SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1211

PREDVIĐANJE MJESTA SEKUNDARNE STRUKTURE PROTEINA IZ SLIJEDA AMINOKISELINSKIH OSTATAKA

Saša Janjić

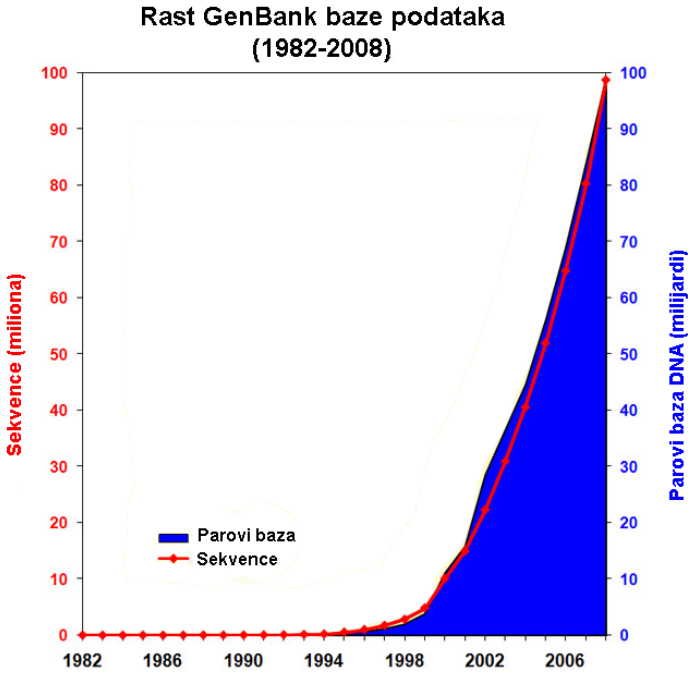
Zagreb, Ožujak 2010.

Hvala roditeljima na bezuvjetnoj podršci i strpljenju tijekom studija,
te Mile Šikiću na pomoći pri izradi rada!

SADRŽAJ

1 UVOD

Proteini imaju ključnu ulogu u gotovo svim biološkim procesima. Uloga proteina definirana je njegovom funkcijom koju s druge strane određuje njegova struktura. Proučavanje svojstava, interakcija i funkcija proteina zadatak je proteomike (1), znanstvene discipline čiji je cilj opisati ukupnost proteina koji tvore organizme (proteome). Proteini su kompleksne organske strukture koje se sastoje od aminokiselina povezanih peptidnim vezama, a čiji je slijed određen genima koji ih kodiraju (2). Istraživanje genoma urodilo je spoznavanjem ogromnog broja aminokiselinskih sljedova koje kodiraju geni (Slika 1-1), međutim funkcija, struktura i interakcije proteina pripadnih sljedova uglavnom su nepoznate.



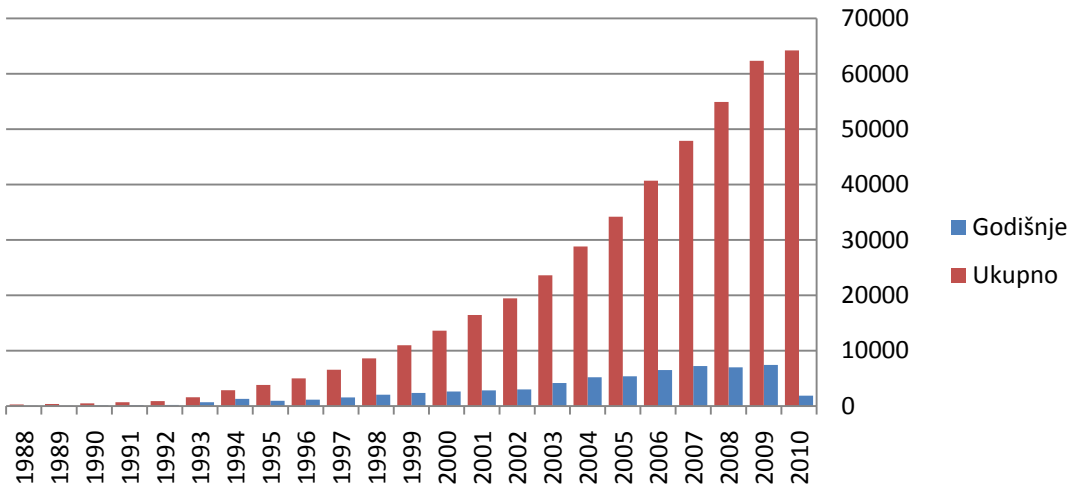
Slika 1-1 Broj sljedova nukleotidnih baza i primarnih sekvenci

Linearan niz aminokiselina koje tvore protein uvija se u specifičnu trodimenzionalnu strukturu koja određuje njegovu funkciju. Zbog toga se ulaže golem napor da se te strukture odrede, bilo eksperimentalno, bilo računski. Zadaća je biotehnologa da eksperimentalno odrede strukture proteina metodama rendgenske kristalografije i nuklearne magnetske rezonance. To su skupe, sofisticirane i vremenski zahtjevne metode pa je sve veća, i sve brže raste, razlika između broja poznatih sljedova i pripadnih struktura (Slika 1-2). Stoga je nužno stvoriti, unaprijediti i iskoristiti računalne metode koje će na temelju poznatih aminokiselinskih nizova vjerodostojno predvidjeti konačnu strukturu proteina.

Predviđanje sekundarne strukture proteina predstavlja bitan međukorak u takvom nastojanju određivanja 3D strukture proteina. To je zapravo predviđanje uobičajenih lokalnih struktura poput α-uzvojnica i β-niti unutar zadanog proteinskog slijeda (primarne strukture). Uz poznatu sekundarnu strukturu, a na temelju znanja o ponašanju i oblikovanju takvih struktura moguće je dobiti relativno malen broj mogućih 3D struktura. Sekundarne

strukture su tijekom evolucije puno očuvanije u odnosu na slijed aminokiselina; stoga bi točno predviđanje elemenata sekundarne strukture značilo velik doprinos strukturnoj bioinformatici. Poznavanje elemenata sekundarne strukture može primjerice pomoći pri provedbi poravnanja sljedova ili poboljšati postojeće poravnanje udaljeno povezanih sljedova male sličnosti. Predviđanje sekundarne strukture dobra je polaznica u utvrđivanju konačne trodimenzionalne strukture, ono služi kao međukorak u postupcima prepoznavanja smatanja (engl. *folds*), tj. identifikaciji predložaka za modeliranje po sličnosti i može pružiti korisna ograničenja kako u modeliranju po sličnosti (engl. *comparative / homology modeling*) tako i u *de novo* modeliranju.

Problem predviđanja sekundarne strukture svodi se na sljedeće: uz dan proteinski slijed $a_1a_2a_3...a_n$, odrediti koju sekundarnu strukturu poprima svaki od aminokiselinskih ostataka a_i. Često se podjela ciljnih struktura svodi na: -uzvojnicu (*H*), β-ploču (*E*) i ostale strukture (*C*).



Slika 1-2. Broj riješenih struktura proteina (Protein data bank)

Cilj ovog rada je na temelju triju skupova podataka ocijeniti i usporediti uspješnost dvaju ansambl sustava klasifikatora u predviđanju sekundarne strukture proteinskih ostataka (rezidua). Klasifikacija se vrši metodom slučajne šume te AdaBoost metodom uz pomoć Rattle programa. Pri tome su podaci o proteinskim lancima pojedinih skupova dohvaćeni iz PDB baze podataka, nakon čega su DSSP programskim alatom određene njihove pripadne sekundarne strukture. Navedeni podaci konačno su objedinjeni pročišćavanjem uz pomoć vlastito izrađenih Perl skripti, zajedno sa informacijama o evolucijskoj očuvanosti aminokiselinskih ostataka u obliku profila slijeda dobivenih PSI-BLAST programom. Ciljni atribut sekundarne strukture raspoređuje se u tri klase: *H* (Helix), *C* (Coil) te *E* (Extended) te se analizira točnost tako provedene klasifikacije.

2 ANSAMBL SUSTAVI KLASIFIKATORA

Razvrstavanje ili klasifikacija je postupak pronalaska modela koji opisuje princip podjele podataka po razredima da bi se na temelju tog modela predvidjela pripadnost podataka za koje je razred odnosno klasa nepoznata. Izgradnja i testiranje modela klasifikacije zahtjeva podjelu dostupnih zapažanja poznatog razreda u dva skupa:

1. pokazni ili skup uzoraka za treniranje (engl. *training set*) - skup podataka koji se koristi za izgradnju modela

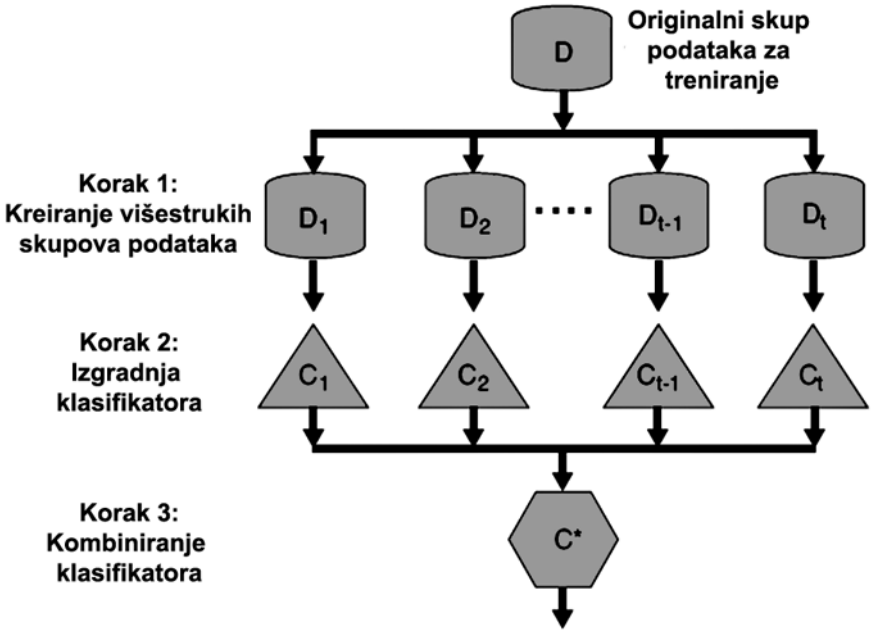
2. testni skup uzoraka (engl. *test set*) - skup podataka nad kojim se provjerava ispravnost modela.

Nedostatak ovakve podjele je u tome što se podaci testnog skupa ne koriste za izgradnju modela. Nužno je međutim izdvojiti dio podataka u testni skup jer on služi za ispitivanje točnost izgrađenog modela. Točnost modela je postotak testnih uzoraka koji su točno klasificirani izgrađenim modelom. Izdvajanje testnog skupa nužno je jer bi u suprotnom određivanje točnosti modela nad uzorcima pokaznog skupa rezultiralo pretreniranošću (engl. *overfitting*) modela, tj. moglo bi se desiti da model „zapamti“ neke anomalije svojstvene samo pokaznom skupu podataka.

Ako se pokaže da je točnost razvrstavanja uzoraka iz testnog skupa podataka korištenjem izgrađenog modela zadovoljavajuća, onda se dobiveni model može koristiti i za razvrstavanje podataka nepoznatog razreda (klase).

Metode klasifikacije pripadaju općenitijoj skupini algoritama za nadgledano učenje (engl. *supervised learning*). To su postupci u kojima se za izgradnju modela koriste podaci unaprijed poznatih pripadnih razreda, te kojima se zatim na temelju izgrađenog modela predviđa razred pripadnosti novih, nepoznatih podataka. Cilj klasifikacije je donošenje odluke odabirom jedne vrijednosti iz prethodno definiranog skupa izbora. Ekspert (klasifikator) stvara hipotezu o klasifikaciji primjera podatka u jednu od predefiniranih kategorija koje predstavljaju različite odluke. Odluka se temelji na prethodnom treniranju klasifikatora korištenjem reprezentativnog skupa podataka za koji su pripadne odluke a priori poznate.

Strojno učenje proučava automatske tehnike učenja kojih je cilj na temelju prošlih opažanja ostvariti točna predviđanja. Otkrivanje jedinstvenog pravila koje dovodi do točnog predviđanja je kompleksan i teško ostvariv zadatak. S druge strane, puno je lakše na temelju opažanja izgraditi skup jednostavnih i umjereno točnih pravila. Ansambl sustavi su paradigma strojnog učenja koja se temelji na treniranju višestrukih baznih sustava učenja (engl. *base learner*) za rješavanje istog problema (Slika 2-1). Klasični pristupi strojnog učenja nastoje naći jednu najbolju hipotezu $\mathcal{H}$ kao rješenje danog problema učenja. Za razliku od takvih postupaka koji traže jedinstvenu hipotezu za tumačenje podataka, ansambl algoritmi učenja stvaraju skup hipoteza koje se zajedno koriste za rješavanje problema. Nekoliko je teoretski i praktično opravdanih razloga za korištenje ansambl sustava klasifikatora (3).



Slika 2-1. Ansambl sustav klasifikatora

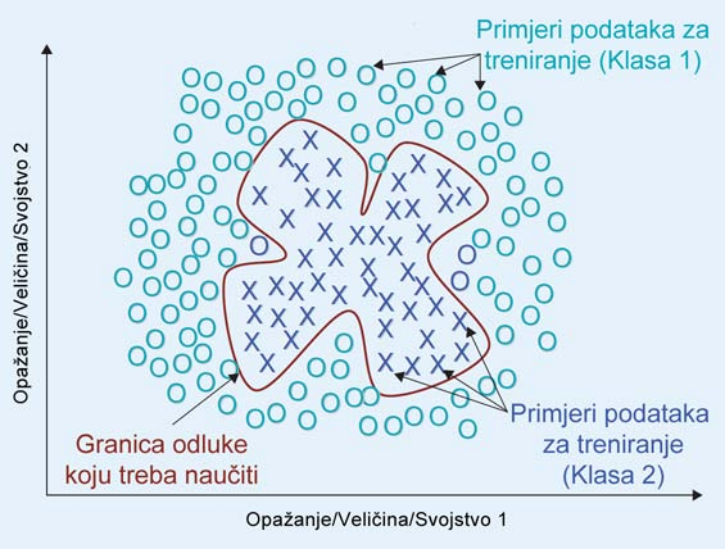
Statistički razlozi. Poznato je da učinkovitosti raznih automatiziranih klasifikatora na skupu za treniranje ne znači nužno i njihovu dobru sposobnost generalizacije, odnosno učinkovitost nad podacima koji nisu bili dostupni tijekom treniranja. Skup klasifikatora slične učinkovitosti pri treniranju može imati različitu učinkovitost generalizacije. Čak i klasifikatori sličnih sposobnosti generalizacije mogu ostvariti različitu učinkovitost, naročito ako podaci korišteni za provjeru nisu dovoljno reprezentativi za neke buduće slučajeve. Kombiniranje izlaza nekoliko klasifikatora u takvim slučajevima može smanjiti rizik odabira neučinkovitog klasifikatora. Uprosjećivanje takvih izlaza može (ali i ne mora) narušiti učinkovitost najboljeg klasifikatora u ansamblu, ali svakako smanjuje ukupni rizik.

Velik obujam podataka. U određenim primjenama količina podataka koju treba analizirati prevelika je da bi se učinkovito obradila samo jednim klasifikatorom. Podjela skupa podataka u manje podskupove, treniranje klasifikatora različitim participijama podataka, te kombiniranje njihovih izlaza korištenjem nekog smislenog pravila daleko je učinkovitiji pristup.

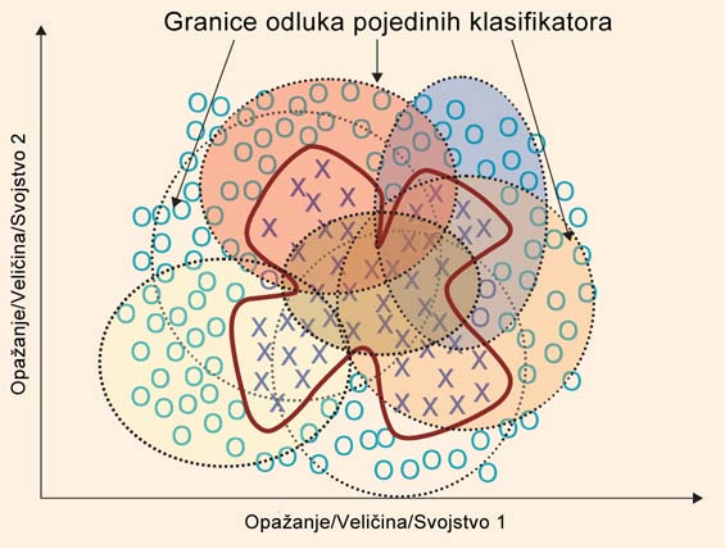
Premalo podataka. Ansambl sustavi također se mogu koristiti i u suprotnome, kada je količina dostupnih podataka premala. Da bi algoritam klasifikacije uspješno naučio inherentnu distribuciju podataka od primarne je važnosti postojanje prikladnog i reprezentativnog skupa podataka za treniranje. U slučaju manjkavog skupa podataka za treniranje mogu se primijeniti tehnike ponovnog uzorkovanja podataka (engl. *resampling techniques*) u kombinaciji sa ansambl sustavom tako da se slučajno odabrani podskupovi izvornog skupa podataka iskoriste za treniranje pojedinih klasifikatora. Takvi pristupi pokazali su se vrlo učinkovitima.

„Podijeli pa vladaj“. Bez obzira na obim dostupnih podataka neki problemi su preteški da bi ih dani klasifikator mogao riješiti. Konkretnije, granica odluke koja odvaja podatke

različitih klasa može biti presložena, ili može ležati izvan prostora funkcija koju je moguće implementirati odabranim modelom klasifikacije. Može se na primjer razmotriti dvodimenzionalan slučaj podjele u dvije klase sa kompleksnom granicom odluke (Slika 2-2). Linearan klasifikator sposoban učiti linearnu podjelu ne može naučiti tako kompleksnu nelinearnu granicu. Primjerena kombinacija ili ansambl takvih linearnih klasifikatora s druge strane može naučiti prikazano nelinearno razgraničenje. Podjelom prostora podataka u manje dijelove koje je lakše naučiti i pri čemu svaki klasifikator uči („specijalizira se“) samo jednu od jednostavnijih podjela, klasifikacijski sustav u određenom smislu slijedi princip „podijeli pa vladaj“. Dotična, kompleksna, granica odluke može se potom aproksimirati prikladnom kombinacijom različitih klasifikatora (Slika 2-3).



Slika 2-2. Granica odluke



Slika 2-3. Kombiniranje klasifikatora za dobivanje granice odluke

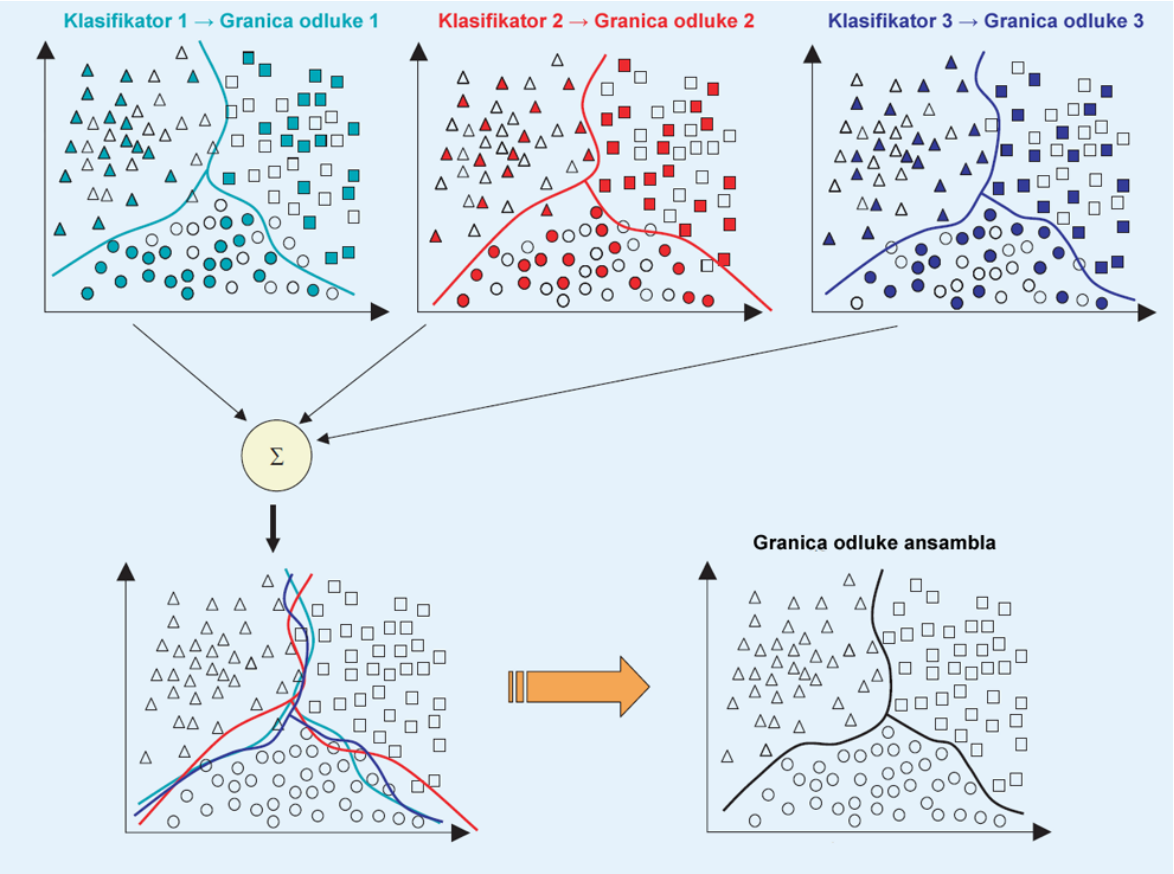
Fuzija podataka. Ukoliko imamo više skupova podataka dobivenih iz različitih izvora, pri čemu je i priroda svojstava različita (heterogena svojstva), jedan klasifikator nije sposoban

naučiti informaciju sadržanu u svim podacima. Npr. pri dijagnozi neuroloških poremećaja, neurolog mora napraviti više testova (MRI, EEG, krvne pretrage, …) da bi uspješno donio dijagnozu. Svaki test rezultira podacima sa različitim brojem i tipom svojstava koji se ne mogu koristiti zajedno za treniranje klasifikatora. U tom slučaju se rezultati pojedine vrste testa mogu koristiti za treniranje različitih klasifikatora čiji se izlazi zatim sjedinjuju. Primjene u kojima se kombiniraju podaci iz različitih izvora da bi se dobila informativnija odluka naziva se fuzija podataka (engl. *data fusion*) te se ansambl pristupi uspješno koriste za takve primjene.

Općenito postoje dva načina kombiniranja klasifikatora: *selekcija* te stapanje ili *fuzija*. Kada se klasifikator trenira na nekom lokalnom dijelu ukupnog prostora svojstava riječ je o selekciji klasifikatora. Tada kombinacija klasifikatora ovisi o pripadnom vektoru svojstava: klasifikator treniran podacima koji se u smislu neke metrike udaljenosti nalaze najbliže okolini vektora svojstava dobiva najveću važnost. Može postojati jedan ili više lokalnih eksperata koji donose odluku. Fuzija podrazumijeva treniranje svih klasifikatora nad cjelokupnim prostorom svojstava. U tom slučaju kombiniranje podrazumijeva stapanje individualnih (slabijih) klasifikatora u jednog (jačeg) eksperta superiornih performansi. Primjer takvog pristupa uključuju *bagging* i *boosting* tehnike kao i njihove mnoge varijacije.

2.1 Raznolikost: Temelj ansambl sustava

Klasifikator savršene sposobnosti generalizacije uklonio bi potrebu za korištenjem ansambl tehnika. Međutim, zbog postojanja šuma, *outlier*-a i preklapajućih distribucija podataka takav klasifikator nemoguće je ostvariti. Strategija ansambl sustava je stvaranje većeg broja klasifikatora te povezivanja njihovih izlaza tako da je ukupna učinkovitost bolja u odnosu na pojedine klasifikatore. To međutim zahtjeva da pojedini klasifikatori griješe na različitim instancama podataka. Ako svaki klasifikator ima drukčiju pogrešku, intuitivno je jasno da njihovo strateško kombiniranje može smanjiti ukupnu pogrešku. Stoga je izgradnja što raznolikijih klasifikatora, osobito s obzirom na pogrešno klasificirane primjere, osnovni princip ansambl sustava. Drukčije rečeno, potrebno je ostvariti klasifikatore čije se granice odluke dovoljno razlikuju od granica odluka ostalih klasifikatora u ansamblu. Raznolikost klasifikatora može se postići na više načina. Najčešće korištena metoda je korištenje različitih skupova podataka za treniranje. Takvi se skupovi obično dobivaju tehnikama ponovnog uzorkovanja (engl. resampling techniques) kao što su *bagging* i *boosting*, gdje se podskupovi podataka za treniranje biraju nasumce, i obično sa zamjenom. Takav pristup prikazan je na slici (Slika 2-4.); tri klasifikatora trenirana različitim, slučajno odabranim i preklapajućim podskupovima podataka za treniranje proizvode različite granice odluke. Te granice se kombiniraju s ciljem dobivanja točnije klasifikacije.



