 2002, Task 2: Yeast Gene Regulation Prediction* http://www.biostat.wisc.edu/~craven/kddcup/tasks.html, 2002
35. Kira K., Rendell L., *A Practical Approach to Feature Selection.* Proceedings of the Ninth International Conference on Machine Learning: 249-256. Aberdeen, Scotland: Morgan Kaufmann, 1992.
36. Kohavi, R. Provost, F. Glossary of Terms. *Machine Learnin*g, 30(2/3): 271–274., 1998.
37. Schoonjans F., *ROC Curve Analysis: Introduction*, http://www.medcalc.be/manual/mpage0613a.php, 2004
38. Hamilton H. J., *ROC graph*, http://www2.cs.uregina.ca/~hamilton/courses/831/notes/ROC/ROC.html, 2002
39. Witten I. H., Frank E., *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
40. Paynter G., Trigg L., Eibe F., Kirkby R., *Attribute-Relation File Format* (*ARFF*) http://www.cs.waikato.ac.nz/~ml/weka/arff.html, 2002
41. Kowalczyk A., Raskutti B., *Single Class SVM for Yeast Gene Regulation Prediction*, 2002.

42. Wu S., Flach P., *Data Mining on Thrombin Dataset*, 2001.
43. Jones J., *Vocabulary Workshop: 1000 Most Common Words in English*, http://esl.about.com/library/vocabulary/bl1000_list1.htm, 2004
44. BNC Lema List, *The most useful 2000 words in English*, http://www1.harenet.ne.jp/~waring/vocab/wordlists/lemma.html
45. Craven M., *KDD Cup 2002 Task 2 Evaluation*, http://www.biostat.wisc.edu/~craven/kddcup/task2-scoring, 2002
46. Luz S., *Evaluation Metrics and Comparisons,* 4ICT2: Information Management http://www.cs.tcd.ie/courses/baict/baim/ss/part2/eval-4up.pdf, 2004
47. Kononenko I., *Estimating Attributes: Analysis and Extensions of RELIEF*, European Conference on Machine Learning:171–182, 1994.
48. Quinlan J., *C4.5: Programs for Machine Learning*, San Mateo, Morgan Kaufmann, 1992.
49. Parthasarathy S., *Clustering*, CIS 694Z: Introduction to Data Mining, http://www.cis.ohio-state.edu/~srini/694Z/clst.ppt, 2004
50. John C. P., *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*,Technical Report MSR-TR-98-14, 1998.
51. Han J., *From Data Mining to Web Mining: An Overview*, Conference Tutorial, 2000 Int. Database Systems Conf. (IDS'2000), Hong Kong, June 2000.
52. Liu H., Motoda H., *Feature Selection for Knowledge Discovery and Data Minin*g. Kluwer Academic Publishers, London, GB, 1998.
53. Gavade A., *Aspects Of Feature Selection for KDD,* Advanced Topics in Artificial Intelligence, 2001.
54. Manevitz L. M., Yousef M., *One-Class SVMs for Document Classification*, Journal of Machine Learning Research 2 139-154, 2001.
55. Schőlkopf B., Platt J. C., Shawe-Taylor J., Smola A.J., Williamson R.C., *Estimating the Support of a High-dimensional Distribution*, Technical Report, Microsoft Research, MSR-TR-99-87,1999.
56. Datta P., *Characteristic Concept Representations*, PhD thesis, University of California, Irvine, 1997.
57. Fan R.-E., Chen P.-H., Lin C.-J., *Working Set Selection Using the Second Order Information for Training SVM*. Journal of Machine Learning Research 6, 1889-1918, 2005.
58. EL-Manzalawy Y., Honavar V., *WLSVM: Integrating LibSVM into Weka Environment*, http://www.cs.iastate.edu/~yasser/wlsvm, 2005.
59. Vapnik V., *Statistical Learning Theory*, Wiley-Interscience, New York, 1998.
60. Duda R., Hart P., *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

Dodatak A: Sažetak

Biološki procesi opisani su mnoštvom podataka prikupljenih promatranjima i eksperimentima. Kako bi se iz složene strukture bioloških podataka izlučile nove informacije koriste se metode dubinske analize podataka. S obzirom na složenost strukture podataka, prije same dubinske analize podatke je potrebno adekvatno pripremiti. Priprema podataka uključuje pročišćavanje podataka, integraciju i transformaciju podataka. U ovom je radu poseban naglasak stavljen na transformaciji podataka, posebno na metodama smanjivanja dimenzijske složenosti podataka.

Podaci korišteni u praktičnom dijelu rada odnose se na gene za koje je potrebno predvidjeti da li utječu na promatrani biološki sustav ili ne. S obzirom na problematiku, proučeni su postupci dubinske analize podataka s naglaskom na metode razvrstavanja (klasifikacije) podataka. Uspoređeni su odabrani algoritmi razvrstavanja, te je objašnjeno što je utjecalo na bolje, odnosno lošije rezultate odabranih algoritama. Uočeno je da najbolje rezultate daju algoritam VFI i algoritam 1-razred SVM.

Ključne riječi:
dubinska analiza podataka, biološki podaci, smanjivanje dimenzijske složenosti podataka, algoritam razvrstavanja

Dodatak B: Summary

Title: Comparison of Data Mining Techniques Applied on Biological Data

Biological processes are described using huge amount of data collected through the observation and experiments. Data mining is used for interesting knowledge extraction. Due to data complexity, data mining process should succeed the process of cleaning data, integration of data from different sources and transformation of data. Considering the problems involved with the data set used in this work, the emphasis is set on the transformation of data, especially on the feature selection.

Considering the given data set, the goal of the work is to learn a model that can predict whether the particular gene influences the specified biological system. The overview of data mining techniques is given with the special attention to classification algorithms, because the model for the given data set is build using classification algorithms. The comparison of classification algorithms showed that the best results are provided using VFI (*Voting Feature Interval*) algorithm and one-class SVM (*Support Vector Machines*) algorithm.

Keywords:
data mining, biological data, feature selection, data classification methods