Slika 2-4. Fuzija granica odluka pojedinih klasifikatora

Da bi se unatoč korištenju prilično sličnih podataka za treniranje osigurala neovisnost pojedinih granica, kao bazni modeli koriste se nestabilni klasifikatori (engl. *unstable classifiers*) jer su takvi klasifikatori sposobni proizvesti dovoljno različite granice odluke čak i za male perturbacije parametara za treniranje. Ukoliko se podskupovi podataka za treniranje sastavljaju bez zamjena, postupak se zove *jackknife* ili k-struka (engl. *k-fold*) podjela: čitav se skup dijeli u k dijelova te se svaki od klasifikatora trenira na samo njih *k-1*. Za svaki od klasifikatora bira se *k* različitih podskupova. Raznolikost se može postići i na drugi način, korištenjem različitih parametara za pojedine klasifikatore. Podešavanje parametara omogućuje kontrolu nestabilnosti klasifikatora što doprinosi njihovoj raznolikosti. Mogućnost kontrole nestabilnosti neuronskih mreža i stabala odluke čini te klasifikatore prikladnim kandidatima za korištenje u ansambl sustavima. Za postizanje još veće raznolikosti alternativno je moguće kombinirati različite tipove klasifikatora poput višeslojnih perceptrona, stabala odluke, klasifikatora najbližeg susjeda, te SVM klasifikatora. Kombiniranje različitih modela ili čak različitih arhitektura istog modela, koristi se međutim samo u određenim primjenama koje ih zahtijevaju. Raznolikost se uobičajeno postiže ponovnim uzorkovanjem podataka jer taj postupak ima čvršću teoretsku podlogu. Konačno, raznolikost se može postići i korištenjem različitih svojstava. Generiranje različitih klasifikatora korištenjem slučajnih podskupova svojstava je zapravo prilično rasprostranjeno u određenim primjenama i poznato je pod nazivom metoda slučajnog podprostora (engl. *random subspace method*).

Svi ansambl sustavi sastoje se od dvije ključne komponente. Prvo je strategija potrebna za izgradnju što raznolikijeg ansambla, a drugo je strategija kombiniranja izlaza pojedinih elemenata ansambla tako da se ispravne odluke pojačavaju, a pogrešne anuliraju. Pri stvaranju ansambla također je bitno odgovoriti na dva pitanja: *a)* kako proizvesti pojedine (bazne) klasifikatore? i *b)* po čemu će se oni međusobno razlikovati? Svaka strategija izgradnje pojedinih elemenata treba nastojati unaprijediti raznolikost ansambla. Općenito, međutim, ansambl algoritmi ne nastoje maksimizirati određenu mjeru raznolikosti već se veća raznolikost postiže raznim, i donekle heurističkim, postupcima ponovnog uzorkovanja ili biranja različitih parametara treniranja.

2.2 Bagging

Bagging, skraćenica od *bootstrap aggregating*, jedan je od prvih ansambl algoritama, a ujedno i jedan od najintuitivnijih i najjednostavnijih za implementaciju. Raznolikost kod *bagging* metode postiže se korištenjem *bootstrap* kopija podataka za treniranje: različiti podskupovi podataka za treniranje biraju se nasumce, sa zamjenama, iz ukupnog skupa podataka za treniranje (4). Svaki od podskupova koristi se za treniranje drugog klasifikatora iste vrste. Pojedini klasifikatori ujedinjuju se temeljem njihovih odluka glasovanjem većine. Za bilo koju danu instancu, odluka klasifikatora je ona klasa koju je odabrala većina klasifikatora. *Bagging* je osobito koristan kada su dostupni podaci ograničene veličine. Broj odabranih primjera u podskupu obično čini relativno velik dio izvornih uzoraka (75% do 100%) kako bi se osigurala dovoljna količina primjera u svakom podskupu. To uzrokuje značajno preklapanje pojedinih podskupova za treniranje, pri čemu su neke instance prisutne u većini podskupova dok se druge javljaju i više puta u istom podskupu. Da bi se u takvim uvjetima osigurala raznolikost i dobile različite granice odluka pojedinih klasifikatora a na temelju malih perturbacija skupova za treniranje, koristi se relativno nestabilan model (5).

2.3 Boosting

1990. godine Schapire (6) je dokazao da se algoritam koji proizvodi klasifikatore, *weak learner*, i čija je sposobnost predviđanja tek nešto bolja od nasumičnog pogađanja, može unaprijediti u tzv. s*trong learner*, algoritam za generiranje mnogo moćnijeg klasifikatora. Kao i *bagging*, *boosting* algoritam stvara ansambl klasifikatora ponovnim uzorkovanjem podataka koji se zatim ujedinjuju glasanjem većine. Tu međutim završava sva sličnost dvaju algoritama. Kod *boosting* tehnike ponovno uzorkovanje je strateški osmišljeno da bi se svakom od klasifikatora pružio čim informativniji skup podataka za treniranje. Greška takvog višečlanog klasifikatora ograničena je odozgo i manja je od greške najboljeg klasifikatora u ansamblu, uz uvjet da je stupanj pogreške svakog od klasifikatora manji od ½.

Boosting je opća metoda poboljšanja točnosti algoritma učenja. Za primjenu *boosting* principa potrebno je prije svega odrediti metode ili algoritme za iznalaženje jednostavnih pravila (4). *Boosting* algoritam uzastopno poziva takve „slabe“ ili „bazne“ algoritme učenja te ih u svakoj iteraciji opslužuje drukčijim podskupom primjera za učenje (odnosno

drukčijom distribucijom težina nad primjerima za učenje). Bazni algoritam učenja pri svakom pozivu generira novo slabo pravilo predviđanja koje *boosting* algoritam nakon određenog broja iteracija ujedinjuje u jedinstveno pravilo predviđanja koje je po mogućnosti daleko točnije od bilo kojeg baznog pravila zasebno.

Pri takvom pristupu javljaju se dva glavna pitanja: prvo, kako odabrati distribuciju unutar pojedine iteracije, i drugo, kako slaba pravila ujediniti u jedinstveno? Određivanje distribucije oslanja se na tehniku koja veće težine postavlja na one primjere koji najlošije podliježu prethodnim slabim pravilima odnosno koji su na temelju takvih pravila najčešće krivo klasificirani. Time je bazni algoritam prisiljen da se usredotoči na „teže" primjere. Što se tiče objedinjenja jednostavnih pravila, ono se prirodno i učinkovito vrši po principu većine a na temelju predviđanja tih jednostavnih pravila.

Boosting ideja nastala je osamdesetih godina prošlog stoljeća tijekom teoretskih razmatranja pri razvoju PAC modela učenja. Tada su Kearns i Valiant došli na ideju o združivanja „slabih" algoritama učenja u jedan jači i točniji algoritam. Prvi boosting algoritam sa polinomijalnim vremenom dokazan je 1989. godine, a samo godinu kasnije razvijena je i mnogo učinkovitija inačica koja je unatoč napretku i dalje bila ograničena u praktičnoj primjeni. 1995. Freund i Schapire predstavili su AdaBoost algoritam kojim su riješena praktična ograničenja prijašnjih *boosting* algoritama (7).

2.4 Adaboost

Adaboost je općenitija inačica izvornog *boosting* algoritma (8). Od 1997. kada su Freund i Schapire predstavili osnovni *AdaBoost* algoritam za binarnu klasifikaciju, pa do danas, razvijene su mnoge varijacije za rješavanje višeklasnih (*AdaBoost.M1*) i regresijskih (*AdaBoost.R*) problema. U nastavku je dan prikaz osnovne verzije algoritma koja vrši razvrstavanje u 2 razreda.

Neka je dan skup od *N* primjera, $S = [(\boldsymbol{x_i}, y_i)], i = 1, \dots, N,\ \boldsymbol{x_i} \in X,\ y_i \in Y = \{-1, +1\}$.

Inicijaliziraj $D_1(i) = {}^1\!/_m$.

Za svaku iteraciju $t = 1, \dots, T$:

- Nađi klasifikator $h_t : X \to \{-1,1\}$ koji minimizira pogrešku s obzirom na distribuciju D_t.

$$h_t = \arg\min_{\mathrm{h_j} \in \mathcal{H}} \epsilon_j\,,\ \ \epsilon_j = \sum_{i=1}^{m} D_t(i)\big[y_i \neq h_j(x_i)\big] \tag{2.1}$$

- Ako je $\epsilon_t \geq \frac{1}{2}$, onda stani.

$$\epsilon_t = \sum_{i:h_t(x) \neq y_i} D_t(i) \tag{2.2}$$

- Postavi $\alpha_t \in R$, pri čemu je ϵ_t pogreška klasifikatora h_t

$$\alpha_t = \frac{1}{2} ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) = \frac{1}{2} ln\left(\frac{1}{\beta_t}\right) \tag{2.3}$$

- Osvježi distribuciju težina:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t \cdot y_i \cdot h_t(x_i)\right)}{Z_t} \tag{2.4}$$

Z_t je normalizacijski faktor odabran tako da je D_{t+1} distribucija vjerojatnosti;

$$Z_t = \sum_{i=1}^{n} D_t e^{-\alpha_t Y_i h_t(x_i)}$$

Izlazni klasifikator je sljedeći:

$$\mathcal{H}(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2.5}$$

Algoritam na ulazu dobiva skup za učenje $(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)$ pri čemu $\boldsymbol{x_i}$ pripada nekoj domeni ili prostoru događaja *X*, a svaka oznaka y_i nalazi se u skupu *Y*. U svakom koraku $t = 1, \ldots, T$ Adaboost iznova poziva bazni algoritam učenja. Osnovna ideja algoritma je stvaranje skupa hipoteza te njihovo kombiniranje težinskim glasanjem većine (engl. *weighted majority voting*) nad klasama predviđenim pojedinim hipotezama. Hipoteze se generiraju treniranjem slabog klasifikatora primjerima odabranim iz iterativno ažurirane distribucije podataka za treniranje. Takvo ažuriranje ili korigiranje distribucije osigurava da su krivo klasificirani primjeri s većom vjerojatnošću uključeni u skup podataka za treniranje narednog klasifikatora. Stoga se uzastopni skupovi podataka za treniranje usmjeravaju prema sve težim primjerima za klasificiranje. Podskup podataka za treniranje S_t svakog od narednih klasifikatora (hipoteza) h_t bira se na temelju distribucije težina $D_t(i)$ nad primjerima za treniranje $x_i, i = 1, \ldots, N$. Početna distribucija je uniformna, tako da svi primjeri imaju jednaku šansu da budu odabrani u početni skup za treniranje. Jedna od glavnih ideja algoritma je održavanje distribucije odnosno skupa težina nad skupom za učenje. Težina distribucije nad uzorkom *i* u koraku *t* označava se sa $D_t(i)$. Početno jednake težine se mijenjaju te se u svakom koraku težine pogrešno klasificiranih primjera povećavaju što prisiljava slabi klasifikator da se usredotoči na teže primjere iz skupa za učenje. Krivo klasificirani uzorak *i* daje $Y_i h_t(x_i) = -1$ pa iz izraza 2.4 slijedi da se težina nad uzorkom i povećava za faktor $e^{\alpha_t}/_{Z_t}$. Isto tako, ispravno klasificirani uzorak daje $Y_i h_t(x_i) = 1$, pa se njegova težina smanjuje za faktor $e^{-\alpha_t}/_{Z_t}$. Zadatak slabog algoritma učenja je pronalaženje slabe hipoteze $h_t: X \to \{-1,1\}$ prikladne distribuciji D_t. Mjera valjanosti hipoteze je pogreška

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i) \tag{2.6}$$

Pogreška treniranja ϵ_t klasifikatora h_t također određuje težine te distribucije na način da je ϵ_t zbroj težina distribucije primjera koji su pogrešno klasificirani hipotezom h_t. Normalizirana pogreška računa se na temelju faktora $\beta_t = \epsilon_t/(1-\epsilon_t)$ pa za $0 < \epsilon_t < 1/2$ vrijedi $0 < \beta_t < 1$. Nakon što su ažurirane težine ponovo normalizirane D_{t+1} postaje trenutna distribucija, a težine krivo klasificiranih primjera se povećavaju. Zahtjev za pogreškom osnovnog modela algoritma učenja (*Wean Learn*) manjom od ½ garantira točnu klasifikaciju bar jednog prethodno pogrešno klasificiranog primjera. Vidljivo je da se pogreška određuje s obzirom na distribuciju D_t koja se koristi za učenje slabog algoritma učenja. U praksi, slabi algoritam učenja može koristiti težine D_t ili kada to nije moguće, podskup primjera za učenje uzorkuje se u skladu sa D_t pa se ti ponovno uzorkovani primjeri koriste za učenje slabog algoritma.

Jednom kada se odredi hipoteza h_t, AdaBoost podešava parametar α_t kako je navedeno. Parametar α_t kojim se korigira vektor težina je funkcija pogreške ϵ_t i intuitivno odražava mjeru važnosti pridruženu hipotezi h_t. Pravilo korekcije težina smanjuje vjerojatnost pridruženu primjerima koje hipoteza dobro predviđa, a povećava vjerojatnost netočno predviđenih.

Nakon što je skup od inicijalno određenog broja T klasifikatora izgrađen, AdaBoost je spreman za klasifikaciju novih primjera nepoznatog razreda. Za razliku od *bagging* ili *boosting* metode, *AdaBoost* koristi ponešto nedemokratsku shemu glasovanja, takozvano težinsko glasanje većine. Ideja je intuitivna: klasifikatori koji su pokazali bolje performanse tijekom treniranja nagrađuju se većim težinama glasovanja. Recipročna vrijednost spomenute normalizirane pogreške - β_t – je stoga mjera učinkovitosti klasifikatora i može se koristiti za dodjeljivanje težina klasifikatorima. Da bi se u slučaju male pogreške klasifikatora izbjegle asimptotski velike vrijednosti ($1/\beta_t$), kao težina za h_t se koristi logaritam od $1/\beta_t$. Konačna odluka ansambla je klasa koja glasovanjem svih klasifikatora na kraju dobije najveći ukupni glas. Konačna hipoteza $\mathcal{H}$ dobiva se većinskom odlukom T pojedinih slabih hipoteza.

Pogreška $\epsilon = Pr_{i\sim D}[H(x_i) \neq y_i]$ konačne hipoteze $\mathcal{H}$ odozgo je ograničena vrijednošću:

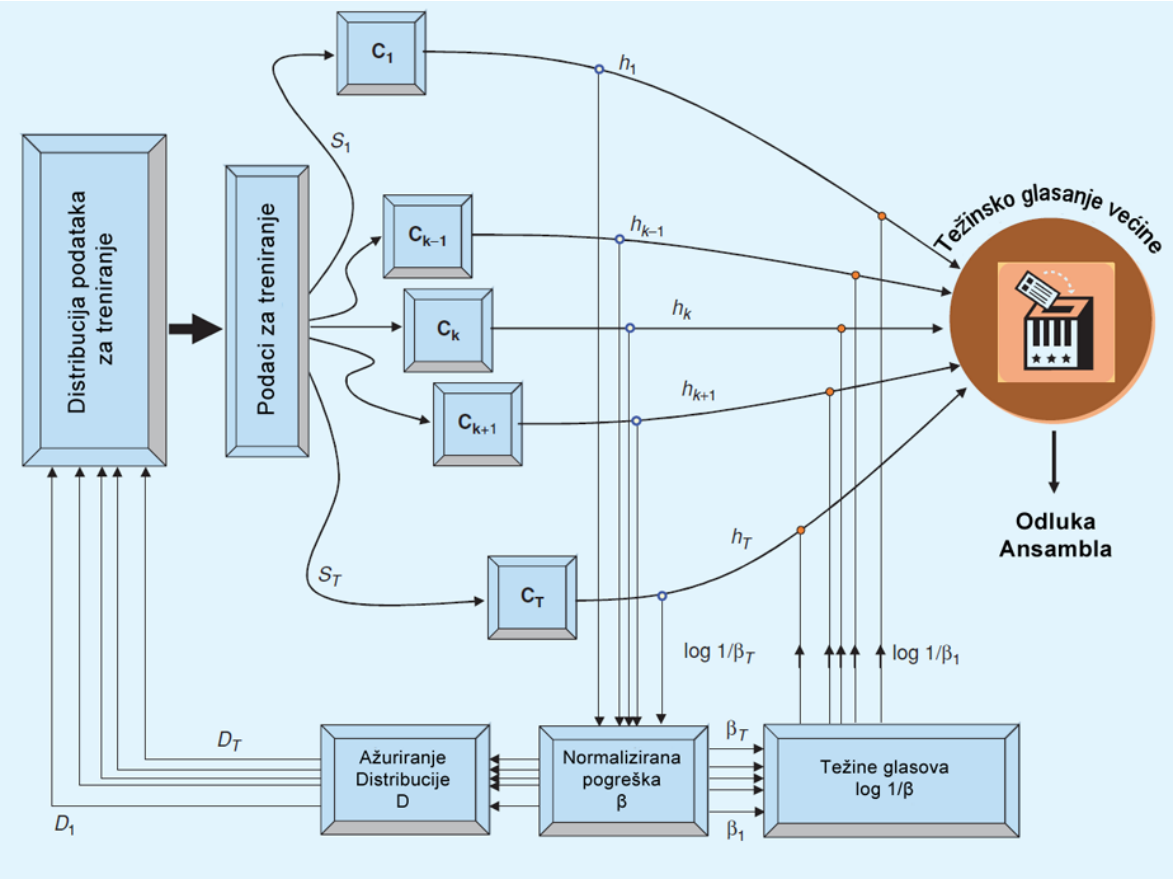
$$\epsilon \leq 2^{T-1} \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} \qquad (2.7)$$

Pri čemu su $\epsilon_1, \ldots, \epsilon_T$ pogreške hipoteza koje generiraju pojedini bazni algoritmi učenja.

Do sada opisani postupak odnosi se na problem binarne klasifikacije. Kad je riječ o binarnoj klasifikaciji, klasifikator preslikava ulazni prostor u binarni izlazni prostor koji sadrži dva moguća stanja, $\{-1,+1\}$. Često je, međutim, potrebno baratati sa $C>2$ klase pa je definicija izlaznog prostora $\{1, 2, \ldots, C\}$. Razvijeno je više inačica AdaBoost algoritma za klasifikaciju u više klasa (9). *AdaBoost.M1* je direktno proširenje osnovnog AdaBoost algorima koji su razvili sami autori. Razlika u odnosu na opisanu inačicu je u izrazu za grešku $|h_j(x_i) - y_i|$ koji se zamjenjuje sa $[\![h_j(x_i) \neq y_i]\!]$, te skup oznaka (klasa), $y_i \in \Omega, \Omega = \{\omega_1, \ldots, w_C\}$. Pri tome se za bilo koju tvrdnju π, $[\![\pi]\!]$ definira kao *1* (logička istina)

ukoliko je π istinita tvrdnja, odnosno *0* (logička laž) u suprotnom. Također, za slučaj *x*, konačna hipoteza $\mathcal{H}$ na izlazu sada daje oznaku *y* koja maksimizira sumu težina slabih hipoteza koje predviđaju tu oznaku

$$h_t = \arg\max_{y \in Y} \sum_{t=1}^{T} \alpha_t [\![h_t(x) \neq y_i]\!] \tag{2.8}$$



Slika 2-5. AdaBoost

Za određeni primjer za učenje (x_i, y_i) hipoteza *h* se koristi da bi se odgovorilo na *C*-1 binarnih pitanja. Za svaku od neispravnih oznaka $y \neq y_i$ postavlja se pitanje: „koja oznaka pripada *x*: y_i ili *y*?“ Drugim riječima tražimo da se točna oznaka y_i diskriminira od netočne oznake *y*. Velika mana *M1* klasifikatora su visoki zahtjevi na točnost baznih algoritama učenja. U slučaju binarne klasifikacije (*C=2*) vjerojatnost slučajnog točnog predviđanja iznosi ½, ali kada je *C*>2, ta vjerojatnost iznosi samo $1/k < 1/2$. Stoga je zahtjev da točnost slabe hipoteze bude veća od ½ mnogo jači od zahtjeva da njena točnost bude tek nešto bolja od slučajnog pogađanja (10).

Blok dijagram algoritma (Slika 2-5) treba interpretirati sekvencijalno: klasifikator C_K stvara se prije klasifikatora C_{K+1}.

AdaBoost.M2 je alternativan algoritam u smislu da prostor oznaka više nije konačan. Takav pristup omogućuje slabom algoritmu učenja mnogo veću fleksibilnost pri donošenju predviđanja. Osim toga, *M2* algoritam uklanja strogi uvjet za pogreškom pojedinog

klasifikatora manjom od ½. Slabi algoritam generira hipoteze oblike $h: X \times Y \rightarrow [0,1]$. Grubo rečeno, *h(x,y)* mjeri stupanj vjerojatnosti da je y ispravno dodijeljena oznaka primjeru *x*. Ako za dani *x*, *h(x,y)* poprimi iste vrijednosti za sve *y* tada se kaže da je hipoteza za primjer *x* neinformativna.

Neke od heurističkih varijacija algoritma modificiraju ili pravilo ažuriranja ili pravilo kombiniranja klasifikatora. Npr. *AveBoost* algoritam uprosječuje težine distribucije da bi se ostvarila što manja korelacija pogrešaka hipoteza. *Learn*$^{++}$ uvodi ovisnost pravila ažuriranja distribucije o pogrešci ansambla što omogućuje učinkovito dodatno učenje novih podataka koji mogu sadržavati još nepostojeće klase.

2.5 Slučajne šume

Slučajna šuma je klasifikacijski algoritam koji su razvili Leo Breiman i Adele Cutler (11). Termin „slučajna šuma odlučivanja“ prvi je predložio Tin Kam Ho 1995. godine (12) . Metoda slučajne šume ujedinjuje *bagging* tehniku koju je razvio Breiman i Ho-ovu metodu slučajnog odabira potprostora (engl. „*random subspace method*“) sa svrhom stvaranja skupa stabala odluke s kontroliranim varijacijama. Osnovna ideja, kao i kod *AdaBoost* metode, je korištenje mnogo, pojedinačno slabih, klasifikatora (stabala odluke) na temelju čijih se izlaza donosi odluka o pripadnosti uzorka nekoj od klasa. Prednosti *random forest* klasifikatora su visoka točnost, mogućnost paralelne implementacije, sprečavanje pretreniranosti (engl. *overfitting*), otpornost na šum i *outlier-e*, te efikasnost u rukovanju širokim spektrom podataka (velik broj ulaznih varijabli, podaci nepoznatih distribucija, visoko-dimenzionalni podaci). Slučajni odabir svojstava omogućuje pogrešku usporedivu sa AdaBoost algoritmom, ali uz mnogo veću robusnost na šum.

2.5.1 Stabla odluka (Decision trees)

Stablo odluke je klasifikator stablaste strukture koji se sastoji od jednog ili više hijerarhijski povezanih čvorova. Svaki od čvorova u stablu može biti *korijenski* (engl. *root node*, nema ulaznih te niti jedan ili više izlaznih bridova), *unutarnji* (engl. *internal node*, ima jedan ulazni i dva ili više izlazna brida) ili *završni* čvor odnosno *list* (engl. *Leaf/terminal node*, točno jedan ulazni i niti jedan izlazni brid). U svakom unutarnjem čvoru ispituje se uvjet nad nekim atributom ulaznog skupa podatka, a svaka grana stabla predstavlja jedan od ishoda ispitivanja. Listovi stabla predstavljaju razrede ili distribucije razreda. Podatak koji se želi klasificirati dovodi se na korijenski čvor te se nizom odluka u unutarnjim čvorovima „spušta“ niz stablo sve do završnog čvora odnosno pripadne klase (13).

Stablo odluke gradi se na temelju skupa atributa ulaznih podataka a mogući broj takvih stabala raste eksponencijalno s obzirom na broj atributa. Neka od mogućih stabala su točnija od ostalih a nalaženje optimalnog stabla u principu je računalno nemoguć zadatak zbog eksponencijalne veličine prostora traženja. Zato su razvijeni učinkovitiji algoritmi za nalaženje suboptimalnih ali prihvatljivo točnih stabala odluke. Takvi algoritmi uglavnom se koriste pohlepnom strategijom gradeći stablo stvaranjem niza lokalno optimalnih odluka o izboru atributa na temelju kojeg se vrši podjela podataka. Najčešće korišteni i

općeprihvaćeni algoritmi za induktivnu izgradnju stabla kao što su *ID3*, *C4.5* i *CART* (14), zasnivaju se na Huntovom algoritmu opisanom u nastavku.

Huntov algoritam podrazumijeva rekurzivan rast stabla podjelom podataka za učenje u sukcesivno čišće podskupove. Neka je D_t skup podataka za treniranje pridruženi čvoru *t*, a $y = \{y_1, y_2, ..., y_c\}$ neka su oznake klasa. U sljedećim točkama sadržana je rekurzivna definicija Huntovog algoritma (15).

1. Ako svi zapisi iz D_t pripadaju istoj klasi y_t, tada je *t* završni čvor (list, engl. *leaf*) sa oznakom y_t.
2. Ukoliko D_t sadrži zapise koji pripadaju više klasa bira se atribut za podjelu podataka u manje podskupove. Za svaki ishod testnog uvjeta stvara se novi čvor (engl. *child node*) i u njih se na temelju ishoda distribuiraju zapisi iz D_t . Algoritam se zatim rekurzivno primjenjuje na svaki stvoreni čvor – dijete.

Nužan uvjet je da skup za treniranje sadrži sve moguće kombinacije vrijednosti atributa i da svaka kombinacija ima jedinstvenu oznaku klase. Takvi zahtjevi su prestrogi za većinu realnih slučajeva. Stoga su potrebni dodatni uvjeti za rješavanje sljedećih slučajeva:

- Moguće je da je neko od čvorova djece iz koraka 2 prazno odnosno da nema podataka pripisanih tom čvoru. Do toga dolazi ako niti jedan od zapisa za treniranje nema kombinaciju atributa povezanu sa tim čvorom. U tom slučaju čvor se proglašava završnim i dodjeljuje mu se oznaka klase kojoj pripada većina zapisa pridružena čvoru roditelju.
- Ukoliko u koraku 2 svi zapisi iz D_t imaju iste vrijednosti atributa (osim oznake klase) tada njihova daljnja podjela nije moguća. U tom slučaju čvor se proglašava završnim i dodjeljuje mu se oznaka klase kojoj pripada većina zapisa pridruženih tom čvoru.

Dva su krucijalna problema koje svaki algoritam učenja za induktivnu izgradnju stabla mora riješiti:

- Način podjele podataka za treniranje. U svakom rekurzivnom koraku tijekom procesa rasta stabla odabire se atribut na kojemu se temelji testni uvjet pripadnog čvora i na temelju kojega se podaci čvora dijele u manje podskupove. Za uspješnu implementaciju ovog koraka, algoritam mora pružiti metodu određivanja testnih uvjeta za različite tipove atributa kao i objektivnu mjeru za izražavanje dobrote svakog od testnih uvjeta.
- Način i trenutak zaustavljanja grananja. Da bi se završila izgradnja stabla, potreban je neki uvjet zaustavljanja. Jedna od mogućih strategija je nastavak širenja čvorova sve dok ili svi podaci ne pripadaju istoj klasi ili dok svi nemaju iste vrijednosti atributa. Iako su oba uvjeta dovoljna mogu se koristiti i drukčiji uvjeti ranijeg zaustavljanja rasta.

Svaki algoritam učenja ujedno mora biti sposoban izraziti testne uvjete za različite tipove atributa. Atributi po tipu mogu biti binarni, nominalni, ordinalni ili kontinuirani.

2.5.2 Mjere za odabir najboljeg grananja

Postoji više mjera za određivanje najboljeg načina podjele podataka prilikom grananja čvora. Te mjere su definirane s obzirom na distribuciju klasa podataka prije i nakon podjele .

Neka $p(i|t)$ označava udio koji u danom čvoru *t* pripadaju klasi *i* (oznaka čvora *t* ponekad se zanemaruje pa se taj omjer može označiti i sa p_i). Mjere za biranje najboljeg grananja često se temelje na *stupnju nečistoće* čvorova djece. Što je manju stupanj nečistoće, to je nepravilnija distribucija klasa.

Neke od tipičnih mjera nečistoće (engl. *impurity measure*) su:

$$Entropija(t) = -\sum_{i=0}^{C-1} p(i|t)\log_2 p(i|t) \tag{2.9}$$

$$Gini(t) = 1 - \sum_{i=0}^{C-1}\left[p(i|t)\right]^2 \tag{2.10}$$

$$Greška\ klasifikacije(t) = 1 - \max_i p(i|t) \tag{2.11}$$

Pri čemu *C* označava broj klasa. Odabir atributa za testiranje može ovisiti o primijenjenoj mjeri nečistoće. Za utvrđivanje učinkovitosti testnog uvjeta potrebno je usporediti stupanj nečistoće roditeljskog čvora (prije podjele) sa stupnjem nečistoće čvorova djece (nakon podjele). Što je njihova razlika veća to je odabrani testni uvjet bolji. Poboljšanje nastalo podjelom čvora naziva se *dobit* (engl. *gain*) i označava sa Δ. Dobit se koristi za određivanje „dobrote“ grananja i definira se na sljedeći način:

$$\Delta = I(roditelj) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{2.12}$$

Pri tome se: $I(\cdot)$ mjera nečistoće čvora, *N* ukupan broj podataka u roditeljskom čvoru, *k* broj atributa, te $N(v_j)$ broj podataka pridruženih čvoru djetetu v_j. Indukcijski algoritmi za izgradnju stabala odluka najčešće biraju uvjete testiranja koji maksimiziraju dobit Δ. Pošto je nečistoća čvora roditelja jednaka za sve testne uvjete, maksimiziranje dobiti ekvivalentno je minimiziranju srednje vrijednosti mjera nečistoća čvorova djece. Kada se kao mjera nečistoće koristi entropija, tada se razlika entropija naziva *informacijska dobit* Δ_{info} (engl. *information gain*).

Mjere nečistoće kao što su entropija i Gini indeks teže atributima sa većim domenama vrijednosti odnosno preferiraju ulazne atribute koji imaju veći broj različitih vrijednosti. Dodavanje takvog atributa u stablo odluke često može rezultirati slabijom točnošću generalizacije. Takve slabosti algoritama uklanjaju se ili ograničavanjem uvjeta testiranja na binarne ishode ili modifikacijom kriterija podjele tako da se u obzir uzme i broj ishoda

koje proizvodi uvjet testiranja. Potonja strategija zapravo se svodi na normalizaciju mjera nečistoća, kao što je npr. *faktor dobitka* (engl. *gain ratio*):

$$Faktor\ dobitka(a_k) = \frac{Dobit(a_k)}{Entropija(a_k)} = \frac{\Delta}{-\sum_i p_i \log_2 p_i} \tag{2.13}$$

Rast stabla nastavlja se do trenutka zadovoljenja nekog kriterija zaustavljanja. Najčešća pravila zaustavljanja su uvjeti poput:

- Svaki od primjera iz skupa za treniranje pripada jednoj od mogućih klasa
- Dosegnuta je maksimalna dubina stabla
- Broj uzoraka u završnom čvoru manji je od minimalnog broja uzoraka koje roditeljski čvor mora sadržavati
- Ako se čvor grana, broj uzoraka u jednom ili više čvorova djece manji je od minimalnog broja uzoraka za čvor dijete
- Kriterij najboljeg grananja manji je od postavljenog praga

Korištenje striktnih pravila kao kriterija zaustavljanja rasta stabla stvara mala podtrenirana stabla. Preblagi uvjeti s druge strane dovode do glomaznih i pretreniranih stabala odluka. Zlatnu sredinu čini metodologija smanjivanja ili podrezivanja (engl. *pruning*) koju je razvio Breiman a koja se zasniva na labavim kriterijima zaustavljanja koja omogućuju stablu odluke potpun rast i pretreniranost s obzirom na podatke za učenje. Zatim se takvo pretrenirano stablo „podrezuje“ odnosno reducira u manje stablo uklanjanjem podgrana koje ne doprinose točnosti generalizacije. Opisani postupak zapravo je poseban slučaj tehnika smanjivanja stabla; stablo se općenito može smanjivati metodom „unaprijed“ (engl. *pre-pruning*) ili „unatrag“ (engl. *post-pruning*).

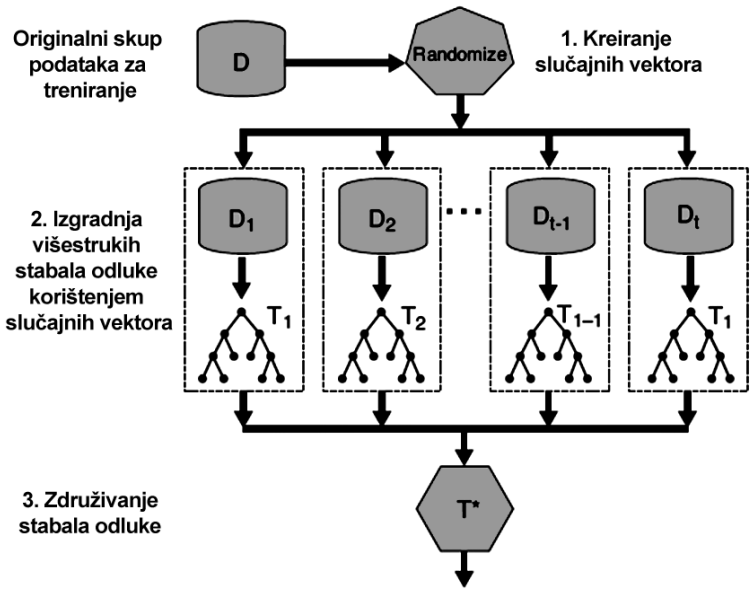
Kod smanjivanja stabla unaprijed prvo se odredi prag ili pravilo kada će se prestati dodavati podstabla. Prednost takvog pristupa je izbjegavanja kompleksnih i pretreniranih podstabala, ali se pokazao teško izvedivim u praksi, pa se te metode ne koriste u većini algoritama.

Alternativni pristup dopušta maksimalni rast stabla koje se zatim smanjuje unazad, „odozdo prema gore“. Smanjivanje se vrši *zamjenom podstabla* (engl. *subtree replacement*) završnim čvorom kojemu se dodjeli oznaka klase određena distribucijom klasa svih primjera sadržanih u podstablu, ili *podizanjem podstabla* (engl. *subtree raising*). Pokazalo se da metode podrezivanja mogu poboljšati generalizacijski učinak stabla odluke, naročito kada je u podacima prisutan šum.

2.5.3 Opis Random Forest algoritma

Slučajna šuma je općenit naziv za skupinu metoda koje se koriste kolekcijom stablastih klasifikatora $\{h(\boldsymbol{x}, \theta_k), k = 1, \ldots\}$ pri čemu je $\{\theta_k\}$ skup nezavisnih slučajnih vektora jednake distribucije, a $\boldsymbol{x}$ ulazni vektorski uzorak. Svako od stabala daje svoj glas za ulazni

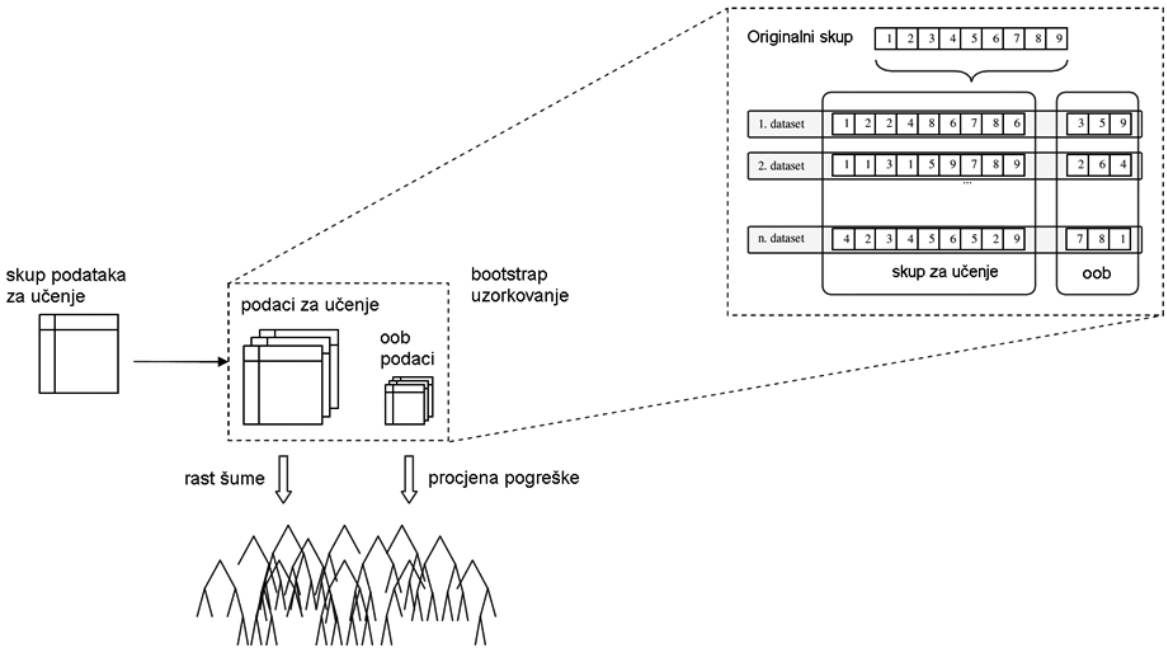
uzorak te šuma određuje klasu na temelju većine glasova. Klasifikacija ulaznog uzorka vrši se po pravilu većine a na temelju izlaza svakog od pojedinih klasifikatora (Slika 2-6).



Slika 2-6. Random Forest

Prilikom treniranja, algoritam slučajne šume stvara velik broj stabala, a skup za treniranje pojedinog stabla dobiva se *bagging* metodom, odnosno izdvajanjem *N* uzoraka iz početnog skupa za treniranje iste veličine slučajnim odabirom s ponavljanjem. Tako odabran skup za treniranje je iste veličine kao i originalan skup, te se neki od primjera mogu pojaviti više puta, a nekih uopće nema. Skupovi tako odabranih primjera nalaze se u korijenima stabala u šumi. Ostali, neodabrani uzorci čine oko trećinu početnog skupa i to su tzv. *out-of-bag* primjeri. Interni estimatori koriste *out-of-bag* primjere za praćenje i procjenu nepristrane greške klasifikacije, generalizacijske pogreške, snage i korelacije pojedinih stabala, kao i za procjenu važnosti pojedinih varijabli (atributa) ulaznih instanci (Slika 2-7).

Osim formiranja skupa za treniranje, za svako od stabala šume po principu slučajnog odabira podprostora slučajnim se odabirom bira *m* od ukupno *M* atributa te se koriste oni koji omogućavaju najbolje grananje. Vrijednost *m* određuje se unaprijed i konstantna je za cijelu šumu. Optimalna vrijednost parametra *m* najčešće nije fiksan broj nego neki raspon u intervalu $[1, M]$. Često se koriste vrijednosti $m = \sqrt{M}$ ili $m = int(\log_2 M + 1)$. Atributi najpogodniji za grananje nalaze se na temelju mjera nečistoće. Atribut sa najvećom vrijednošću indeksa nečistoće je pobjednik i na temelju njega se vrši grananje.



Slika 2-7. Podjela skupa za učenje

Pojedina stabla odluke grade se na sljedeći način:

1. Neka je *N* broj primjera za treniranje a *M* broj atributa
2. Skup podataka za treniranje stabla se formira uzorkovanjem sa zamjenama uzoraka iz ukupnog skupa za treniranje; takva tehnika poznata je pod nazivom „*bootstrapping*“ (16). Broj primjera u takvom skupu jednak je ukupnom broju primjera u skupu podataka za učenje. Taj novi skup može sadržavati više kopija istog primjera iz skupa za učenje. Korištenjem *bootstrapping* tehnike, obično jedna trećina skupa za učenje nije prisutna u tom skupu. Taj preostali dio podataka su tzv. „*out-of-bag*“ podatci („*oob*“) koji se između ostalog koriste za procjenu pogreške stabla.
3. Za svaki čvor stabla nasumce odabrati *m* atributa na temelju kojih se donosi odluka. Korištenjem uobičajenih algoritama za izgradnju stabala, ti atributi zatim formiraju čvorove i listove.
4. Svako stablo raste u najvećoj mogućoj mjeri bez potkresivanja (engl. *pruning*, odbacivanje pojedinih čvorova iz stabla).

Ovakvim postupkom grade se pojedina slučajna stabla. Nakon izgradnje stabla, *out-of-bag* primjeri se koriste za testiranje kako pojedinih stabala tako i čitave šume. *Out-of-bag* procjena pogreške je broj pogrešnih klasifikacija uprosječen po svim stablima. Takva procjena pogreške korisna je za predviđanje učinkovitosti algoritma i bez korištenja skupa za testiranje.

Generalizacijska pogreška šume ovisi o snazi pojedinih stabala i njihovoj međusobnoj korelaciji. Unutarnji estimatori koji nadziru pogrešku, snagu i korelaciju odražavaju utjecaj porasta broja svojstava koja se koriste pri podjeli. Tim estimatorima se također mjeri važnost pojedinih varijabli.

2.5.4 Točnost slučajnih šuma

Uz dan ansambl klasifikatora $h_1(x), h_2(x), \ldots, h_K(x)$, te skup podataka za treniranje odabran nasumce iz distribucije slučajnih vektora *Y*, *X*, definicija funkcije margine je sljedeća:

$$mg(X,Y) = av_k I(h_k(X) = Y) - \max_{y=Y} av_k I(h_k(X) = j) \tag{2.14}$$

pri čemu je $I(\cdot)$ indikatorska funkcija. Margina je mjera koja odražava koliko prosječan broj glasova za ispravnu klasu nadmašuje prosječan broj glasova za bilo koju drugu klasu. Veća margina znači veću pouzdanost klasifikacije. Greška generalizacije:

$$PE^* = P_{X,Y}(mg(X,Y) < 0) \tag{2.15}$$

pri čemu indeksi *X*,*Y* upućuju na vjerojatnost definiranu nad prostorom *X*,*Y*. Kako se povećava broj stabala, za gotovo sve uzorke θ, PE^* konvergira prema

$$P_{X,Y}\left(P_\theta(h(X,\theta) = Y) - \max_{y \neq Y} P_\theta(h(X,\theta) = j) < 0\right) \tag{2.16}$$

Iz izraza se vidi da povećanje broja stabala ne dovodi do pretreniranosti već do granične vrijednosti pogreške generalizacije. Gornja granica generalizacijske pogreške iznosi:

$$PE^* \leq \frac{\bar{\rho}(1-s^2)}{s^2} \tag{2.17}$$

$\bar{\rho}$ je srednja vrijednost korelacije, $\bar{\rho} = E_{\theta,\theta'}(\rho(\theta,\theta')sd(\theta)sd(\theta'))/E_{\theta,\theta'}(sd(\theta)sd(\theta'))$, a s predstavlja snagu skupa klasifikatora $\{h(\boldsymbol{X},\theta)\}$.

3 PODACI I METODE

3.1 Atributi za predviđanje

Cilj rada je procijeniti uspješnost Adaboost te metode slučajne šume pri predviđanju sekundarne strukture aminokiselinskih ostataka. Pri tome je nužno odabrati prikladan skup svojstava koja dobro opisuju informacije bitne za uspješno predviđanje. Atributi odabrani za predviđanje sekundarne strukture aminokiselinskih ostataka su:

1. slijed od n (9 ili 11) aminokiselinskih ostataka
2. profili slijeda za svaki od ostataka.

Time se dobiva ukupno $n \cdot 21$ ulaznih atributa. Ciljni, izlazni atribut je jedna od tri klase sekundarne strukture – *H* (uzvojnica), *E* (ploča) i *C* (ostalo).

Za predviđanje sekundarne strukture aminokiselinskih ostataka proteinskih lanaca korištena su tri standardna skupa podataka koji se često koriste pri testiranju te procjeni modela i metoda predviđanja sekundarne strukture: RS126, CB513 i PSIPRED.

RS126 skup ne-homolognih proteina konstruirali su Rost i Sander pri razvoju PHD (*Profile network from HeiDelberg*) metode za predviđanje sekundarne strukture proteina (17). Ime skupa dolazi od inicijala autora i broja proteinskih lanaca koje sadrži. Ukoliko se dva, najmanje 80 rezidua dugačka lanca podudaraju u 25% ili više dijela slijeda, smatraju se homolognima i samo jedan od njih se uključuje u skup.

CB513, kao i RS126 potječe od inicijala autora skupa te broja proteinskih lanaca u njemu. Konstruirali su ga Cuff i Barton (18) združivanjem RS126 i vlastitog CB396 skupa uz odbacivanje 9 lanaca koji su se pokazali redundantnima.

PSIPRED skup proteinskih lanaca je nastao pri razvoju istoimene metode za predviđanje sekundarne strukture proteina (19). Sljedovi su sakupljeni na temelju više javno dostupnih baza, te se uklanjanjem ponavljajućih sljedova dobio skup od oko 340000 sljedova iz kojeg su SEG programom naknadno uklonjena područja niskog sadržaja informacije. Dodatnim filtriranjem baze uklonjena su područja transmembranskih i pletenastih (engl. *coiled-coil*) struktura te je dobiven konačan skup od 2245 proteinskih lanaca.

3.1.1 Baze primarnih sljedova

Sljedovi proteina u početku su se dobivali izravno iz izdvojenih molekula proteina (sekvencioniranje proteina, gel elektroforeza, masena spektrometrija), ali se velika većina novo identificiranih proteina dobiva iz DNA sljedova. Potrebno je razlikovati dvije vrste primarnih sljedova – sljedove nukleotidnih baza te sljedove proteina koji se dobivaju njihovim prevođenjem. Postoje raznolike baze primarnih sljedova, od jednostavnih repozitorija, koji sadrže podatke bez ili sa nedovoljno podataka o zapisima, pa do stručno održavanih baza koje pokrivaju sve vrste i u kojima su podaci o sljedovima nadopunjeni informacijama o pojedinim zapisima (20).

Najveći i najpoznatiji repozitoriji sljedova baza su GenBank, EMBL (*European Molecular Biology Laboratory*) baza sljedova nukleotida i DDBJ (*DNA Data Bank of Japan*). Primarni sljedovi proteina dobiveni prevođenjem nukleotidnih slijedova navedenih baza objedinjeni su na jednom mjestu u GenPept (*GenBank Gene Products*) bazi podataka (21). GenBank je baza podataka dobro opisanih, to jest anotiranih, sljedova koju održava Nacionalni centar za biotehnološke informacije (NCBI). Cjeloviti pregled trenutne verzije GenBank baze podataka nalazi se na poslužitelju centra NCBI za prijenos podataka (engl. "*NCBI ftp site*"). Novo izdanje baze podataka GenBank izdaje se svaka dva mjeseca. Podacima se može pristupiti na nekoliko načina. Dva najčešće korištena su sustav za pristup podacima *Entrez,* koji povezuje Genbank sa ostalim bazama koje održava NCBI, te pretraživanje pomoću BLAST programa.

Smisao repozitorija podataka je da korisnicima pruži što brži pristup novim sljedovima, ali dodavanje informacija o slijedu korisnicima pruža neizmjerno veću korist. Takve stručno održavane baze podataka obogaćuju sljedove podataka visoko pouzdanim dodatnim informacijama koje validiraju stručnjaci. Najopsežnije „stručne“ baze sljedova proteina su PIR-PSD (*Protein Information Resource Protein Sequence Database*), SwissProt, MIPS (*Munich Information Center for Proteins*), TrEMBL (*Translations from EMBL*) te 3D strukture u PDB bazi. 2003 godine odlukom NIH (*National Institutes of Health*) pokrenut je projekt UniProt koji objedinjuje PIR-PSD, Swiss-Prot i TrEMBL baze u jedinstven izvor podataka i sastoji se od tri dijela: UniProt Knowlddgebase koji nastavlja sa radom Swiss-Prot, TrEMBL i PIR baza pružajući stručno vođenu bazu, UniParc (*UniProt Archive*) u koju se na dnevnoj bazi dodaju i nadopunjavaju zapisi; te UniProt NREF (*UniProt non-redundant reference database*) koji sve zapise sjedinjuje u obliku nereduntantog skupa sljedova (22).

3.1.2 Protein Dana Bank

Informacije o proteinskim lancima koje sadrže skupovi korišteni u radu dohvaćene su iz RCSB (*Research Collaboratory for Structural Bioinformatics*) PDB (*Protein Dana Bank*) baze podataka. PDB je središnji globalni repozitorij strukturnih informacija o biološkim makromolekulama koji uz eksperimentalno utvrđene trodimenzionalne strukture (prostorni položaj svakog od atoma unutar strukture) sadrži mnoge druge korisne informacije poput bibliografskih navoda, informacije o primarnoj i sekundarnoj strukturi, razlučivosti, atomske povezanosti i mnoge druge. Baza trenutno sadrži više od 64000 zapisa (ožujak 2010) i taj broj zadnjih godina sve brže raste. Većina zapisa u bazi su strukture proteina, međutim, manji dio čine i nukleinske kiseline, ugljikohidrati i teoretski modeli. Većina struktura određena je tehnikom rendgenske kristalografije, dok su ostale dobivene nuklearnom magnetskom rezonancijom (NMR) te neznatan broj elektronskom mikroskopijom i hibridnim metodama. Iako PDB sadrži velik broj zapisa, većina proteinskih struktura je redundantna tj. postoji više zapisa istog proteina ali dobivenih pod različitim uvjetima, ili uz drukčiju razlučivost.

Podaci su sa poslužitelja preuzeti u obliku *.pdb* datoteka iz kojih su ekstrahirane informacije o vrsti i redoslijedu aminokiselinskih ostataka (primarna struktura) te koordinate pojedinih atoma u proteinskom lancu.

3.2 Sekundarna struktura

Trodimenzionalne strukture odnosno položaji pojedinih atoma unutar lanaca ne pružaju direktnu informaciju o sekundarnoj strukturi nužnu za izgradnju skupova za treniranje i testiranje odabranih metoda predviđanja. Međutim, elementi sekundarne strukture definirani su svojim fizikalno-kemijskim parametrima i moguće ih je odrediti iz kristalne strukture proteina. Uz poznate 3D koordinate atoma proteinske strukture moguće je odrediti kojoj od sekundarnih struktura pripada pojedini aminokiselinski ostatak. Periodične sekundarne strukture proizvode pravilnosti koje se očituju u nekim prostorno-kemijskim uzorcima (npr. C_α udaljenosti, prostorni kutovi poput α kutova ili ϕ/ψ parova kutova i određenih uzoraka vodikovih veza) te se mogu iskoristiti za njihovo međusobno razlikovanje. Od mnogih postojećih računalnih metoda, za potrebe ovog rada odabrana je DSSP (*Dictionary of Secondary Structure in Proteins*) metoda za automatizirano pridruživanje sekundarne strukture. DSSP (23) razvrstava aminokiselinske ostatke u jedan od 8 tipova sekundarne strukture, a na temelju prepoznavanja uzoraka vodikovih veza i geometrijskih svojstava. 7 karakterističnih elemenata sekundarne strukture uključuje α-uzvojnicu (*H*), β-vrpcu (*E*), α_5 (*G*) i π-uzvojnicu (*I*), izolirani β-most (*B*), luk (*S*), zavoj (*T*), dok se ostale rezidue tretiraju kao nestandardne (*C*).

Tablica 3-1"8/3" redukcijska shema

Redukcija	DSSP kod	Opis
H	H	α-uzvojnica
	G	α_5-uzvojnica
E	E	β – nit
	B	Izolirani β-most
C	I	π-uzvojnica
	T	okret
	S	Luk
	-	Omča/namotaj

Uz pomoć DSSP programa i skripte napisane u Perl programskom jeziku za svaki proteinski lanac napravljena je pripadna *.dssp* datoteka sa informacijama o sekundarnoj strukturi. Predviđanje sekundarne strukture uobičajeno razlikuje samo tri ciljna tipa: uzvojnicu (*H*), ploču (*E*) i ostalo (*C*), pa se javlja problem prevođenja strukturne DSSP abecede od 8 znakova u reduciranu troslovnu abecedu. Postoji više redukcijskih shema konverzije, a u ovom radu primijenit će se shema korištena u PSIPRED metodi (Tablica 3-1).

Na temelju *SEQRES* polja u PDB datoteci i DSSP izlaza, te uz pomoć BioPerl programskog paketa, za svaki proteinski lanac napravljena je tekstualna datoteka sljedećeg oblika (Slika 3-1):

#ChainID	#PDB SEQRES	#DSSP	#SS
.	.	.	.
176	L	L	E
177	L	L	C
178	S	?	?
179	R	?	?
.	.	.	.

Slika 3-1 **1BKS_A.txt**

3.3 Višestruko poravnanje slijedova

Neposredan rezultat molekularne evolucije su varijacije u sljedovima bioloških makromolekula. Sličnost u strukturi sljedova (sličan slijed nukleinskih baza ili aminokiselinskih ostataka), a samim time i u funkciji molekula, može se uočiti u sljedovima koji dijele zajedničkog pretka. Dva slijeda su homologna ako imaju zajedničko porijeklo i tada se mogu grupirati u homologne familije. Homolognost je ključno svojstvo jer porodice proteina često dijele određen broj zajedničkih karakteristika te pripadnost određenog slijeda nekoj od familija može uvelike pomoći pri identifikaciji funkcija pripadne strukture.

Usporedi li se novi, nepoznat slijed (dobiven sekvenciranjem genoma) sa bazom podataka dobro opisanih (anotiranih) sljedova, moguće mu je predvidjeti biološku strukturu. Samim time, moguće je predvidjeti i njegovu funkciju. Kako je u kasnim '70-ima počeo rasti broj dostupnih sljedova DNA i proteina, raslo je i zanimanje za razvojem računalnih programa za analizu istih.

Prvotne metode upotrebljavale su grubu usporedbu (poravnanje) i nisu bile primjenjive na sljedove koji se nedovoljno poklapaju, ili koji sadržavaju umetanja (insercije) i brisanja (delecije). Također je bilo teško pronaći poravnanja sljedova koji nisu dovoljno srodni ili su vrlo dugački.

Prilikom traženja sličnosti u sljedovima DNA ili proteina, standardno programiranje bi zbog količine podataka oduzimalo previše vremena, te ne bi dalo statistički značajne rezultate. Zbog toga se upotrebljava dinamičko programiranje. Dinamičko programiranje je računalna metoda koja se upotrebljava za poravnanje slijedova aminokiselina ili parova nukleotida, te daje optimalno poravnanje dvaju ili više sljedova. Poravnanje se vrši uspoređivanjem svakog para odgovarajućih znakova u slijedu i dodjeljivanjem ocijene svakoj takvoj usporedbi. Pritom se u obzir uzimaju i identičnosti, sličnosti, te praznine i različitosti, tako da se svakoj od tih usporedbi dodjeljuje različit broj bodova. Nakon toga se bodovi zbrajaju, te se u obzir uzima ono poravnanje koje je dobilo statistički najveći broj bodova. Ova je metoda dovoljno pouzdana da biolozima može dati korisne

informacije o sličnosti sljedova. Te su informacije važne za ostvarivanje funkcionalnih, strukturnih i evolucijskih predviđanja vezanih uz molekule DNA ili proteina.

Najčešće upotrebljavane inačice dinamičkog programiranja su globalno i lokalno poravnanje. Programi za globalno poravnanje vrše poravnanja duž cijele duljine slijeda i temelje se na Needleman-Wunsch algoritmu. Prilikom pronalaženja najboljeg poravnanja, algoritam pokušava obuhvatiti čim dulje dijelove sljedova, počevši od krajnje lijevog kraja do krajnje desnog kraja slijeda. Ustanovljeno je međutim da su biološki najvažniji dijelovi u sljedovima DNA i proteina oni koji se nalaze u specifičnim dijelovima, te koji se gotovo potpuno poklapaju. Također je ustanovljeno da ostali dijelovi (koji ne pokazuju dovoljnu sličnost) nisu od važnosti, te da su insercije i delecije vrlo česta evolucijska promjena. Zbog toga je razvijen Smith-Waterman (inačica algoritma Needleman-Wunsch) algoritam. Programi za lokalno poravnanje upotrebljavaju Smith-Waterman algoritam i uspoređuju kraće, lokalne dijelove oba slijeda prilikom pronalaženja lokalnog poravnanja odnosno lokalne sličnosti. U praksi se češće upotrebljava Smith-Waterman algoritam jer stvarni biološki sljedovi često ne pokazuju međusobnu sličnost duž cjelokupnog niza već samo na kratkim, lokalnim, dijelovima (24).

Preciznost poravnavanja, koju pružaju algoritmi dinamičkog programiranja, u suprotnosti je s brzinom rada. Veličina i opseg baza podataka ne dopušta da se, u današnje vrijeme, upotrebljavaju spore metode. Zbog toga su razvijene alternativne, brže metode koje upotrebljavaju drugačije principe a temelje se na heurističkim algoritmima Pritom je program, dobivši na brzini, izgubio na preciznosti.

FASTA (1985) je najstariji alat za pretraživanje baza sljedova i njihovu usporedbu. Traženje homolognih proteina ostvaruje se usporedbom 'riječi' (k-torki) između dva slijeda te ocjenom sličnosti ovisnoj o razini podudaranja (25). BLASTP (1990) koristi matrice sličnosti (BLOSUM ili PAM matrice) za uspoređivanje upitnog slijeda sa onima iz baze. Najčešće upotrebljavan jest računalni program BLAST ("*Basic Local Alignment Search Tool*"). koji je prosječno pedeset puta brži od algoritma dinamičkog programiranja. Algoritam koji se upotrebljava u programu BLAST temelji se na statističkim metodama koje su razvili Altschul i suradnici (26). BLAST uspoređuje, tj. poravnava nepoznati slijed („upit“, engl. "*query*"), sa svakim slijedom koji se nalazi u bazi sljedova od interesa. Tijekom usporedbe program BLAST traži dijelove sljedova, tzv. „riječi“, koji pokazuju najveći stupanj sličnosti. Svaka riječ predstavlja minimalni dio DNA ili proteina (11 za DNA, 3 za proteine), potreban da se dobije broj riječi dovoljno velik da se prihvati kao značajan, ali ne toliko velik da bi se previdjeli neki kratki, ali značajni dijelovi. Kao rezultat, iz baze podataka se izdvajaju oni sljedovi koji sadržavaju dio DNA ili proteina dovoljno sličan nepoznatom slijedu da bi se smatrao statistički značajnim. Tijekom poravnavanja, za bodovanje sličnosti između sljedova program BLAST upotrebljava jednu od supstitucijskih matrica za bodovanje. Supstitucijske matrice određuju vjerojatnost da se određena aminokiselina tijekom evolucije zamijeni nekom drugom. Supstitucija je vjerojatnija između aminokiselina sa sličnim biokemijskim svojstvima. Tako, na primjer, hidrofobne aminokiseline izoleucin (*I*) i valin (*V*) imaju pozitivnu vrijednost u matrici, što znači da je vjerojatnost da će jedna tijekom vremena zamijeniti drugu relativno visoka. S

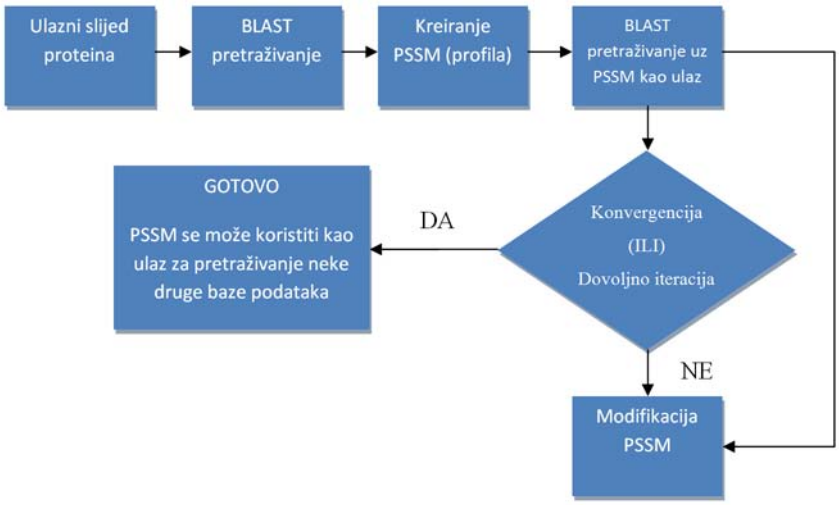
druge strane, vjerojatnost zamjene hidrofobnog izoleucina sa hidrofilnim cisteinom (*C*) je vrlo mala, pa će taj par aminokiselina u supstitucijskoj matrici imati negativnu vrijednost. Najčešće se upotrebljavaju supstitucijske matrice PAM (27) i BLOSUM (28) serije. PAM matrice se zasnivaju na eksplicitnom evolucijskom modelu, dok se BLOSUM matrice zasnivaju na implicitnom modelu evolucije.

Navedeni alati koriste poravnanje u parovima za traženje homolognih sljedova. Napredniji pristup predstavljaju alati koji vrše višestruko poravnanje sljedova i koji su redovito osjetljiviji u traženju udaljenijih homolognih parova.

Višestruko poravnanje sljedova (engl. *multiple sequence alignment - MSA*) otkriva informacije o homolognosti, evolucijskoj i strukturalnoj sličnosti proteina a temelji se na identifikaciji homolognih sljedova pretraživanjem proteinskih baza podataka. Višestruka poravnanja su rezultiraju vjerojatnosnim profilima slijeda (engl. *probability profiles*) odnosno vjerojatnošću pojavljivanja bilo koje od 20 aminokiselina na onome mjestu u slijedu na kojemu se nalazi aminokiselina čiji se profil određuje (29).

Sljedovi na temelju kojih se gradi višestruko poravnanje mogu se odabrati na dva načina: ručnim odabirom ili automatiziranim mehanizmom pretraživanja pojedinih baza. U potonjem slučaju višestruko poravnanje se vrši nad pronađenim homolognim sljedovima. PSI-BLAST (30), punog naziva *Position Specific Iteration BLAST*, inačica je BLAST algoritma u kojemu se profil, odnosno matrica vjerojatnosti (engl. *Position Specific Scoring Matrix,* PSSM) pronalaženja svake od 20 aminokiselina na mjestu aminokiseline čiji se profil traži, gradi iz višestrukog poravnanja sljedova te najviše ocjenjenih lokalnih poravnanja koja se traže u inicijalnom BLAST algoritmu. Visoko očuvane pozicije dobivaju visoke, a one slabije očuvane, niže ocjene. Profil izgrađen u prvoj iteraciji koristi se za drugu iteraciju i tako dalje, sve dok proces ne izvrši zadani broj iteracija ili ne konvergira. Iterativni postupak poboljšava rezultat i povećava osjetljivost. PSI-BLAST omogućava pronalazak udaljenijih odnosno manje sličnih sljedova. Profil izgrađen u prvoj iteraciji temelji se na sljedovima sličnim ulaznom te služi za sljedeću iteraciju u kojoj se pronalaze udaljeniji, a slični sljedovi. Time je PSI-BLAST osjetljiviji na evolucijski udaljenije sljedove proteina.

Algoritam se sastoji od sljedećih elemenata (Slika 3-2); korištenjem BLAST algoritma, u prvoj se iteraciji izgradi profil koji se zatim uspoređuje s bazom podataka proteina, odnosno njihovih sljedova. Početna točka u kreiranju profila jest grupa sljedova koji su poravnati, a ujedno su i izlazni podatak BLAST algoritma. Taj se rezultat reducira u cilju određivanja vrijednosti profila. Za svaki stupac poravnatih sljedova, u obzir se uzimaju i susjedni aminokiselinski ostaci. Tako se poravnati redci sljedova reduciraju, tj. uzimaju se samo oni redovi čiji su stupci postavljeni na način da svaki sadrži određeni ostatak ili prazninu, s time da su redovi iste duljine. Dobiveni profil slijeda (PSSM matrica) zatim se ponovo koristi kao upit za pretraživanje baze. PSI-BLAST procjenjuje statističku važnost pogodaka u bazi te se pretraživanje iterativno nastavlja, obično oko 5 puta. U svakom se koraku novi profil slijeda koristi kao upit.



Slika 3-2. Dijagram PSI-BLAST algoritma

U ovom radu odabrana metoda višestrukog poravnanja - PSI-BLAST (30) – profile slijeda nalazi iterativnim pretraživanjem NR proteinske baze podataka, u obliku PSSM matrice. Profili slijeda dobiveni su PSI-BLAST pretraživanjem NR baze proteinskih lanaca u 4 iteracije te su konstruirani za svaki od proteinskih lanaca na temelju njegove DSSP definicije i pohranjeni u obliku *.pssm* datoteke.

3.4 Priprema podataka

Atributi potrebni za klasifikaciju pročišćeni su iz izlaznih tekstualnih datoteka svakog proteinskog lanca i pripadnog dobivenog profila slijeda te su ujedinjeni tako da je svaki od skupova predstavljen ARFF datotekom. ARFF (*Atribute-Relation File Format*) je tekstualna (ASCII) datoteka koja opisuje skup primjera koji dijele zajednički skup svojstava. Svaka ARFF datoteka sastoji se od dva dijela: prvi dio čini zaglavlje (Slika 3-3), a drugi dio su informacije o podacima (Slika 3-4). Zaglavlje sadrži naziv relacije, te popis atributa i njihov opis.

```
@RELATION RS126
@ATTRIBUTE pdbID STRING
@ATTRIBUTE chainID STRING
@ATTRIBUTE RedBr NUMERIC
@ATTRIBUTE residue1 {A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,U,V,W,X,Y,Z}
@ATTRIBUTE residue2 {A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,U,V,W,X,Y,Z}
@ATTRIBUTE residue3 {A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,U,V,W,X,Y,Z}
.
.
.
@ATTRIBUTE profile1_A NUMERIC
@ATTRIBUTE profile1_R NUMERIC
@ATTRIBUTE profile1_N NUMERIC
.
.
```

Slika 3-3. Zaglavlje ARFF datoteke

Dio koji sadrži podatke je glavni dio datoteke i u njemu se nalaze vrijednosti koje opisuju atributi definirani u zaglavlju.

```
@RELATION RS126

@ATTRIBUTE pdbID STRING

@DATA

1A45,A,5,G,K,I,T,F,Y,E,D,R,16,0,0,0,0,0,0,84,0,0,0,0,0,0,0,0,0,0,0,0,0,17,0,0,0,10,0,3,0,0,0,64,0,0,0,5,0,0,0,0,5,0,0,0,0,0,0,0,0,55,7,0,6,
0,0,0,0,0,0,27,2,5,0,1,8,2,1,0,1,19,5,1,2,2,0,3,34,0,2,14,2,0,1,0,0,0,0,0,0,15,32,0,1,30,0,0,0,0,0,19,0,0,0,0,2,0,0,0,0,0,2,0,0,28,0,0,0,4,64,
0,4,0,1,11,0,2,65,1,0,0,0,2,1,0,0,7,4,0,0,3,1,5,5,28,1,9,12,7,10,0,5,11,2,1,0,1,0,0,0,1,7,18,1,8,2,2,22,2,0,1,0,10,2,0,16,4,4,0,0,2,E

1A45,A,6,K,I,T,F,Y,E,D,R,G,0,17,0,0,0,10,0,3,0,0,0,64,0,0,0,5,0,0,0,0,5,0,0,0,0,0,0,0,0,55,7,0,6,0,0,0,0,0,0,27,2,5,0,1,8,2,1,0,1,19,5,1,2,
2,0,3,34,0,2,14,2,0,1,0,0,0,0,0,0,0,15,32,0,1,30,0,0,0,0,0,0,19,0,0,0,0,2,0,0,0,0,0,2,0,0,28,0,0,0,4,64,0,4,0,1,11,0,2,65,1,0,0,0,2,1,0,0,7,4,0,0,
3,1,5,5,28,1,9,12,7,10,0,5,11,2,1,0,1,0,0,0,1,7,18,1,8,2,2,22,2,0,1,0,10,2,0,16,4,4,0,0,2,3,1,40,16,2,1,1,16,7,0,0,1,1,2,0,4,0,0,5,0,E

1A45,A,7,I,T,F,Y,E,D,R,G,F,5,0,0,0,0,0,0,0,0,55,7,0,6,0,0,0,0,0,0,27,2,5,0,1,8,2,1,0,1,19,5,1,2,2,0,3,34,0,2,14,2,0,1,0,0,0,0,0,0,15,32,0,
1,30,0,0,0,0,0,19,0,0,0,0,2,0,0,0,0,0,2,0,0,28,0,0,0,4,64,0,4,0,1,11,0,2,65,1,0,0,0,2,1,0,0,7,4,0,0,3,1,5,5,28,1,9,12,7,10,0,5,11,2,1,0,1,0,0,
0,1,7,18,1,8,2,2,22,2,0,1,0,10,2,0,16,4,4,0,0,2,3,1,40,16,2,1,1,16,7,0,0,1,1,2,0,4,0,0,5,0,0,0,0,0,2,2,0,0,1,0,2,0,1,77,0,0,0,1,12,2,E

...
```

Slika 3-4. Podaci u ARFF datoteci

3.5 Točnost predviđanja

Procjena metoda i algoritama predviđanja sekundarne strukture proteina bitan je zadatak jer pokazuje koliko je određena metoda uspješna i točna u zadatku predviđanja. Najvažnije u postupku evaluacije je provjera sposobnosti algoritma predviđanja da podjednako dobro radi na novom kao i na skupu podataka za treniranje. Postoji nekoliko mjera koje ocjenjuju učinkovitost, kvalitetu i stabilnost algoritma predviđanja. Kako bi se usporedila izvedba različitih tehnika predviđanja potrebno je definirati univerzalne mjere ili ocjene točnosti predviđanja. Evaluacija klasifikatora odnosi se na mjerenje njegove efikasnosti, tj. sposobnosti ispravnog razvrstanja što većeg broj uzoraka iz testnog skupa podataka. Neovisno o korištenom mjerilu učinkovitosti, treba uzeti u obzir činjenicu da su neki elementi sekundarne strukture (α-uzvojnice) „lakše“ predvidljivi od ostalih. To znači da skup podataka za testiranje jako utječe na dobivenu točnost. U praksi se skupovi podataka za testiranje biraju tako da su pojedini uzorci male međusobne sličnosti (skup nehomolognih proteina).

3.5.1 Q-score

Q-score je raširena i najčešće korištena mjera učinkovitosti. To je udio točno identificiranih (predviđenih) sekundarnih struktura i obično se izražava u postocima.

$$Q = \frac{broj\ točno\ klasificiranih\ sekundarnih\ struktura}{Ukupni\ broj\ aminokiselinskih\ rezidua} \times 100\% \qquad (3.1)$$

Obično se koristi indeks koji označava broj mogućih klasa sekundarne strukture u koje se pojedina rezidua može svrstati. Stoga, ako se koristi DSSP kod, ocjena se označava kao Q_8. Jednostavnija shema sa tri klase (H, E, C) koristi Q_3 oznaku. Ponekad se predviđanje ne vrši za svaku aminokiselinu iz slijeda pa je prikladnije koristiti modificiranu verziju – Q^*-score.

$$Q^* = \frac{broj\ točno\ klasificiranih\ sekundarnih\ struktura}{Ukupni\ broj\ predviđenih\ aminokiselinskih\ rezidua} \times 100\% \qquad (3.2)$$

Podjednako često se koristi i sljedeća definicija:

$$Q_x^{pre} = \frac{broj\ točno\ klasificiranih\ sekundarnih\ struktura\ u\ stanju\ x}{Ukupni\ broj\ aminokiselinskih\ rezidua\ u\ stanju\ x} \times 100\% \qquad (3.3)$$

Pri čemu *x* označava tip sekundarne strukture.

3.5.2 Matrica greške

Matrica greške (engl. *error matrix*) sadrži informaciju od djelovanju klasifikatora nad testnim skupom podataka, a uspoređuju se rezultati dobiveni razvrstavanjem uzoraka u odnosu na stvarne razrede kojima uzorci pripadaju. Matrica greške pokazuje broj stvarnih primjera određene klase naspram broja primjera koji su u tu klasu svrstani određenim modelom (predviđanje). S obzirom da se radi o tablici (dvije dimenzije, stvarna klasa primjera i procijenjena klasa), matrica greške ne samo da pokazuje koliko dobro model reproducira vrijednost klasa, nego daje i detaljan prikaz koji tipovi greške su najčešći. Podaci se prikazuju u matričnom obliku (Tablica 3-2).

Tablica 3-2 Matrica greške

		Stvarna klasa	
		+	**-**
Predviđena klasa	**+**	TP	FP
	-	FN	TN

U slučaju modela s dvije klase možemo definirati pozitivnu i negativnu klasu. Primjere koje je klasifikator označio kao pozitivne mogu biti stvarno pozitivni (engl. *true positives, TP*) i lažno pozitivni (engl. *false positives, FP*). Isto tako, uzorci podataka označeni kao negativni mogu biti stvarno negativni (engl. *true negatives, TN*) i lažno negativni (engl. *false negatives, FN*). Na temelju matrice greške definiraju se sljedeće mjere:

1. **Točnost** (engl. *accuracy*) odražava koliko je aminokiselinskih ostataka točno klasificirano, a definira se kao omjer ispravno klasificiranih primjera naspram ukupnog broja primjera

$$\frac{TP + TN}{TP + TN + FN + FP} \qquad (3.4)$$

2. **Odziv** (engl. *recall*) je mjera točno predviđenih pozitivnih uzoraka odnosno omjer ispravno klasificiranih pozitivnih primjera i ukupnog broja primjera

$$\frac{TP}{TP + FN} \tag{3.5}$$

3. **Preciznost** (engl. *precision*) se definira kao omjer ispravno klasificiranih pozitivnih primjera i primjera koje je klasifikator proglasio pozitivnima

$$\frac{TP}{TP + FP} \tag{3.6}$$

3.6 Implementacija

Proces implementacije zadatka može se svesti na korake opisane u nastavku. Kôd pomoćnih skripti korištenih u postupku priložen je na kraju rada.

1. **Dohvaćanje PDB zapisa**. Nakon što je konstruiran popis proteina iz pojedinih skupova, iz RCSB PDB (www.rcsb.org) baze podataka dohvaćeni su zapisi o proteinima u obliku tekstualnih *.pdb* datoteka. Svaki protein definiran je svojom datotekom čije se ime podudara sa imenom pripadnog proteina duljine 4 znakova.
2. **Pridruživanje sekundarne strukture**. Uz pomoć DSSP (*Dictionary of Secondary Structure in Proteins*) programa i skripte napisane u Perl programskom jeziku (*pdb2dssp.pl*) za svaku .pdb datoteku (odnosno odgovarajući protein) iz prethodnog koraka konstruirana je pripadajuća datoteka sa ekstenzijom *.dssp* u kojoj su između ostalog pohranjene informacije o sekundarnoj strukturi svakog aminokiselinskog ostatka proteina, dobivene proračunima na temelju strukturnih informacija iz PDB zapisa.
3. **Pročišćavanje podataka**. Za svaki protein, informacije sadržane u njegovim *.pdb* i *.dssp* datotekama pročišćeni su Perl skriptom (*parseDSSP.pl*) te uz pomoć BioPerl modula da bi se dobio tekstualni zapis o proteinu u kojem se nalaze informacije o poziciji aminokiselinskog ostatka unutar lanca, njegova PDB te DSSP oznaka ostatka (tip aminokiselinskog ostatka) i konačno, proračunat tip sekundarne strukture.

   Skupovi proteina su zapravo skupovi proteinskih lanaca jer se proteini mogu sastojati od više, ponekad istovrsnih, lanaca, pa kako bi se očuvala homolognost skupova određeni lanci se odbacuju. Proteinski lanci određeni su alfanumeričkim simbolima jedinične duljine te je na temelju PDB odnosno DSSP zapisa proteinskog kompleksa svaki lanac izdvojen u zasebnu datoteku, naziva koji se sastoji od imena proteina te oznake lanca. Npr. protein *1BBP* sastoji se od 4 lanca (*A, B, C* i *D*), no svi su istog primarnog slijeda pa se u razmatranje uzima samo jedan od njih (generira se datoteka *1BBP_A.txt*). Prikazani primjer (Slika 3-5) pokazuje slučaj kada neke od aminokiselinskih rezidua nisu definirane u DSSP izlazu. Tada se redni broj ostatka u lancu označava sa 'x' a aminokiselinski ostatak i njegova sekundarna struktura sa '?'. Drugi slučaj koji se može desiti je kada postoje prekidi u lancu; nepostojeće pozicije potrebno prikladno popuniti.

Slika 3-5 1AZU_A.txt

%Red. br.	RESSEQ	AA	SS
x	A	?	?
x	E	?	?
x	C	?	?
4	S	S	C
5	V	V	E
6	D	D	E
7	I	I	E
8	Q	Q	E
9	G	G	C

4. **Generiranje profila slijeda**. Korištenjem vlastite Perl skripte i PSI-BLAST programa za svaki od proteinskih lanaca generiran je profil slijeda u obliku PSSM matrice i pohranjen kao tekstualna datoteka nastavka *.pssm*. Slika 3-6 pokazuje organizaciju takve matrice; stupci sadržavaju vrijednosti koje označavaju vjerojatnost pojave jedne od 20 aminokiselina na određenoj poziciji unutar lanca. Za generiranje profila korištena je neredundantna NR baza proteinskih lanaca koja obuhvaća sljedove iz GenPept, SwissProt, PIR, PDF te PDB baza.
5. **Konstruiranje ARFF datoteka**. Konačna struktura - ARFF datoteka - predstavlja ulaz u program Rattle koji se koristi za klasifikaciju. U poglavlju 3.4 opisana je struktura i sadržaj ARFF zapisa. Tekstualni zapisi o lancima opisani u točci 3 te profili slijeda iz točke 4, pročišćavaju se skriptom napisanom u Perl programskom jeziku te se dobiveni podaci zapisuju u ARFF datoteku. ARFF datoteka pojedinog skupa podataka sadrži podatke za svaki od lanaca koji se nalazi u tom skupu. Princip izgradnje je taj da se odabere fiksna duljina prozora (koja mora biti neparan broj) koji se pomiče za jedno mjesto duž lanca počevši od početka, te se za svako mjesto gleda središnji aminokiselinski ostatak kojemu se za taj redak pridružuje ciljna klasa (sekundarna struktura). Redoslijed podataka u pojedinom retku je slijedeći: Oznaka proteina; oznaka lanca; redni broj aminokiselinskog ostatka u lancu; okvir od *n* aminokiselinskih ostataka; profil slijeda za svaki od ostataka; oznaka sekundarne strukture središnjeg ostatka u okviru. ARFF datoteke konstruiraju se zasebno za svaku metodu klasifikacije. Naime, inačica AdaBoost algoritma implementirana u Rattle omogućuje samo binarnu klasifikaciju a kako je zadatak razvrstavanje u tri klase, potrebno je prilagoditi strukturu klasa. Stoga se u slučaju AdaBoost metode klasifikacije za svaku od tri klase generira po jedna ARFF datoteka za pojedini skup podataka; jedna klasa se diskriminira tako da se ostale dvije tretiraju kao jedna. Pri tome je bitna pretpostavka da je točnost diskriminacije jedne klase od ostalih dviju jednakovrijedna razvrstavanju u tri klase. Na primjer, pokušaj predviđanja stanja *'H'* sekundarne strukture znači modifikaciju podataka tako da se svim aminokiselinskim ostacima koji pripadaju jednoj od preostale dvije klase (*'C'* ili *'E'*) dodijeli nova, zajednička klasa (npr. *'O'*).

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
F	0	0	0	0	0	0	0	0	0	6	6	0	0	88	0	0	0	0	0	0
V	17	0	0	2	0	0	0	0	0	3	0	0	0	5	3	0	2	0	0	68
N	0	0	72	0	0	0	0	3	0	0	0	4	0	0	0	17	4	0	0	0
Q	0	16	0	0	0	78	0	0	0	0	0	3	0	0	2	0	2	0	0	0
H	0	6	0	0	0	2	0	0	85	0	0	0	0	0	0	0	0	0	7	0
L	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
C	0	0	0	0	95	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
G	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
S	3	0	0	0	0	0	2	0	0	0	0	0	0	0	3	93	0	0	0	0
H	0	0	11	2	0	0	2	0	83	0	0	0	0	0	0	2	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
E	0	0	0	29	0	2	69	0	0	0	0	0	0	0	0	0	0	0	0	0
A	92	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0
L	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	5	0	5	0	0	0	0	5	0	0	0	0	85	0
L	0	0	0	0	0	0	4	0	0	0	75	0	6	3	0	7	0	0	0	5
V	5	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	89
C	0	0	0	0	95	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
G	2	5	0	0	0	4	0	87	0	0	0	3	0	0	0	0	0	0	0	0
E	0	0	3	33	0	0	53	0	0	0	0	0	0	0	7	3	0	0	0	0
R	0	66	2	5	0	0	0	0	0	0	0	22	0	0	0	3	2	0	0	0
G	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	2	0	0	2	0	0	94	0	2	0	0	0	0
F	0	0	0	0	0	0	0	0	0	4	0	0	0	89	0	0	0	0	7	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	7	0	0	86	0
T	2	2	14	0	0	3	2	0	0	11	2	0	7	0	1	9	43	0	2	2
P	0	0	3	3	0	0	0	0	0	0	0	0	0	0	92	0	2	0	0	0
K	0	8	3	2	0	0	0	0	0	0	1	76	2	0	0	3	4	0	0	0
A	31	0	0	0	0	0	0	13	0	0	0	0	0	0	0	25	29	0	0	3

Slika 3-6 Profil slijeda (PSSM matrica) za 2MHU_A

6. **Klasifikacija Rattle programom**. Nakon učitavanja ARFF zapisa u Rattle, moguće je odrediti značaj svake od varijabli (atributa); atribut može biti ulazni (engl. *input*) ukoliko predstavlja ulaznu, korisnu informaciju za klasifikator, ciljni (engl. *target*), ukoliko sadrži informaciju o ciljnoj klasi primjera, ili ga može označiti kao nebitnog (engl. *ignore*), ukoliko ne nosi informaciju bitnu za klasifikaciju pa ga se zanemaruje. U konkretnom slučaju, ulazni atributi su *n* aminokiselinskih ostataka unutar prozora kao i pripadni profili slijeda (*residue1, …, residueN, profile1_A, …, …profileN_V*), element sekundarne strukture (*Class*) je

ciljni atribut, dok se ime proteina (*pdbID*), oznaka lanca (*chainID*) i redni broj središnjeg ostatka unutar okvira (*RedBr*) zanemaruju.

Najvažniji parametri AdaBoost metode su broj stabala te maksimalna dubina pojedinog stabla. Broj stabala postavljen je na 50 a maksimalna dubina iznosi 30.

Metodu slučajnih šuma karakteriziraju broj stabala u ansamblu te broj atributa koji se po principu slučajnog odabira potprostora uzimaju za pojedino stablo. Pri učenju podacima iz RS126 skupa korišteno je 500 stabala za generiranje slučajne šume, a zbog računalne zahtjevnosti pri većim skupovima je korišten manji broj stabala; 200 za CB513, odnosno 100 stabala za PSIPRED skup. Broj atributa za grananje za sve skupove iznosi 13.

4 REZULTATI

4.1 Metoda slučajne šume

Evaluacija modela se nakon treniranja ansambla vrši skupom podataka za testiranje te se rezultati prikazuju u obliku matrice greške (engl. *error matrix*). Radi usporedbe sa AdaBoost metodom koja klasificira diskriminacijom u dvije klase te sa slučajem kada se vrši podjela u tri klase, klasifikacija se nad svakim skupom vrši na dva načina: klasifikacijom u tri klase te diskriminacijom svake od klasa naspram ostale dvije. Klasifikacija u tri klase metodom slučajne šume provedena Rattle programom daje sljedeće rezultate za svaki od testnih skupova – RS126 (Tablica 4-1), CB513 (Tablica 4-2) te PSIPRED (Tablica 4-3).

Tablica 4-1 Matrica greške za RS126 skup

9 rezidua				11 rezidua			
Stvarno / Predviđeno	C	E	H	Stvarno / Predviđeno	C	E	H
C	39	12	10	C	39	13	10
E	2	10	1	E	1	10	1
H	3	2	21	H	3	2	22

Tablica 4-2 Matrica greške za CB513 skup

9 rezidua				11 rezidua			
Stvarno / Predviđeno	C	E	H	Stvarno / Predviđeno	C	E	H
C	35	9	8	C	35	9	8
E	2	11	1	E	2	11	1
H	4	3	27	H	4	3	27

Tablica 4-3 Matrica greške za PSIPRED skup

9 rezidua				11 rezidua			
Stvarno / Predviđeno	C	E	H	Stvarno / Predviđeno	C	E	H
C	37	7	6	C	38	7	6
E	2	17	1	E	2	17	1
H	3	1	26	H	2	1	26

Iz prikazanih matrica greške može se odrediti točnost klasifikacije tako da se odredi omjer točno predviđenih (elementi na dijagonali matrice) i ukupnog broja primjera. Tablica 4-4 prikazuje točnost odnosno Q vrijednost točnosti klasifikacije u 3 klase za svaki od skupova postignutu metodom slučajne šume.

Tablica 4-4 Točnost razvrstavanja u 3 klase postignuta metodom slučajne šume

Q (%)					
9 rezidua			11 rezidua		
RS126	CB513	PSIPRED	RS126	CB513	PSIPRED
70.0	**73.0**	**80.0**	**70.3**	**73.0**	**81.0**

Binarnom klasifikacijom metodom slučajne šume dobiveni su sljedeće matrice greške za RS126 (Tablica 4-5), CB513 (Tablica 4-6), odnosno PSIPRED (Tablica 4-7) skup.

Tablica 4-5 Matrice greške za RS126 skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	11	1	E	5	1	C	22	5
O	21	67	O	18	77	O	22	51
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	10	0	E	2	0	C	21	5
O	22	67	O	22	76	O	22	52

Tablica 4-6 Matrice greške za CB513 skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	17	1	E	5	1	C	23	6
O	19	63	O	18	77	O	18	53
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	18	1	E	4	0	C	22	4
O	17	64	O	18	77	O	19	55

Tablica 4-7 Matrica greške za PSIPRED skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	20	1	E	11	1	C	29	5
O	14	66	O	13	75	O	13	53
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	19	1	E	11	1	C	29	5
O	14	66	O	14	75	O	13	53

Tablica 4-8 prikazuje ukupnu Q_3 točnost binarne klasifikacije postignutu metodom slučajne šume, proračunatu na temelju navedenih matrica greške.

Tablica 4-8 Točnost binarne klasifikacije metodom slučajne šume

RF (%)	**RS126**		**CB513**		**PSIPRED**	
Duljina okvira (br. rezidua)	9	11	9	11	9	11
H	78.0	77.8	80.0	82.0	86.0	85.0
E	81.2	78.8	81.2	81.2	86.0	85.2
C	73.0	73.0	76.0	77.0	82.0	82.0
Q_3	**77.4**	**76.5**	**80.0**	**80.1**	**84.6**	**84.1**

4.2 AdaBoost

AdaBoost inačica algoritma implementirana u korišteni alat, Rattle, vrši binarnu klasifikaciju, odnosno diskriminaciju jedne klase naspram ostale dvije. Prikazani su rezultati klasifikacije za svaki od skupova podataka (RS126 - Tablica 4-9, CB513 - Tablica 4-10, PSIPRED - Tablica 4-11) u obliku matrica greške.

Tablica 4-9 Matrica greške za RS126 skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	15	4	E	6	2	C	28	10
O	17	64	O	18	75	O	15	46
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	15	3	E	6	1	C	29	11
O	17	64	O	19	74	O	15	45

Tablica 4-10 Matrica greške za CB513 skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	19	4	E	5	1	C	27	10
O	17	60	O	18	76	O	15	49
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	19	4	E	5	1	C	26	9
O	17	60	O	17	76	O	15	49

Tablica 4-11 Matrica greške za PSIPRED skup (binarna klasifikacija)

9 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	15	3	E	6	2	C	27	10
O	18	64	O	19	73	O	15	48
11 rezidua								
H			**E**			**C**		
Stvarno / Predviđeno	H	O	Stvarno / Predviđeno	E	O	Stvarno / Predviđeno	C	O
H	15	3	E	6	2	C	27	10
O	18	64	O	19	74	O	15	48

Odgovarajuća Q_3 točnost AdaBoost metode za RS126, CB513 i PSIPRED skupove je sljedeća (Tablica 4-12):

Tablica 4-12 Točnost binarne klasifikacije postignuta AdaBoost metodom

AdaBoost (%)	**RS126**		**CB513**		**PSIPRED**	
Duljina okvira (br. rezidua)	9	11	9	11	9	11
H	79.0	79.8	79.0	79.0	79.0	79.0
E	79.2	80.0	81.0	82.0	79.0	79.2
C	74.8	74.0	75.3	75.8	75.0	75.0
Q_3	**77.9**	**77.9**	**78.4**	**78.9**	**77.7**	**77.7**

4.3 Usporedba sa dosadašnjim rezultatima

4.3.1 Pregled metoda predviđanja sekundarne strukture

Rani pokušaji predviđanja sekundarne strukture začeti još 60-tih godina prošlog stoljeća bili su ograničeni malim brojem dostupnih struktura i nedovoljnim računalnim resursima. Prvi empirički sustavi previđanja na temelju aminokiselinskog slijeda bazirali su se na statistici struktura riješenih rendgenskom kristalografijom. Fasman i Chou autori su prve takve metode. Ta jednostavna metoda temelji se na analizi relativne učestalosti pojavljivanja pojedinih rezidua u specifičnim sekundarnim strukturama (uzvojnice, ploče i omče). Poboljšanje Chou-Fasman (31) metode ostvareno je 1978. godine razvojem GOR (32) algoritma (prema imenima autora – Garnier, Osuguthope, i Robson) koji se temeljio na teoriji informacija i Bayesovoj statistici, a uveo je i utjecaj susjedstva rezidua kao dodatni faktor pri predviđanju. Chou-Fasman, GOR te Lim metoda (33) predstavnici su

prve generacije metoda predviđanja sekundarne strukture. U svojim početnim inačicama te metode su bile ograničene točnosti od tek 50-60% (po Q ocjeni).

Tijekom vremena broj otkrivenih proteinskih struktura ubrzano je rastao, što je omogućilo raspoznavanje evolucijskih informacija u bazama podataka. Drugu generaciju metoda karakterizira uvođenje višestrukog poravnanja sljedova (engl. – *multiple sequence alignment*, MSA) za dobivanje informacija o homolognosti, evolucijskoj i strukturnoj sličnosti, koje se koristi u sprezi sa tehnikama strojnog učenja (ANN – *artificial neural network*, SVM – *support vector machine*, NN – nearest neighbor), naprednim statističkim metodama (HMM – *hidden Markov model*), teorijom grafova, itd. (34). Višestruko poravnanje slijeda temelji se na identifikaciji homologa pretraživanjem proteinskih baza podataka. Nalaženje višestrukog poravnanja iterativnim pretraživanjem baza uobičajen je dio današnjih metoda predviđanja. Uzorci varijabilnosti sljedova uočeni na temelju poravnanja obično pružaju informaciju o očuvanosti elemenata jezgre (hidrofobna jezgra i dijelovi bitni za funkciju proteina), dok položaji lokalnih brisanja i umetanja dijelova slijeda nagoviještaju površinski izložene petlje. Treći element koji je omogućio nagli napredak predviđanja sekundarne strukture je nagli porast broja poznatih proteinskih sljedova i struktura te sve veća osjetljivost alata za automatsko pretraživanje baza. To je omogućilo točnu identifikaciju divergentnijih homologa pa stoga i stvaranje profila većih strukturnih familija koje utjelovljuju evolucijske informacije. Povećanje dostupne količine informacija također je značilo veće skupove podataka, što je algoritmima strojnog učenja omogućilo bolju točnost i osjetljivost. Složeniji pristup problemu rezultirao je boljom točnošću predviđanja ovih metoda (poput PHD, SAM-T98 i PSIPRED) koja je uglavnom iznad 70%.

Trenutno, preporučen pristup predviđanju sekundarne strukture podrazumijeva kombiniranje (konsenzus) rezultata različitih metoda (35); ono može uključivati napredne pristupe strojnog učenja, poput glasanja, linearne diskriminacije, neuronskih mreža i stabala odluka. Takva ideja prvi put je implementirana u JPRED metodi (36), meta-serveru koji radi konsenzus različitih strategija predviđanja, implementiranih u PHD (17), NNSSP (37), DSC (38) i PREDATOR (39). Novija klasa meta-pristupa, kao što je npr. PROTEUS (40), kombinira uobičajeno predviđanje homolognih segmenata slijeda i *de novo* predviđanje dijelova slijeda koji odudaraju od predloška dobivenog višestrukim poravnanjem. Takav pristup rezultira još većim točnostima predviđanja koje prelaze 80%.

Dok su rane metode pružale točnost od 50-60%, uz gotovo zanemarivu točnost predviđanja β-ploča, najbolje moderne metode dosižu točnost od više od 80% po rezidui, odnosno oko ~10% manje za aminokiselinske ostatke β-ploča (41). Razlika između teoretske granice i trenutne točnosti predviđanja, kao i razlika u točnosti predviđanja pojedinih struktura, posljedica su uglavnom teško uočljivih interakcija dugog dometa koje mogu utjecati na formaciju sekundarne strukture, ali i inherentnih nesavršenosti tehnika za eksperimentalno određivanje struktura (rendgenska kristalografija i NMR). Pokazano je da jedan te isti aminokiselinski slijed može oblikovati α-uzvojnicu kada se nalazi na određenom položaju unutar primarnog slijeda, ali i β-ploču kada se nađe u drukčijem kontekstu slijeda. Osim toga, tijekom procesa smotavanja, određeni dio proteina može u prvom trenutku poprimiti

sekundarnu strukturu pogodnu lokalnom slijedu i kasnije se zbog nelokalnih interakcija preoblikovati u drugu sekundarnu strukturu. 3D-JUFO (42) uzima u obzir posljednju činjenicu i kombinira iterativno *de novo* predviđanje pristupom sličnim onome u PSIPRED metodi, sa predviđanjem tercijarne strukture ROSETTA (43) metodom, nakon čega slijedi re-predikcija elemenata sekundarne strukture na temelju lokalnog okoliša rezidua vidljivih u modelu tercijarne strukture. 3D-JUFO ostvaruje točnosti od 80%, sa značajnim poboljšanjem predviđanja β-vrpci (76%) u odnosu na čisto sekvencijalne metode.

Značajnije nove metode približavaju točnost predviđanja njenoj teoretskoj granici komponirajući bioinformatičku metodologiju sa eksperimentalnim tehnikama cirkularnog dihronizma i FTIR spektroskopije (44) za procjenu cjelokupnog sadržaja sekundarne strukture. Sličan pristup združivanja na profilu zasnivanih metoda te *de novo* predviđanja sekundarne strukture i izloženosti otopini implementiran je u DISTILL paketu (45).

Tablica 4-13 prikazuje sažetak dosadašnjih relevantnijih metoda predviđanja sekundarne strukture; korištene skupove podataka i način (metodu) implementacije, način predviđanja sekundarne strukture, te postignutu točnost.

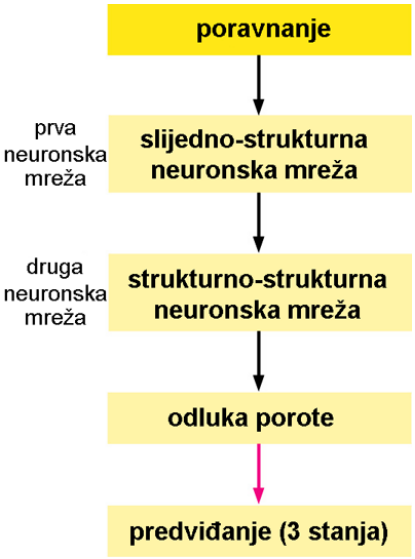
Tablica 4-13. Prikaz dosadašnjih metoda predviđanja sekundarne strukture

Autor	Metoda	Skup podataka	Pridruživanje sekundarne strukture	Metoda predviđanja	Provjera modela	Rezultati
Chou PY, Fasman GD	**Chou-Fasman** (31)	29 riješenih proteinskih lanaca	DSSP	Statistika + heuristička pravila	-	46.88% (Q_3)
Garnier J, Gibrat JF, Robson B.	**GOR** (32) (46)	CB513	DSSP	Teorija informacije + Bayesova statistika	Jackknife	GOR - 64.4% (Q_3) GOR V - 73.4% (Q_3)
Burkhart R, Chris S	**PHD** (17)	RS126	DSSP	Neuronske mreže + višestruka poravnanja	7-struka unakrsna validacija	70.8% (Q_3)
Frishman D, Argos P	**PREDATOR** (39)	RS125	DSSP	Metoda najbližeg susjeda	Jackknife	68% (Q_3 pojedina sekvenca) 75% (Q_3 skup srodnih sekvenci)
Cuff JA., Barton GJ	**JNet** (47)	CB513 (480 proteinskih lanaca)	DSSP	Neuronske mreže + višestruka poravnanja	7-struka unakrsna validacija	76.4% (Q_3)
Jones DT.	**PSIPRED** (19)	Treniranje: 2235 proteina 187 proteina (CATH T-razina)	DSSP	Neuronske mreže + PSI-BLAST profil (PSSM)	Unakrsna validacija	78% (Q_3)
Pollastri G, Przybylski D, Rost B, Baldi P	**SSPro** (48) (49)	TRAIN (1180 sljedova) R126, EVA (common3) i CASP4	DSSP	11 BRNN modela	7-struka unakrsna validacija	SSPro 2.0 – 77.67-80.65% (Q_3) SSPro8 2.0 – 62-63% (Q_8)
Ouali M, Kong RD.	**PROF** (50)	CB513 (496 proteinskih lanaca)	DSSP	Višestruki kaskadni klasifikatori u sprezi sa linearnom diskriminacijom	Pojedinačna unakrsna validacija (engl. *leave-one-out cross validation*)	76.7% (Q_3)
Pollastri G, McLysaght A	**PORTER** (45) (51)	25%-pdb_select, NCBI NR baza (3.3.2004)	DSSP	5 dvorazinskih BRNN modela	5-struka unakrsna validacija	Q_3=79.01%, SOV=75.0% (NR) Q_3=76.8%, SOV=72.0% (EVA common2)
Montgomerie S, Sundararaj S, Gallin WJ, Wishart DS.	**PROTEUS** (40)	PROTEUS-2D, EVA (ožujak 2006.)	VADAR	Strukturno poravnanje sekvenci, konsenzus neuronskih mreža	Pojedinačna unakrsna validacija, Testiranje „na neviđeno“ (engl. blind testing)	Q_3 = 88.0% SOV = 86.5%
Lin K, Simossis VA., Taylor WR., Heringa J	**YASPIN** (52)	PDB25 SCOP1.65	DSSP	Hidden Neural Networks (HNN=HMM+NN)	6-struka unakrsna validacija	Q_3 = 66.41% (PDB25), 77.06% (EVA common5) SOV = 62.15% (PDB25), 73.88% (EVA common5)
Ofer D, Yaoqi Z	**SPINE** (53)	PISCES (<25% sličnosti, <3 Å) – 2640 proteina, Carugo-338, CB-513	DSSP	Konsenzus neuronskih mreža	10-struka unakrsna validacija	Q_3 = 79.5% (PISCES, 2640 proteina) Q_3 = 77.07% (Carugo-338) Q_3 = 76.77% (CB-513)

4.3.2 RS126

RS126 je skup proteinskih lanaca koji su konstruirali Rost i Sander za potrebe razvoja PHD (17) metode predviđanja sekundarne strukture. Skup čine nehomologni lanci duljine najmanje 80 rezidua odabrani tako da ne dijele više od 25% sličnosti slijeda sa bilo kojim drugim pripadnikom skupa; sastoji se od 130 sljedova od čega su 4 lanci transmembranskih proteina te se ne uzimaju u obzir.

PHD je prva metoda koja je probila granicu točnosti predviđanja sekundarne strukture od 70%, a temelji se na neuronskim mrežama i višestrukom poravnanju sljedova. PHD algoritam (Slika 4-1) koristi informacije poravnanja homolognih sljedova, lokalne statističke informacije i strukturne parametre enkodirane u dvije neuronske mreže. Za svaki od lanaca opisanog nehomolognog skupa iz HSSP baze se dohvaćaju višestruka poravnanja i pripadajuća pridruženja sekundarne strukture. Profili slijeda dobiveni su MAXHOM programom, a redukcijska „8/3“ shema pridruživanja istovjetna je korištenoj u ovom radu (Tablica 3-1) što omogućuje vjerodostojnu usporedbu rezultata.



Slika 4-1 PHD algoritam

Tablica 4-14 daje usporedni prikaz točnosti postignutih u ovom radu i PHD metodi. U potonjem slučaju točnost je procijenjena sedmerostrukom unakrsnom validacijom. Bez obzira na vrstu ansambl klasifikatora, uočljiva je nadmoćnost binarne klasifikacije nad PHD metodom do čak 10%, dok je razvrstavanje u tri klase metodom slučajne šume dalo razočaravajuće rezultate, nešto čak i niže od onih postignutih PHD metodom.

Tablica 4-14 Usporedba rezultata sa PHD metodom

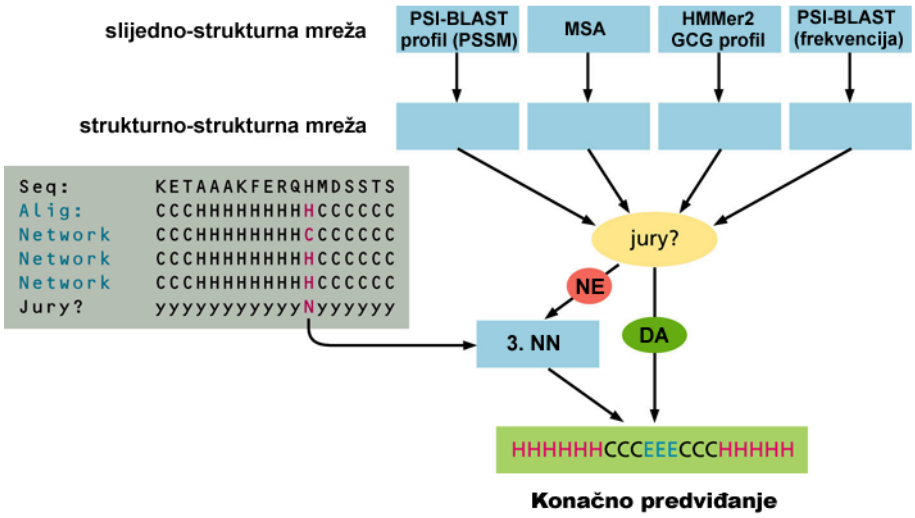
	PHD	AdaBoost		RF (3 klase)		RF (2 klase)	
Duljina okvira	13	9	11	9	11	9	11
Q_3 (%)	**70.8**	**77.9**	**77.9**	**70.0**	**70.3**	**77.4**	**80.0**

4.3.3 CB513

Rezultati postignute korištenjem CB513 skupa proteinskih lanaca uspoređeni su sa tri poznate metode za predviđanje sekundarne strukture: GOR, JNet i PROF.

GOR (32) metoda koristi teoriju informacija i Bayesovu statistiku za predviđanje sekundarne strukture proteina, a za razliku od prijašnjih metoda u obzir uzima interakcije kratkog dometa i utjecaj susjednih rezidua ne sekundarnu strukturu koju poprima središnji aminokiselinski ostatak. Izvorna inačica algoritma je tijekom godina unaprijeđena uključivanjem većih skupova podataka i detaljnijom statistikom, a trenutna inačica , GOR V (46), iskorištava i evolucijske informacije u obliku profila slijeda dobivenih PSI-BLAST algoritmom. Točnost predviđanja GOR V metode procijenjene *jackknife* tehnikom iznosi 73.4% (Q_3).

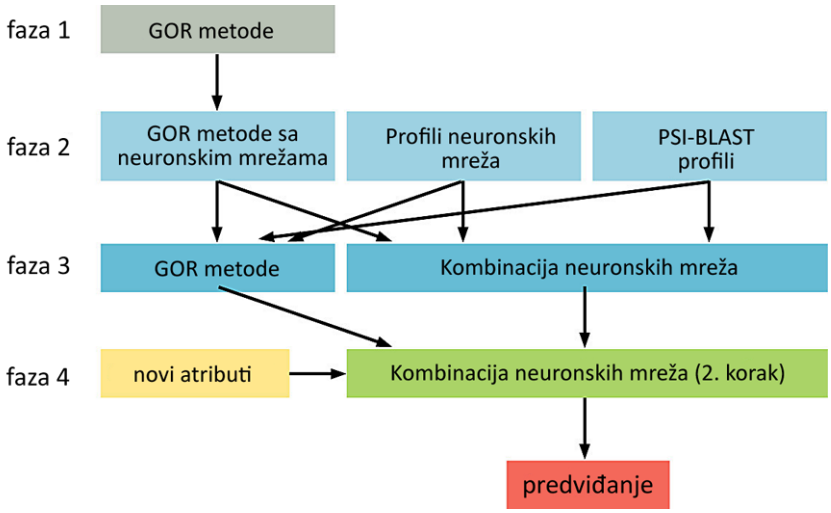
JNet algoritam (47) koristi istu strukturu neuronske mreže kao i PHD metoda. Razlika algoritma je u korištenju proširenog skupa proteinskih lanaca, drukčijoj „8/3“ redukcijskoj shemi i novim načinima dobivanja višestrukih poravnanja sljedova. JNet se sastoji od 4 skupa slijedno-strukturnih i strukturno-strukturnih neuronskih mreža (Slika 4-2). Ulaz svake od 4 slijedno-strukturnih mreža je jedna od 4 alternativne reprezentacije skupa poravnatih sljedova. Za pozicije za koje ne postoji jednoglasno slaganje u predviđanju, koristi se treća mreža za određivanje konačnog predviđanja. Točnost metode postignuta 7-strukom unakrsnom validacijom iznosi 76.4% (Q_3).



Slika 4-2 Pojednostavljen prikaz JNet metode

PROF (*PROtein Forecasting*) (50) se temelji na kaskadi različitih načina predviđanja ili algoritama poravnanja unutar jednog programa, uz korištenje neuronskih mreža i linearne diskriminacije za izbor konačnog predviđanja (Slika 4-3). Za ostvarivanje različitih klasifikatora korištene su na GOR formalizmu temeljene metode potpomognute linearnom i kvadratnom diskriminacijom, višestrukim poravnanjem sljedova, te metodom neuronskih mreža. Teoretska podloga PROF algoritma dolazi iz osnovne teorije vjerojatnosti: svaki

podatak relevantan za predviđanje trebao bi se iskoristiti za ostvarivanje tog predviđanja. To znači da bi uvijek trebalo biti moguće poboljšati predviđanje združivanjem različitih algoritama ili jednim algoritmom uz različite načine treniranja i/ili uz različite skupove podataka, dok god klasifikatori nisu skloni istom tipu pogreške (klasifikatori proizvode nekorelirane pogreške). Korištenjem pojedinačne unakrsne validacije (engl. „*leave-one-out*“) postignuta je točnost od 76.7% po Q_3 ocjeni.



Slika 4-3 Arhitektura PROF kaskadnog klasifikatora

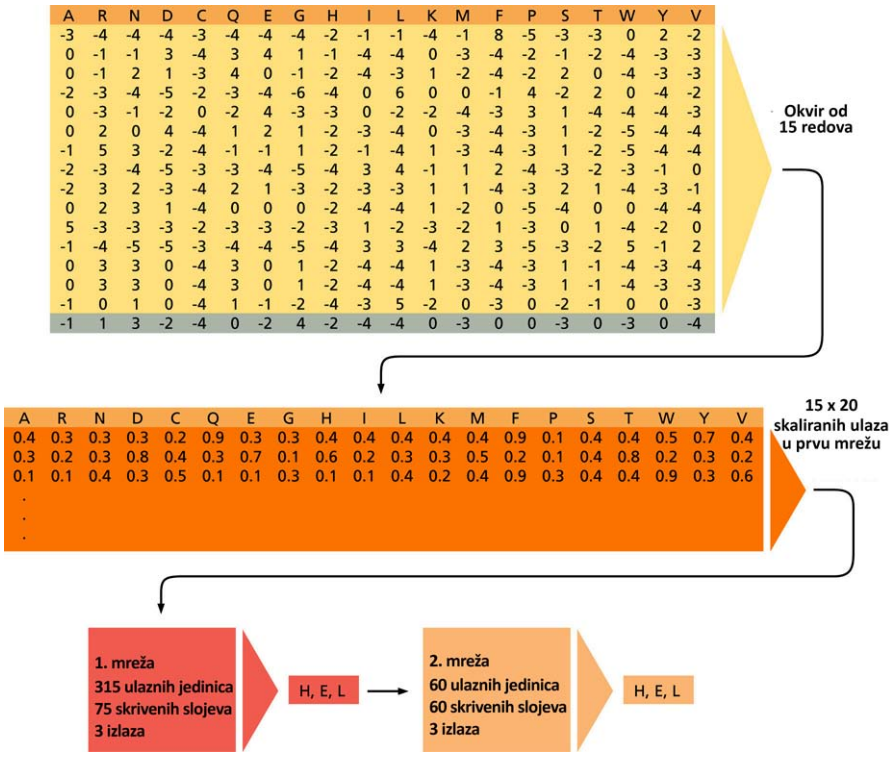
Iako, nažalost, svaka od triju opisanih metoda koristi drukčiju shemu redukcije 8-slovne abecede sekundarne strukture, pretpostavljen je zanemariv utjecaj tog faktora te je u nastavku dan usporedni prikaz rezultata navedenih metoda kao i onog postignutog u ovom radu (Tablica 4-15);

Tablica 4-15 Usporedba rezultata za CB513 skup

	GOR V	JNet	PROF	AdaBoost		RF (3 klase)		RF (2 klase)	
Duljina okvira	18	17	13	9	11	9	11	9	11
Q_3 (%)	**73.4**	**76.4**	**76.7**	**78.4**	**78.9**	**73.0**	**73.0**	**80.0**	**80.1**

4.3.4 PSIPRED

PSIPRED (19) je jednostavna metoda predviđanja sekundarne strukture temeljena na dvije unaprijedne neuronske mreže (engl. *feedforward neural network*) koje vrše analizu izlaza dobivenih iz PSI-BLAST programa. Metoda predviđanja podijeljena je u tri faze (Slika 4-4): generiranje profila slijeda, inicijalno predviđanje sekundarne strukture, te konačno filtriranje predviđene strukture.



Slika 4-4 Shema PSIPRED metode

Oko 340,000 sljedova izdvojenih iz više neredundantnih, javnih baza reducirano je filtriranjem područja malog sadržaja informacija (filtriranje SEG programom) te uklanjanjem transmembranskih i pletenastih struktura (engl. *coiled-coil structures*). PSSM matrica dobivena PSI-BLAST-om u tri iteracije proslijeđena je na ulaz neuronske mreže. Ta matrica sastoji se od $20 \cdot M$ elemenata, pri čemu je M duljina ciljnog slijeda, a svaki od elemenata predstavlja vjerojatnost supstitucije tog aminokiselinskog ostatka na toj poziciji unutar predloška (na temelju prosjeka ocjena BLOSUM62 matrice za danu poziciju poravnanja). Elementi matrice profila su skalirani na interval [0,1] sigmoidnom funkcijom($\frac{1}{1+e^{-x}}$). PSIPRED se sastoji od dvije slijedne neuronske mreže temeljene na uobičajenoj unaprijednoj *back-propagation* arhitekturi sa jednim skrivenim slojem. Za prvu mrežu (slijedno-strukturna mreža) korišten je ulazni okvir od 15 rezidua, pa uz dodatni parametar za oznaku prelaska okvira van granica proteinskog lanca, mreža ima ukupno 315 ulaznih jedinica, podijeljenih u 15 grupa od 21 jedinice. Skriveni sloj sadrži 75 čvorova, nakon kojega slijedi izlazni sloj sa tri izlazna čvora za svako od mogućih stanja sekundarne strukture. Druga neuronska mreža za filtriranje (strukturno-strukturna mreža) također se temelji na okviru od 15 uzoraka i ima 60 ulaznih jedinica podijeljenih u 15 skupina od 4 (3 stanja+dodatni parametar) jedinice. Skriveni sloj druge mreže čini 60 čvorova. Testni skup temelji se na CATH T-razini iz koje su izdvojene strukture razlučivosti <1.8 Å i sastoji se od 187 proteinskih lanaca. Referentna stanja sekundarnih struktura za svaku od struktura iz skupa za testiranje i skupa za treniranje izvedena su iz DSSP definicija i reducirana u tri stanja prema Rost i Sander shemi (Tablica 3-1). Autor metode Jones utvrdio je točnost metode od 76.5% po Q_3 ocjeni za izdvojeni skup od 187 proteina, procijenjenu trostrukom unakrsnom validacijom. Trenutna inačica ove metode (PSIPRED v2.0) još povećava

točnost predviđanja uvođenjem konsenzusa predviđanja četiri nezavisno treniranih parova neuronskih mreža. Korištenjem vrlo strogog postupka unakrsne validacije za procjenu učinkovitosti metode, PSIPRED 2.0 postiže prosječne Q_3 točnosti od 77.62% (EVAc4 (54)), 81.01% (RS126), odnosno 79.95% (CB513). Potonje točnosti za navedene skupove nisu rezultati izvornog rada (19), već su dobiveni testiranjem PSIPRED poslužitelja za predviđanje sekundarne strukture (http://bioinf4.cs.ucl.ac.uk:3000/psipred).

PSIPRED metoda je svojim pojavljivanjem 1999. godine napravila veliki pomak u točnosti predviđanja sekundarne strukture te je stalnim unapređivanjem tijekom godina i do danas ostala standard. Upravo se zbog svoje dugovječnosti i provjerenosti često uzima kao etalon za usporedbu metoda predviđanja sekundarne strukture proteina.

U nastavku je prikazana usporedba točnosti PSIPRED metode i rezultata dobivenih u ovom radu korištenjem slučajnih šuma i AdaBoost algoritma (Tablica 4-16). Rezultati PSIPRED metode dobiveni su vrlo strogim postupkom trostruke unakrsne validacije nad pažljivo odabranim podskupom PSIPRED skupa podataka od 187 proteinskih lanaca.

Tablica 4-16 Usporedba sa rezultatima PSIPRED metode

	PSIPRED	AdaBoost		RF (3 klase)		RF (2 klase)	
Duljina okvira	15	9	11	9	11	9	11
Q_3 (%)	**76.5**	**77.7**	**77.7**	**80.0**	**81.0**	**84.6**	**84.1**

Vidi se da je točnost postignuta u ovom radu redovito veća od one dobivene izvornom PSIPRED metodom, bez obzira na korištenu metodu klasifikacije. AdaBoost sustav klasifikatora postiže najlošiju točnost usporedivu onoj PSIPRED metode, dok binarna klasifikacija metodom slučajne šume najbolje obavlja zadaću predviđanja sa do 8% većom točnošću.

5 ZAKLJUČAK

Dobiveni rezultati potvrđuju da se uz relativno mali broj svojstava aminokiselinskih ostataka može dobiti relativno dobra točnost koja je u rangu točnosti današnjih metoda predviđanja sekundarne strukture. Uspoređujući dvije metode korištene u radu može se uočiti monotonost rezultata postignutih AdaBoost metodom bez obzira na duljinu okvira ili količinu (veličinu skupa) primijenjenih podataka s jedne strane, te razmjernu ovisnost točnosti metode slučajnih šuma o količini dostupnih podataka. Najbolji rezultati postignuti su binarnom klasifikacijom metodom slučajnih šuma, koja se pokazala značajno točnijom za sva tri skupa podataka, što je u suprotnosti sa očekivanom nadmoćnošću klasifikacije algoritma u tri razreda. Usporedba dobivenih rezultata sa dosadašnjim naročito iznenađuje zbog duljine korištenih okvira koja je redovito veća u svim ostalim metodama.

Doseg i raspon varijabli bitnih za klasifikaciju korištenih u ovom radu relativno je ograničen, pa su u cilju kvalitetnijeg ispitivanja točnosti korištenih algoritama moguća poboljšanja. Prije svega, dostupni računalni resursi ograničili su kompleksnost modela; iako naočigled ne utječe na ishod, veličina okvira jedan je od parametara čije bi povećanje moglo utjecati na točnost. Osim toga, tu su i parametri samih algoritama klasifikacije – broj stabala i veličina slučajnog potprostora atributa kod slučajnih šuma odnosno broj i maksimalna dubina stabala u AdaBoost metodi, čije variranje, optimizacija i eventualno povećanje možda dovodi do boljih rezultata. Dobiveni rezultati upućuju na to da parametri modela imaju veći utjecaj na točnost od duljine okvira. Konačno, odabrani skup atributa koji opisuje podatke čini samo slijed aminokiselinskih ostataka i odgovarajući profili slijeda pa uvođenje dodatnih, odgovarajućih atributa gotovo sigurno vodi poboljšanju rezultata predviđanja.

U nastavku je prikazan općenit pogled na današnje stanje u području predviđanja sekundarne strukture proteina te neki pokazatelji u kojem smjeru se odgovarajuće metode kreću.

Predviđanje 3D strukture proteina na temelju jednodimenzionalnog slijeda aminokiselina jedan je od glavnih problema molekularne biologije. Uobičajene laboratorijske metode rješavanja tog problema su prespore da premoste jaz između brzo rastućeg broja sljedova i njihovih predviđenih struktura. Uspješno predviđanje sekundarne strukture pravi je smjer za približavanje 3D strukturi i rješenju problema smatanja proteina. Različiti pristupi predviđanju uzimaju u obzir različita kemijska i fizikalna svojstva. To je dovelo do nastanka brojnih alata, metoda i tehnika, od kojih su neki specijalizirani za rad ili na određenim vidovima ili određenim kategorijama proteina. Ne postoji jedinstvena metoda dovoljno točna ili pouzdana za predviđanje svih vrsta proteinskih struktura. Jedan od načina unapređenja kvalitete modela je zajednička primjena različitih tehnika predviđanja kao i kombiniranje njihovih rezultata. Tome se mora pristupiti pažljivo jer različite tehnike imaju različite formate predstavljana svoga ulaza i izlaza. Kako svaka od njih koristi drukčiji pristup, usporedba i interpretacija rezultata može biti težak zadatak. Ponekad integracija i ne daje najbolje rezultate. Ali, općenito, dobra detekcija homolognih sljedova i dobro višestruko poravnanje može poboljšati rezultate. Unapređenja u predviđanju

sekundarnih struktura u budućnosti se može očekivati sa više strana. Kao prvo, najveći napredak donijet će porast broja eksperimentalno utvrđenih struktura. Razumljivo je da će povećanjem broja „riješenih“ struktura sve više ciljnih sljedova imati kompatibilne strukture unutar strukturnih baza podataka. To ne samo da će povećati šanse da modeliranje po sličnosti točno pridruži strukturu nego i vjerojatnost da je prepoznata struktura sličnija ciljnoj, što potencijalno povećava točnost modela. Drugo, naredna poboljšanja slijedno-strukturnih poravnanja također mogu poboljšati točnost modela. Trenutne metode usporedbe sljedova mogu poravnati tek dio aminokiselinskih ostataka koji se poravnavaju u strukturnom poravnanju. Bolje poravnati ostaci bez sumnje mogu poboljšati točnost modela. Konačno, oplemenjivanje modela dobivenih modeliranjem po sličnosti *threading* pa čak i *ab initio* metodama može se izvršiti molekularnom dinamikom s točnim atomskim fizičkim potencijalima.

Trenutna točnost predviđanja sekundarnih struktura još uvijek je oko 5-10% ispod svoje teoretske granice koja iznosi ~90%. Ta gornja granica točnosti predviđanja procijenjena je na temelju dva zapažanja: prvo, 5-15% točaka sekundarne strukture može se razlikovati između rendgenskih struktura i NMR modela istog proteina; drugo, postoji nedosljednost pridruživanja sekundarnih struktura različitim metoda, primjerice DSSP i STRIDE. Trenutne tehnike predviđanja još ne mogu dostići teoretsku granicu dijelom i zbog toga što ne uzimaju u obzir interakcije dugog dometa između aminokiselinskih ostataka.

6 LITERATURA

1. **Tsai, Stan C.** Proteomics: Prediction of protein structures. *An introduction to computational biochemistry.* New York : Wiley-Liss, Inc., 2002.

2. **Cox, Michael M. i Nelson, David L.** Amino acids, peptides, and proteins. *Lehninger Principles of Biochemistry.* s.l. : W. H. Freeman, 2004.

3. *ensemble based systems in decision making.* **Polikar, Robi.** 2006, IEEE Circuits and Systems Magazine, Svez. 6, str. 21-45.

4. *Bagging, Boosting, and C4.5.* **Quinlan, J R.** 1996. Proceedings of the Thirteenth National Conference on Artificial Intelligence. str. 725-730.

5. *Bagging Predictors.* **Breiman, Leo.** 2, s.l. : Springer Netherlands, 2005, Machine Learning, Svez. 24, str. 123-140.

6. *The Strength of Weak Learnability.* **Schapire, Robert E.** 2, s.l. : Kluwer Academic Publishers, 1990, Machine Learning, Svez. 5, str. 197-227.

7. *A decision-theoretic generalization of on-line learning and an application to boosting.* **Freund, Yoav i Schapire, Robert E.** 1, s.l. : Elsevier Inc., 1996, Journal of Computer and System Sciences, Svez. 55, str. 119-139.

8. **Schapire, Robert E.** The Boosting Approach to Machine Learning: An Overview. *Princeton University, Department of Computer Science.* [Mrežno] 19. 12 2001. www.cs.princeton.edu/~schapire/uncompress-papers.cgi/msri.ps.

9. *Multi-class AdaBoost.* **Zhu, Ji, i dr.** s.l. : International Press of Boston, 2009, Statistics and its interface, Svez. 2, str. 349-360.

10. *Theoretical Views of Boosting and applications.* **Schapire, Robert E.** s.l. : Springer-Verlag, 2009. Proceedings of the 10th International Conference on Algorithmic Learning Theory. str. 13-25.

11. *Random Forests.* **Breiman, Leo.** [ur.] Robert E Schapire. s.l. : Kluwer Academic Publishers, 2001, Machine Learning, Svez. 45, str. 5-32.

12. *The random subspace method for constructing decision forests.* **Ho, Tim Kan.** 8, s.l. : IEEE Computer Society, 1998, IEEE Transactions on Pattern Analysis and Machine Intelligence, Svez. 20, str. 832-844.

13. **Rokach, Lior i Maimon, Oded.** Decision Trees. *The data mining and knowledge discovery handbook.* s.l. : Springer, 2005, str. 165-187.

14. **Pujari, Arun K.** Decision Trees. *Data Mining Techniques.* s.l. : Universities Press (India) Private Ltd., 2001, 6.

15. Classification: Basic concepts, decision trees, and model evaluation. [aut. knjige] Pang Ning Tan, Michael Steinbach i Vipin Kumar. *Introduction to data mining.* s.l. : Addison-Wesley, 2006, str. 145-205.

16. *Bootstrap methods: Another look at the Jackknife.* **Efron, B.** 1, 1979, Annals of statistics, Svez. 7, str. 1-26.

17. *Prediction of protein secondary structure at better than 70% accuracy.* **Rost, Brukhard i Sander, Chris.** 1993, J. Mol. Biol., str. 584-599.

18. *Evaluation and improvement od multiple sequence methods for protein secondary structure prediction.* **Cuff, James A. i Barton, Geoffrey J.** s.l. : Wiley-Liss, Inc., 1999, PROTEINS: Structure, Function, and Genetics, Svez. 34, str. 508-519.

19. *Protein secondary structure prediction based on position-specific scoring matrices.* **Jones, David T.** s.l. : Elsevier, 1999, Journal of molecular biology, Svez. 292, str. 195-202.

20. **Fink, Lynn J, Weissig, Helge i Bourne, Philip E.** Other structure-based databases. [aut. knjige] Jenny Gu i Philip E Bourne. *Structural bioinformatics.* s.l. : John Wiley & Sons, Inc., 2009, 13.

21. *Protein sequence databases.* **Apweiler, Rolf, Bairoch, Amos i Wu, Cathy H.** s.l. : Elsevier Ltd., 2004, Current Opinion in Chemical Biology, Svez. 8, str. 76-80.

22. —. **Apweiler, Rolf, Bairoch, Amos i Wu, Cathy H.** s.l. : Elsevier Ltd., 2004, Current Opinion in Chemical Biology.

23. *Protein secondary structure assignment revisited: a detailed analysis of different assignment methods.* **Martin, Juliette, i dr.** 17, s.l. : BioMed Central Ltd., 2005, BMC Structural Biology, Svez. 5.

24. **Kaminska, Katarzina H, Milanowska, Kaja i Bujnicki, Janusz M.** The Basics of Protein Sequence Analysis. [aut. knjige] Janusz M. Bujnicki. *Prediction of Protein Structures, Functions and Interactions.* Wiltshire : John Wiley & Sons, Ltd., 2009.

25. *Improved tools for biological sequence comparison.* **Pearson, William R i Lipman, David J.** 8, s.l. : National Academy of Sciences, 1988, Proc. Natl. Acad. Sci., Svez. 85, str. 2444-2448.

26. *Basic local alignment search tool.* **Altschul, S F, i dr.** 3, s.l. : Academic Press, 1990, Journal of molecular biology, Svez. 215, str. 403-410.

27. *Amino acid substitution matrices from protein blocks.* **Henikoff, Steven i Henikoff, Jorja G.** 22, s.l. : National Academy of Sciences, 1992, Proc Natl Acad Sci U S A, Svez. 89, str. 10915-10919.

28. *Where did the BLOSUM62 alignment score matrix come from?* **Sean, Eddy R.** 8, s.l. : Nature Publishing Group, 2004, Nature Biotechnology, Svez. 22, str. 1035-1036.

29. **Majorek, Karolina, i dr.** First steps of protein structure prediction. [aut. knjige] Janusz M Bujnicki. *Prediction od Protein Structures, Functions, and Interactions.* s.l. : John Wiley & Sons, Ltd., 2009, 2.

30. *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* **Altschul, S F, i dr.** 17, s.l. : Oxford University Press, 1997, Nucleic Acids Res., Svez. 25, str. 3389–3402.

31. *Prediction of protein conformation.* **Chou, P Y i Fasman, G D.** 2, s.l. : American Chemical Society, 1974, Biochemistry, Svez. 13, str. 222-245.

32. *Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins.* **Garnier, J, Osguthorpe, D J i Robson, B.** 1, s.l. : Academic Press, 1978, Journal of molecular biology, Svez. 120, str. 97-120.

33. **Fasman, Gerald D.** The development of the prediction of protein structures. *Prediction of protein structure and the principles of protein conformation.* New York : Plenum Press, 1989.

34. *Literature survey of protein secondary structure prediction.* **Arjunan, Satya Nanda, Deris, Safaai i Illias, Rosli Md.** s.l. : UTM Press, 2001, Jurnal Teknologi, Svez. 34, str. 63-72.

35. **Deris, Safai, i dr.** *Protein secondary structure prediction from amino acid sequence using artificial intelligence technique.* s.l. : Faculty of Computer Science and Information System, Univerisit Teknologi Malaysia, 2007.

36. *The Jpred 3 secondary structure prediction server.* **Cole, Christian, Barber, Jonathan D i Barton, Geoffrey J.** 2008, Nucleic Acids Research, Svez. 36.

37. *Prediction of Protein Secondary Structure by Combining Nearest-neighbor Algorithms and Multiple Sequence Alignments.* **Salamov, Asaf A i Solovyev, Victor V.** 1, s.l. : Elsevier Ltd., 1995, Journal of Molecular Biology, Svez. 247, str. 11-15.

38. *Identification and application of the concepts important for accurate and reliable protein secondary structure prediction.* **King, Ross D i Sternberg, Michael J E.** s.l. : Cambridge University Press, 1996, Protein Science, Svez. 5, str. 2298-2310.

39. *Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence.* **Frishman, Dmitrij i Argos, Patrick.** 2, s.l. : Oxford University Press, Protein Engineering, Svez. 9, str. 133-142.

40. *Improving the accuracy of protein secondary structure prediction using structural alignment.* **Montgomerie, Scott, i dr.** 301, 2006, BMC Bioinformatics, Svez. 7.

41. **Branden, Carl i Tooze, John.** Prediction, Engineering and Design of Protein Structures. *Introduction to protein structure.* New York : Garland Publishing, 1999, 17.

42. *Coupled prediction of protein secondary and tertiary structure.* **Meller, Jens i Baker, David.** 21, s.l. : National Academy of Sciences, 2003, Proceedings of the National Academy of Sciences of the United States of America, Svez. 100.

43. *Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions.* **Simons, K T, i dr.** 1, s.l. : Academic Press, 1997, Journal of molecular biology, Svez. 268, str. 209-225.

44. *Combining sequence-based prediction methods and circular dichroism and infrared spectroscopic data to improve protein secondary structure determinations.* **Lees, Jonathan G i Janes, Robert W.** 24, s.l. : Biomed Central, 2008, BMC Bioinformatics, Svez. 9.

45. *Accurate prediction of protein secondary structure and solvent accessibility by consensus combiners of sequence and structure information.* **Pollastri, Gianluca, i dr.** 201, s.l. : Biomed Central Ltd., 2007, BMC Bioinformatics, Svez. 8.

46. *GOR V server for protein secondary structure prediction.* **Sen, Taner Z, i dr.** 11, s.l. : Oxford University Press, 2005, Svez. 21.

47. *Application of Multiple Sequence Alignment Profiles to Improve Protein Secondary Structure Prediction.* **Cuff, James A i Barton, Geoffrey J.** s.l. : Wiley-Liss Inc., 2000, PROTEINS: Structure, Function, and Genetics, Svez. 40, str. 502-511.

48. *Exploiting the past and the future in protein secondary structure prediction.* **Baldi, Pierre, i dr.** 11, s.l. : Oxford university press, 1999, Bioinformatics, Svez. 15.

49. *Improving the prediction of protein secondary structure in three and eight classes using reccurent neural networks and profiles.* **Pollastri, Gianluca, i dr.** s.l. : Wiley-Liss, Inc., 2002, PROTEINS: Structure, Function, and Genetics, Svez. 47, str. 228-235.

50. *Cascaded multiple classifiers for secondary structure prediction.* **Ouali, Mohammed i King, Ross D.** 6, s.l. : The protein society, 2000, Protein Science, Svez. 9, str. 1162–1176.

51. *Porter: a new, accurate server for protein secondary structure prediction.* **Pollastri, Gianluca i McLysaght, Aoife.** 8, s.l. : Oxford University Press, 2004, Bioinformatics, Svez. 21, str. 1719-1720.

52. *A simple and fast secondary structure prediction method using hidden neural networks.* **Lin, Kuang, i dr.** 2, s.l. : Oxford University Press, 2005, Bioinformatics, Svez. 21, str. 152-159.

53. *Achieving 80% Ten-fold Cross-validated Accuracy for Secondary Structure Prediction by Large-scale Training.* **Dor, Ofer i Zhou, Yaoqi.** s.l. : Wiley-Liss, 2007, PROTEINS: Structure, Function, and Bioinformatics, Svez. 66, str. 838-845.

54. *EVA_ evaluation of protein structure prediction servers.* **Koh, Ingrid Y, i dr.** 13, s.l. : Oxford University Press, 2003, Nucleic Acids Research, Svez. 31.

55. *Is protein structure still an enigma.* **Sobha, K, Kanakaraju, C i Yadav, Siva Krishna.** 25, Andhra Pradesh : Academic Journals, 2008, African Journal of Biotechnology, Svez. 7.

56. **Xu, Ying, Xu, Dong i Liang, Jie.** *Computational methods for protein structure prediction and modeling.* s.l. : Springer Science+Business Media, 2007. Svez. I.

57. **Konopka, Andrzej K. i Crabbe, M. James C.** *Compact handbook of computational biology.* New York : Marcel Dekker, 2004.

58. **Senekal, Frederik Petrus.** Protein secondary structure prediction using amino acid regularities. *University of pretoria electronic theses and dissertations.* [Mrežno] July 2008. http://upetd.up.ac.za/thesis/available/etd-01232009-120040/unrestricted/dissertation.pdf.

59. **Hu, Xiaohua i Pan, Yi.** *Knowledge discovery in bioinformatics: Techniques, Methods, and Applications.* New Jersey : John Wiley & Sons, 2007.

7 DODATAK

7.1 Perl skripte

7.1.1 pdb2dssp.pl

```
#      pdb2dssp.pl
#      proracun sekundarne strukture aminokiselinskih ostataka
#      na temelju PDB zapisa koristenjem DSSP programa

use strict;
use warnings;

my @skupovi=('RS126', 'CB513', 'PSIPRED');
if ( ! -e "dssp"){
       mkdir("dssp");
}
foreach (@skupovi){
       my $outdir='dssp/'.$_.'/';
       if ( ! -e $outdir){
              mkdir($outdir);
              print 'sasa';
       }
       my $indir="$_";
       print $indir,"\n",$outdir,"\n";
       print $_, "\n";
       my $nextname;
       while (defined($nextname = <$indir/*.pdb>)) {
       system
"dsspcmbi.exe","$nextname",($outdir.substr($nextname,length($_)+1,4).'.dssp');
       }
}
```

7.1.2 parseDSSP.pl

```
#      parseDSSP.pl
#      prociscavanje podataka dobivenih DSSP programom i
#      stvaranje zapisa o proteinima oblika:
#      "RedBR   |    PDB_SEQRES   |    DSSP_amino    |    SekStr"

use strict;
use warnings;
use Bio::Structure::SecStr::DSSP::Res;
use Bio::Structure::IO;

my @skupovi=('RS126', 'CB513', 'PSIPRED');
foreach (@skupovi){
       if ( ! -e 'sekvence'){
              mkdir('sekvence');
       }
       my $txtdir='sekvence/'.$_.'/';
       if ( ! -w $txtdir){
              mkdir($txtdir);
       }
       my $dssp_dir="dssp/"."$_";
       my $pdb_dir=$_;
       my $nextname;
       my $len=length $dssp_dir;
       while (defined($nextname = <$dssp_dir/*.dssp>)){
              print ((substr $nextname,$len+1,9)."\n");
                 my $dssp = new Bio::Structure::SecStr::DSSP::Res('-file'=>
"$nextname");
                 my $pdbID = $dssp->pdbID();
                 my $chainRef = $dssp->chains();
                 my @chains = sort  @{$chainRef};
                 my $structio = Bio::Structure::IO->new( -file =>
"$pdb_dir/$pdbID".'.pdb');
                 my $struc = $structio->next_structure;
                 foreach my $ch (@chains) {
                    my $chainid=$ch;
                    my $sek=$struc->seqres($chainid)->{'seq'};
                    my $file_txt=$txtdir.$pdbID.'_'."$ch".'.txt';
                    my $ss_elements_pts = $dssp->secBounds($ch);
                    print "Chain $ch:\n";
                    my $pos = 0;
                    my $max = 0;
                    foreach my $stretch (@{$ss_elements_pts}) {
                           my $start = $stretch->[0];
                           my $end = $stretch->[1];
                           if ($end =~ m/(\d+)/) { $end = $1; }
                           if ($end  > $max) { $max = $end; }
                    }
                    my ( @amino,@ss,@redbr,$seg,$pocetak );
                    my $segmenti = $dssp->_contSegs();
                    foreach $seg ( @{ $segmenti} ){
                           if ( $ch eq $seg->[ 2 ] ){
                                  $pocetak=$seg->[ 0 ];
                                  last;
                           }
                    }
                    my $br=$dssp->brojRezidua("$ch");
                    my $prekidi=0;
                    my $prekid=$br-1;
                    for my $res ($pocetak..$pocetak+$br-1) {
                           if ($dssp->{'Res'}->[$res]->{'amino_acid'} eq '!') {
                                  $prekidi=$prekidi+1;
                                  my $start=$dssp->{'Res'}->[$res-1]-
>{'pdb_resnum'}+1;
                                  $prekid=$start-$dssp->{'Res'}->[1]-
>{'pdb_resnum'}-1 if $prekidi==1;
```

```
                                    my $end=$dssp->{'Res'}->[$res+1]-
>{'pdb_resnum'}-1;
                                    for my $i ($start..$end) {
                                          push @redbr, $i;
                                          push @amino, '?';
                                          push @ss,'?';
                                    }
                              }
                              else {
                                    if ($dssp->{'Res'}->[$res]->{'amino_acid'} =~
/[a-z]/) {
                                          push @amino,'C';
                                    }
                                    else {
                                          push @amino, $dssp->{'Res'}->[$res]-
>{'amino_acid'};
                                    }
                                    push @redbr, $dssp->{'Res'}->[$res]-
>{'pdb_resnum'};
                                    push @ss, SekStr($dssp->{'Res'}->[$res]-
>{'ss_summary'});

                              }
                        }
                        my $AAniz=join "", @amino[0..$prekid-1];
                        my $indeks=index $sek, $AAniz;
                        if ($indeks){
                              for my $i (1..$indeks){
                                    unshift @amino, '?';
                                    unshift @ss, '?';
                                    unshift @redbr, 'x';
                              }
                        }
                        my $diff=(length $sek)-(scalar @amino);
                        if ($diff>0){
                              for my $i (1..$diff){
                                    push @amino,'?';
                                    push @ss,'?';
                                    push @redbr,'x';
                              }
                        }
                        $"="";
                        open TXT, "> $file_txt";
                        my @sekv=split "",$sek;
                        for my $red (1..length $sek){
                              print TXT "$redbr[$red-1] $sekv[$red-1] $amino[$red-1]
$ss[$red-1] \n";
                        }
                        close TXT;
                  }
            }
}

sub SekStr {
    my $sekI = shift;
    if ( $sekI eq 'H' || $sekI eq 'G' ) {
       return 'H';
    }
    if ( $sekI eq 'B' || $sekI eq 'E' ) {return 'E';}
    if ( $sekI eq 'T' || $sekI eq 'S' || $sekI eq 'I' || $sekI eq ' ' || !( $sekI
) ) {return 'C';}
    else {print "\nNe postoji!!!!\n"}
}
```

7.1.3 2PSSM.pl

```
#	2PSSM.pl
#	generiranje profila slijeda na temelju podataka prociscenih
#	parseDSSP.pl skriptom

use strict;
use warnings;
use Bio::Structure::SecStr::DSSP::Res;
use Bio::Structure::IO;
use File::Copy;

my @skupovi=('RS126', 'CB513', 'PSIPRED');
if ( ! -e "PSSM"){
	mkdir("PSSM");
}
foreach (@skupovi){
	my $txt_dir="sekvence/"."$_";
	my $pssm_dir="PSSM/"."$_";
	my $len=length $txt_dir;
	if ( ! -e "$pssm_dir"){
		mkdir("$pssm_dir");
	}
	if ( ! -e "$txt_dir/gotovo"){
		mkdir("$txt_dir/gotovo");
	}
	my $nextfile;
	while (defined($nextfile = <$txt_dir/*.txt>)){
		my (@aa, $pdbID, $chainID, @line);
		$pdbID=substr $nextfile,$len+1,4;
		$chainID=substr $nextfile,$len+6,1;
		print ((substr $nextfile,$len+1,6)."\n");
		open DATOTEKA, $nextfile;
		while (<DATOTEKA>){
			@line=split " ",$_;
			push @aa, $line[1];
		}
		close DATOTEKA;
		my $duljina=scalar @aa;

	#	Kreiranje PSIBLAST  ulazne datoteke (slijeda)
		$"="";
		open INPUT, ">input.txt";
		print INPUT '>'."$pdbID"."_$chainID\n";
		print INPUT @aa;
		close INPUT;

	#	PSIBLAST pretraga baze i generiranje PSSM matrice
		$"=" ";
		system "psiblast -db nr -query input.txt -out output.txt -
out_ascii_pssm pssm.txt -num_iterations 4";
		my $izvor="pssm.txt";
		my $cilj="$pssm_dir/$pdbID"."_$chainID.pssm";
		move($izvor,$cilj);
		move($nextfile,"$txt_dir/gotovo");
	}
}
```

7.1.4 2ARFF.pl

```
#	2ARFF.pl
#	stvaranje ARFF datoteke za svaki od skupova na temelju prociscenih
#	DSSP podataka i generiranih profila slijeda
use strict;
use warnings;
use Bio::Structure::SecStr::DSSP::Res;
use Bio::Structure::IO;

my @skupovi=('RS126', 'CB513', 'PSIPRED');
my @amino =
("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y","V")
;
if ( ! -e "pars"){
	mkdir("pars");
}
if ( ! -e "ARFF"){
	mkdir("ARFF");
}
my $prozor;
if (scalar @ARGV==2){
	$prozor=$ARGV[1];
	if ($prozor % 2==0){
	print("Duljina prozora mora biti neparan broj");
	exit;
	}
}
else{
	print("Pretpostavljena duljina prozora: 9\n");
	$prozor=9;
}
foreach (@skupovi){
	my $txt_dir="sekvence/"."$_";
	my $len=length $txt_dir;
	my $pssm_dir="PSSM/"."$_";
	my $nextfile;

	#	kreiranje zaglavlja ARFF datoteke
	open ARFF, "> ARFF/$_"."_2_$prozor".".ARFF";
	print ARFF '@RELATION '. "$_\n\n";
	print ARFF '@ATTRIBUTE pdbID STRING'."\n";
	print ARFF '@ATTRIBUTE chainID STRING'."\n";
	print ARFF '@ATTRIBUTE RedBr NUMERIC'."\n";
	for my $i (1..$prozor){
		print ARFF ('@ATTRIBUTE residue'."$i"."
{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y}\n")
	}
	for my $m (1..$prozor){
		for my $v (0..19){
			my $mn='@ATTRIBUTE profile'.$m.'_'.$amino[$v]." NUMERIC\n";
			print ARFF $mn;
		}
	}
	print ARFF '@ATTRIBUTE Class {E,C,H}'."\n\n";

	#	kreiranje podatkovnog dijela ARFF datoteke
	print ARFF '@DATA'."\n";
	my $brojac=0;
	while (defined($nextfile = <$txt_dir/*.txt>)){
		my (@aa, @dssp, @ss, $pdbID, $chainID, @redbr, @line);
		$pdbID=substr $nextfile,$len+1,4;
		$chainID=substr $nextfile,$len+6,1;
		print ((substr $nextfile,$len+1,6)."\n");
		open DATOTEKA, $nextfile;
		while (<DATOTEKA>){
			@line=split " ",$_;
```

```
                        push @redbr, $line[0];
                        push @aa, $line[1];
                        push @dssp, $line[2];
                        push @ss, $line[3];
                }
                close DATOTEKA;
                my $duljina=scalar @aa;
                $"=" ";
                my $pssm_dat=$pssm_dir."/".$pdbID."_$chainID.pssm";
                open my $PSSM, "< $pssm_dat" or die "pogreska";
                # ucitavanje profila slijeda (PSSM matrice) u strukturu polja
                my @pssm=@{radi($PSSM)};
                print"";
                my $odmak=int $prozor/2;
                my @AAokvir;
                my $resSS;
                my @PSSMokvir;
                $brojac=$brojac+1;
                #       pomicanje okvira duz slijeda i
                #       zapisivanje podataka u datoteku
                for my $res ($odmak..$duljina-$odmak-1){
                        if ($ss[$res] ne '?'){
                                @AAokvir=@aa[$res-$odmak..$res+$odmak];
                                @PSSMokvir=@pssm[($res-$odmak..$res+$odmak)];
                                $resSS=$ss[$res];
                                print ARFF "$pdbID,$chainID,$redbr[$res],";
                                print ARFF join ",", @AAokvir;
                                print ARFF ",";
                                for my $step(0..$prozor-1){
                                        print ARFF join ",",@{$PSSMokvir[$step]};
                                        print ARFF ",";
                                }
                                print ARFF "$resSS\n";
                        }
                }
        }
        print "\n $brojac";
        close ARFF;
}

sub radi{
        my $fh=shift;
        my @niz;
        my @a;
        #my $pom;
        while (<$fh>){
                @niz=split " ",$_;
                #@niz=split " ",$pom;
                if ((scalar @niz !=0) && ($niz[0]=~ /[0-9]/)){
                        my @v;
                        @niz=split " ",(substr $_, 70);
                        for my $i (0..19){
                                #push @v, $niz[22+$i];
                                push @v, $niz[$i];
                        }
                push @a, [@v];
                }
        }
        return \@a;
}
```

7.2 Skupovi proteinskih lanaca

7.2.1 RS126

1A45_A	1BMV_2	1ECA_A	1GD1_O	1LMB_3	1RBP_A	2AAT_A	2GLS_A	2MHU_A	2BUK_A	3CD4_A	3RNT_A
4RHV_1	5CYT_R	6CTS_A	8ADH_A	1ACX_A	1CBH_A	1ETU_A	1GDJ_A	1MCP_L	1RHD_A	2AK3_A	2GN5_A
2OR1_L	2TGP_I	3CLA_A	3TIM_A	4RHV_3	5ER2_E	6DFR_A	9API_A	1AZU_A	1CC5_A	1FC2_C	1GP1_A
1MRT_A	1S01_A	2ALP_A	2HMZ_A	2PAB_A	2TMV_P	3CLN_A	4BP2_A	4RHV_4	5HVP_A	6HIR_A	9API_B
1BBP_A	1CDT_A	1FDL_H	1HIP_A	1OVO_A	1SH1_A	2CAB_A	2I1B_A	2PCY_A	2TSC_A	3EBX_A	4CMS_A
4RXN_A	5LDH_A	6TMN_E	9INS_B	1BDS_A	1CRN_A	1FKF_A	1IL8_A	1PAZ_A	1TGS_I	2CCY_A	2LHB_A
2PHH_A	2UTG_A	3HMG_A	4CPA_I	4SDH_A	5LYZ_A	7CAT_A	9PAP_A	1BKS_A	1CSE_I	1FND_A	1IQZ_A
1PPT_A	1TNF_A	2CYP_A	2LTN_A	2RSP_A	2OZ9_R	3HMG_B	4CPV_A	4SGB_I	6ACN_A	7ICD_A	9WGA_A
1BKS_B	1CYO_A	1FXI_A	1L58_A	1PYP_A	1UBQ_A	2FOX_A	2LTN_B	2SNS_A	3AIT_A	3ICB_A	4GR1_A
4TS1_A	6CPA_A	7RSA_A	1BMV_1	1DUR_A	1G6N_A	1LAP_A	1R09_2	256B_A	2GBP_A	2MEV_4	2SOD_B
3BLM_A	3PGM_A	4PFK_A	4XIA_A	6CPP_A	8ABP_A						

7.2.2 CB513

154L_A	1AAZ_B	1ACX_A	1ADD_A	1ADE_B	1AHB_A	1ALK_B	1AMG_A	1AMP_A	1AOR_B	1AOZ_B	1ASW_A
1ATP_I	1AVH_B	1AYA_B	1AZU_A	1BAM_A	1BBP_A	1BCX_A	1BDO_A	1BDS_A	1BET_A	1BFG_A	1BMV_1
1BMV_2	1BNC_B	1BOV_B	1BPH_A	1BRS_E	1BSD_B	1CBG_A	1CBH_A	1CC5_A	1CDL_G	1CDT_A	1CEI_A
1CEL_B	1CEM_A	1CEO_A	1CEW_I	1CFB_A	1CFR_A	1CGU_A	1CHB_E	1CHD_A	1CHK_B	1CHM_B	1CKS_C
1CLC_A	1CNS_B	1COI_A	1COL_B	1COM_C	1CPC_L	1CPN_A	1CQA_A	1CRN_A	1CSE_I	1CSM_B	1CTF_A
1CTH_B	1CTM_A	1CTN_A	1CTU_A	1CXS_A	1CYX_A	1DAA_B	1DAR_A	1DEL_B	1DFJ_I	1DFN_B	1DIH_A
1DIK_A	1DIN_A	1DKZ_A	1DLC_A	1DNP_B	1DPG_B	1DSB_B	1DTS_A	1DUP_A	1DYN_B	1ECA_A	1ECE_B
1ECL_A	1ECP_F	1EDD_A	1EDM_C	1EDN_A	1EFT_A	1EFU_D	1EPB_B	1ESE_A	1ESL_A	1ETU_A	1EUU_A
1FBA_B	1FBL_A	1FC2_C	1FDL_H	1FDT_A	1FDX_A	1FIN_D	1FJM_B	1FKF_A	1FND_A	1FUA_A	1FUQ_B
1FXI_A	1GAL_A	1GCB_A	1GCM_C	1GD1_O	1GDJ_A	1GEP_A	1GFL_B	1GHS_B	1GKY_A	1GLN_A	1GMP_B
1GND_A	1GOG_A	1GP1_A	1GP2_A	1GP2_G	1GPC_A	1GPM_D	1GRJ_A	1GTM_C	1GTQ_B	1GYM_A	1HAN_A
1HCG_B	1HCR_A	1HIP_A	1HIW_S	1HJR_D	1HMP_B	1HMY_A	1HNF_A	1HOR_B	1HPL_B	1HSL_B	1HTR_P
1HUP_A	1HVQ_A	1HXN_A	1HYP_A	1IGN_B	1IL8_A	1ILK_A	1INP_A	1IRK_A	1ISA_B	1ISU_B	1JUD_A
1KIN_B	1KNB_A	1KPT_B	1KRC_A	1KRC_B	1KTE_A	1KTQ_A	1KUH_A	1L58_A	1LAP_A	1LAT_B	1LBA_A
1LBU_A	1LEH_B	1LIB_A	1LIS_A	1LKI_A	1LMB_3	1LPB_A	1LPE_A	1MAI_A	1MAS_B	1MCT_I	1MDA_J
1MDA_M	1MDT_A	1MJC_A	1MLA_A	1MMO_H	1MNS_A	1MOF_A	1MRR_B	1MRT_A	1MSP_B	1NAL_4	1NAR_A
1NBA_C	1NCG_A	1NDH_A	1NFP_A	1NGA_A	1NLK_L	1NOL_A	1NOX_A	1NOZ_B	1OAC_B	1ONR_B	1OTG_C
1OVB_A	1OVO_A	1OXY_A	1OYC_A	1PAZ_A	1PBP_A	1PBW_B	1PDA_A	1PDN_C	1PDO_A	1PGA_A	1PHT_A
1PII_A	1PKY_C	1PMI_A	1PNM_B	1PNT_A	1POC_A	1POW_B	1PPI_A	1PPT_A	1PTR_A	1PTX_A	1PYP_A
1PYT_A	1QBB_A	1QRD_B	1R09_2	1RBP_A	1REC_A	1REG_Y	1REQ_C	1RHD_A	1RHG_C	1RIE_A	1RIS_A
1RLD_S	1RLR_A	1RPO_A	1RSY_A	1RVV_Z	1S01_A	1SCU_D	1SCU_E	1SEI_B	1SES_A	1SFE_A	1SFT_B
1SH1_A	1SMN_B	1SMP_I	1SPB_P	1SRA_A	1SRJ_A	1STF_I	1STM_E	1SVB_A	1TAB_I	1TAQ_A	1TCB_A
1TCR_A	1TFR_A	1THT_B	1THX_A	1TIE_A	1TIF_A	1TIG_A	1TII_C	1TML_A	1TND_B	1TNF_A	1TPL_B
1TRB_A	1TRH_A	1TRK_B	1TSP_A	1TSS_B	1TUL_A	1TUP_C	1UBD_C	1UBQ_A	1UDH_A	1UMU_B	1VCA_B
1VCC_A	1VHH_A	1VHR_B	1VID_A	1VJS_A	1VMO_B	1VNC_A	1VOK_B	1VPT_A	1WAP_V	1WFB_B	1WHI_A
1WSY_A	1WSY_B	1XVA_B	1YPT_B	1YRN_A	1ZNB_B	1ZYM_B	256B_A	2AAI_B	2AAT_A	2ABK_A	2ADM_B
2AFN_C	2AK3_A	2ALP_A	2ASR_A	2BAT_A	2BLT_B	2BOP_A	2CAB_A	2CCY_A	2CMD_A	2CPO_A	2CYP_A
2DKB_A	2DLN_A	2DNJ_A	2EBN_A	2END_A	2ERL_A	2FOX_A	2FXB_A	2GBP_A	2GCR_A	2GLS_A	2GN5_A
2GSQ_A	2HFT_A	2HHM_B	2HIP_B	2HMZ_A	2HPR_A	2I1B_A	2LTN_A	2LTN_B	2MEV_4	2MHU_A	2MLT_B
2MTA_C	2NAD_B	2NPX_A	2OLB_A	2OR1_L	2PAB_A	2PGD_A	2PHH_A	2PHY_A	2POL_B	2REB_A	2RSL_A
2RSP_A	2SCP_B	2SIL_A	2SNS_A	2SOD_B	2SPT_A	2STV_A	2TGI_A	2TGP_I	2TMD_B	2TMV_P	2TRT_A
2TSC_A	2UTG_A	2WRP_R	2YHX_A	3AIT_A	3B5C_A	3BCL_A	3BLM_A	3CD4_A	3CHY_A	3CLA_A	3CLN_A
3COX_A	3ECA_B	3GAP_A	3HMG_A	3HMG_B	3ICB_A	3INK_D	3MDD_B	3PGK_A	3PGM_A	3PMG_B	3RNT_A
3TIM_A	4BP2_A	4CPA_I	4FIS_B	4GR1_A	4PFK_A	4RHV_1	4RHV_3	4RHV_4	4RXN_A	4SDH_A	4SGB_I
4TS1_A	4XIA_A	5CYT_R	5ER2_E	5LDH_A	5LYZ_A	5SIC_I	6ACN_A	6CPA_A	6CPP_A	6CTS_A	6DFR_A
6HIR_A	6RLX_C	6RLX_D	6TMN_E	7CAT_A	7ICD_A	7RSA_A	821P_A	8ADH_A	9API_A	9API_B	9INS_B
9PAP_A	9WGA_A										

7.2.3 PSIPRED

119L A	13PK A	153L A	1A04 A	1A0A A	1A0B A	1A0I A	1A0P A	1A0T P	1A34 A	1AA0 A	1AA2 A
1AA6 A	1AA7 A	1AAB A	1AAC A	1AAF A	1AAP A	1AAY A	1AB2 A	1AB3 A	1AB9 B	1AB9 C	1ABA A
1ABO A	1ABO A	1ABR A	1ABR B	1ABV A	1ABZ A	1AC0 A	1AC5 A	1AC6 A	1ACA A	1ACC A	1ACF A
1ACI A	1ACP A	1ACW A	1ACX A	1AD0 A	1AD0 B	1AD2 A	1AD3 A	1AD5 A	1AD9 H	1AD9 L	1ADE A
1ADJ A	1ADN A	1ADO A	1ADR A	1ADS A	1ADW A	1ADX A	1AE5 A	1AE6 H	1AE6 L	1AE7 A	1AE9 A
1AEI A	1AEP A	1AER A	1AEW A	1AF5 A	1AF6 A	1AF7 A	1AF8 A	1AFC A	1AFI A	1AFO A	1AFP A
1AFR A	1AFS A	1AFV H	1AFW A	1AFW B	1AG2 A	1AG8 A	1AG9 A	1AGD A	1AGG A	1AGI A	1AGJ A
1AGN A	1AGO A	1AGR E	1AGT A	1AGX A	1AH6 A	1AH7 A	1AH9 A	1AHD P	1AHL A	1AHO A	1AHP A
1AHO A	1AHS A	1AHT H	1AHT L	1AI1 H	1AIE A	1AIH A	1AIJ H	1AIJ L	1AIJ M	1AIK C	1AIK N
1AIL A	1AIM A	1AIP C	1AIR A	1AIS A	1AIS B	1AJ3 A	1AJ6 A	1AJ8 A	1AJE A	1AJJ A	1AJS A
1AJY A	1AJZ A	1AK0 A	1AK1 A	1AK2 A	1AK4 C	1AK5 A	1AK6 A	1AKE A	1AKO A	1AKP A	1AKY A
1AKZ A	1AL0 1	1AL0 B	1AL3 A	1ALA A	1ALK A	1ALL A	1ALL B	1ALO A	1ALV A	1ALY A	1AM5 A
1AM7 A	1AMB A	1AMF A	1AMK A	1AMM A	1AMP A	1AMU A	1AMW A	1AMY A	1AN2 A	1AN4 A	1AN9 A
1ANF A	1ANG A	1ANN A	1ANS A	1ANU A	1ANV A	1ANW A	1AO0 A	1AO5 A	1AO7 D	1AO7 E	1AOA A
1AOC A	1AOE A	1AOG A	1AOH B	1AOK B	1AOL A	1AON O	1AOO A	1AOP A	1AOO A	1AOR A	1AOT F
1AOY A	1AOZ A	1AP2 B	1AP6 A	1AP8 A	1APA A	1APF A	1APJ A	1APO A	1APQ A	1APS A	1APX A
1APY A	1APY B	1AO0 A	1AO5 A	1AO6 A	1AOB A	1AOC A	1AOD A	1AOD B	1AOK H	1AOK L	1AOT A
1AOZ A	1AR1 A	1AR1 B	1AR1 C	1AR1 D	1ARB A	1ARD A	1ARK A	1ARS A	1ARU A	1AS4 A	1AS4 B
1AS8 A	1ASH A	1ASS A	1ASZ A	1AT0 A	1ATA A	1ATI A	1ATL A	1ATT A	1ATU A	1ATX A	1ATY A
1ATZ A	1AU7 A	1AUA A	1AUI A	1AUI B	1AUK A	1AUN A	1AUO A	1AUT C	1AUT L	1AUU A	1AUV A
1AUW A	1AUY A	1AVC A	1AVD A	1AVK A	1AVM A	1AVO A	1AVO B	1AVP A	1AVQ A	1AVS A	1AVY A
1AW0 A	1AW2 A	1AWC A	1AWC B	1AWD A	1AWE A	1AWJ A	1AWS A	1AX4 A	1AXC A	1AXH A	1AXI A
1AXI B	1AXJ A	1AXN A	1AXS H	1AXS L	1AYA A	1AYJ A	1AYL A	1AYM 1	1AYM 2	1AYM 3	1AYM 4
1AZC A	1AZS A	1AZS B	1AZS C	1AZV A	1AZZ A	1B5M A	1BAB A	1BAB B	1BAF H	1BAH A	1BAK A
1BAL A	1BAM A	1BBA A	1BBD H	1BBH A	1BBI A	1BBJ H	1BBJ L	1BBO A	1BBP A	1BBR L	1BBT 1
1BBT 2	1BBT 3	1BBT 4	1BCF A	1BCO A	1BCP A	1BCP B	1BCP C	1BCP D	1BCP F	1BDB A	1BDM A
1BDO A	1BDS A	1BEB A	1BEC A	1BED A	1BEO A	1BET A	1BFD A	1BFG A	1BFI A	1BFM A	1BFS A
1BFT A	1BGC A	1BGE A	1BGF A	1BGK A	1BGL A	1BGP A	1BGW A	1BHA A	1BHG A	1BHP A	1BI6 H
1BIA A	1BIF A	1BIN A	1BJM A	1BKF A	1BLE A	1BLF A	1BLJ A	1BLU A	1BMF A	1BMF D	1BMF G
1BMG A	1BMP A	1BMT A	1BMV 1	1BMV 2	1BNB A	1BNC A	1BND A	1BND B	1BNO A	1BOL A	1BOM A
1BOM B	1BOR A	1BOV A	1BP1 A	1BPB A	1BPY A	1BOU B	1BRE A	1BRN L	1BRO A	1BRS D	1BRY Y
1BSR A	1BTK A	1BTL A	1BTM A	1BTN A	1BTO A	1BTS A	1BUC A	1BUN A	1BUN B	1BUS A	1BV1 A
1BVD A	1BVP 1	1BW3 A	1BYB A	1C2R A	1C5A A	1CAA A	1CAU A	1CAU B	1CB1 A	1CB2 A	1CBG A
1CBH A	1CBI A	1CBN A	1CBS A	1CBY A	1CC5 A	1CCD A	1CCH A	1CCR A	1CD1 A	1CD8 A	1CDC B
1CDG A	1CDK A	1CDK I	1CDL G	1CDO A	1CDO A	1CDT A	1CDW A	1CDY A	1CEA A	1CEI A	1CEL A
1CEM A	1CEO A	1CER O	1CEW I	1CEX A	1CFA A	1CFB A	1CFE A	1CFG A	1CFH A	1CFI A	1CFP A
1CFR A	1CFV H	1CFV L	1CFY A	1CG2 A	1CGD A	1CGH A	1CGM E	1CGO A	1CGT A	1CHC A	1CHD A
1CHK A	1CHL A	1CHM A	1CID A	1CII A	1CIS A	1CIU A	1CIY A	1CJL A	1CKA A	1CKI A	1CKM A
1CKS A	1CLC A	1CLD A	1CLE A	1CLF A	1CLH A	1CLL A	1CLO L	1CLP A	1CLX A	1CLZ H	1CMB A
1CMR A	1CNP A	1CNT 1	1CNV A	1COD A	1COI A	1COL A	1COO A	1COR A	1COS A	1COT A	1COV 1
1COV 2	1COV 3	1COV 4	1COY A	1CP3 A	1CPC A	1CPC B	1CPN A	1CPO A	1CPQ A	1CPT A	1CPY A
1COA A	1CRB A	1CRE A	1CRK A	1CRY A	1CSC A	1CSE E	1CSE I	1CSH A	1CSK A	1CSN A	1CSP A
1CSY A	1CTA A	1CTF A	1CTJ A	1CTN A	1CTO A	1CTT A	1CTX A	1CUK A	1CVL A	1CWP A	1CX2 A
1CXC A	1CYD A	1CYG A	1CYJ A	1CYN A	1CYU A	1CYX A	1D66 A	1DAA A	1DAD A	1DAN H	1DAN L
1DAP A	1DAR A	1DBB H	1DBO A	1DBR A	1DCO A	1DCT A	1DDF A	1DDT A	1DEA A	1DEC A	1DEF A
1DEH A	1DEK A	1DEM A	1DFB H	1DFI A	1DFN A	1DHK A	1DHK B	1DHM A	1DHP A	1DHR A	1DHS A
1DHY A	1DIF A	1DIK A	1DIN A	1DIP A	1DIV A	1DJA A	1DJX A	1DKG A	1DKG D	1DKT A	1DKZ A
1DLC A	1DMB A	1DMC A	1DME A	1DMR A	1DMX A	1DNP A	1DOI A	1DOK A	1DOR A	1DOS A	1DOT A
1DOX A	1DPE A	1DPG A	1DPO A	1DRF A	1DRO A	1DRS A	1DRW A	1DSU A	1DTC A	1DTK A	1DTX A
1DUB A	1DUP A	1DUT A	1DVF C	1DVF D	1DVH A	1DXG A	1DXY A	1DYN A	1DYR A	1EAF A	1EAG A
1EAL A	1EAP A	1EAP B	1EBD A	1EBD C	1EBP A	1ECA A	1ECE A	1ECF A	1ECI A	1ECI B	1ECL A
1ECM A	1ECP A	1ECR A	1EDE A	1EDG A	1EDH A	1EDI A	1EDM B	1EDN A	1EDT A	1EFN B	1EFT A

1EFU A	1EFU B	1EFV A	1EFV B	1EG1 A	1EGD A	1EGF A	1EGO A	1EHS A	1EIT A	1ELG A	1ELP A
1ELT A	1EMA A	1EMN A	1EMY A	1ENH A	1ENP A	1ENY A	1EPA A	1EPM E	1EPS A	1ERD A	1ERI A
1ERK A	1ERP A	1ERV A	1ERY A	1ESC A	1ESF A	1ESL A	1ETF B	1ETH A	1ETP A	1EUR A	1EXG A
1EXN A	1EXP A	1EXT A	1EZM A	1F3Z A	1FAI H	1FAO A	1FAS A	1FAT A	1FBA A	1FBI H	1FBR A
1FBT A	1FC2 C	1FC2 D	1FCA A	1FCB A	1FCD A	1FCD C	1FCT A	1FDH G	1FDR A	1FDS A	1FEC A
1FFH A	1FGJ A	1FGJ B	1FGK A	1FGN H	1FGN L	1FGP A	1FGV H	1FGV L	1FIE A	1FIG H	1FIM A
1FIP A	1FIT A	1FIV A	1FJL A	1FJM A	1FKF A	1FKX A	1FLE I	1FLI A	1FLP A	1FLR H	1FMB A
1FMC A	1FMD 1	1FMD 2	1FMD 3	1FMK A	1FMT A	1FMT B	1FNA A	1FNC A	1FOK A	1FON A	1FOR H
1FOS E	1FOS F	1FPT H	1FRB A	1FRD A	1FRE A	1FRO A	1FRP A	1FRR A	1FRS A	1FRV A	1FRV B
1FSB A	1FSD A	1FSU A	1FT1 A	1FT1 B	1FTA A	1FTN A	1FTP A	1FTT A	1FTZ A	1FUA A	1FUI A
1FUJ A	1FUR A	1FUS A	1FVC A	1FVC B	1FVK A	1FVL A	1FVP A	1FWC A	1FWC B	1FWC C	1FWP A
1FXD A	1FXI A	1FXR A	1FYC A	1FZB A	1FZB B	1G3P A	1GAB A	1GAD O	1GAF H	1GAF L	1GAI A
1GAL A	1GAN A	1GAR A	1GAT A	1GBG A	1GBO A	1GCA A	1GCB A	1GCL A	1GCM A	1GCN A	1GD1 O
1GDH A	1GEC E	1GEN A	1GES A	1GFC A	1GFF 1	1GFF 2	1GGA O	1GGG A	1GGI H	1GGI L	1GHC A
1GHF H	1GHJ A	1GHS A	1GIA A	1GIF A	1GIG H	1GKS A	1GKY A	1GLA G	1GLN A	1GLO A	1GLU A
1GND A	1GNH A	1GNW A	1GOF A	1GOT A	1GOT B	1GOT G	1GOW A	1GOX A	1GP1 A	1GP2 G	1GPB A
1GPC A	1GPH 1	1GPL A	1GPM A	1GPO H	1GPO L	1GPR A	1GPS A	1GPT A	1GRJ A	1GSA A	1GSE A
1GSU A	1GTA A	1GTM A	1GTP A	1GTO A	1GTR A	1GUA A	1GUA B	1GUO A	1GUR A	1GVP A	1GYP A
1GZI A	1HA1 A	1HAE A	1HAN A	1HAV A	1HBG A	1HBH A	1HBH B	1HC1 A	1HCB A	1HCC A	1HCD A
1HCG A	1HCG B	1HCL A	1HCN A	1HCN B	1HCO A	1HCR A	1HCV A	1HCW A	1HCZ A	1HDA A	1HDA B
1HDC A	1HDG O	1HDJ A	1HDP A	1HDS A	1HDS B	1HEV A	1HFC A	1HFI A	1HFS A	1HFX A	1HFY A
1HGE A	1HGU A	1HGX A	1HIA I	1HIL A	1HIL B	1HIP A	1HIW A	1HJR A	1HKS A	1HLB A	1HLC A
1HLE A	1HLE B	1HLM A	1HLO A	1HLP A	1HMA A	1HME A	1HML A	1HMP A	1HMT A	1HMY A	1HNA A
1HNF A	1HNR A	1HOC A	1HOE A	1HP8 A	1HPG A	1HPH A	1HPI A	1HPL A	1HPM A	1HPT A	1HOI A
1HRA A	1HRC A	1HRD A	1HRH A	1HRJ A	1HRM A	1HRN A	1HRO A	1HRT I	1HRY A	1HSB A	1HSB B
1HSL A	1HSO A	1HST A	1HTI A	1HTM B	1HTN A	1HTP A	1HTR B	1HTR P	1HUC A	1HUC B	1HUE A
1HUI B	1HUL A	1HUM A	1HUP A	1HUR A	1HUS A	1HUW A	1HXN A	1HXP A	1HYH A	1HYM A	1HYM B
1HYP A	1IAB A	1IAG A	1IAI H	1IAI I	1IAI M	1IBA A	1IBE A	1IBE B	1IBG H	1IBG L	1ICA A
1ICE A	1ICE B	1IDA A	1IDK A	1IDO A	1IDS A	1IDY A	1IEA A	1IEA B	1IF1 A	1IFC A	1IFI A
1IFT A	1IGC L	1IGD A	1IGF H	1IGJ B	1IGL A	1IGM H	1IGM L	1IGN A	1IGT A	1IGT B	1IHF A
1IHF B	1IHP A	1IHV A	1IIB A	1IKF H	1IKU A	1IL6 A	1ILR 1	1IML A	1IMP A	1IND H	1INP A
1IOA A	1IOB A	1IOW A	1IPH A	1IPS A	1IR3 A	1IRF A	1IRO A	1IRS A	1ISC A	1ISK A	1ISO A
1ISU A	1ITB B	1ITF A	1ITG A	1ITH A	1IUZ A	1IVA A	1IVD A	1IVY A	1IXH A	1IYU A	1JAC A
1JAF A	1JBC A	1JCV A	1JDB B	1JDB C	1JDC A	1JDW A	1JER A	1JET A	1JHG A	1JHL H	1JHL L
1JLI A	1JLY A	1JMC A	1JOI A	1JPC A	1JRH H	1JRH I	1JSG A	1JSU C	1JSW A	1JTB A	1JUD A
1JUG A	1JUL A	1JUN A	1JVR A	1JXP A	1KAL A	1KAO A	1KAP P	1KAW A	1KAZ A	1KBA A	1KBC A
1KCW A	1KDU A	1KEL H	1KEV A	1KFS A	1KID A	1KIT A	1KLA A	1KLO A	1KMM A	1KNB A	1KNY A
1KOA A	1KOB A	1KPF A	1KPT A	1KRN A	1KRS A	1KSI A	1KSR A	1KTE A	1KTX A	1KUH A	1KVD A
1KVE A	1KVE B	1KVO A	1KXI A	1KXU A	1KZU A	1KZU B	1LAB A	1LAM A	1LAT A	1LBA A	1LBD A
1LBE A	1LBU A	1LCC A	1LCI A	1LCL A	1LCT A	1LDG A	1LDL A	1LDN A	1LDR A	1LEA A	1LED A
1LEH A	1LEN B	1LFA A	1LFB A	1LFO A	1LGH A	1LGH B	1LGR A	1LGY A	1LHT A	1LIA A	1LIA B
1LID A	1LIL A	1LIS A	1LIT A	1LKI A	1LKK A	1LKT A	1LLA A	1LLC A	1LLD A	1LLP A	1LMB 3
1LMK A	1LML A	1LMQ A	1LMW B	1LNH A	1LOE B	1LOI A	1LOP A	1LPB A	1LPB B	1LPF A	1LPP A
1LQH A	1LRE A	1LRV A	1LSI A	1LST A	1LT5 D	1LTE A	1LTS A	1LTS C	1LUC A	1LUC B	1LVE A
1LVL A	1LXA A	1LXD A	1LYB A	1LYB B	1LYL A	1LYP A	1LZR A	1MAH A	1MAI A	1MAJ A	1MAM H
1MAT A	1MAZ A	1MBA A	1MBE A	1MBG A	1MBL A	1MBS A	1MCP H	1MCT A	1MCT I	1MCW W	1MDA H
1MDA L	1MDC A	1MDL A	1MDY A	1MEA A	1MEE A	1MEK A	1MEL A	1MEM A	1MEY C	1MFA L	1MFA H
1MFE H	1MFE L	1MGS A	1MH1 A	1MHC A	1MHC B	1MHL A	1MHL C	1MHY B	1MHY D	1MHY G	1MIL A
1MIM H	1MIM L	1MIO A	1MIO B	1MJC A	1MKA A	1MLA A	1MLB B	1MLD A	1MMC A	1MML A	1MMQ A
1MN1 A	1MNG A	1MNM A	1MNM C	1MNT A	1MOF A	1MOL A	1MPA H	1MPG A	1MPP A	1MRE H	1MRG A
1MRJ A	1MRK A	1MRP A	1MSC A	1MSI A	1MSK A	1MSP A	1MTR A	1MTX A	1MTY B	1MTY D	1MTY G
1MUC A	1MUG A	1MUP A	1MUT A	1MVI A	1MVJ A	1MVM A	1MWE A	1MXA A	1MYJ A	1MYL A	1MYN A
1MYT A	1MZM A	1NAH A	1NAL 1	1NAR A	1NAW A	1NBA A	1NBC A	1NBV H	1NCB H	1NCB L	1NCI A
1NCS A	1NCT A	1NCV A	1NDH A	1NEA A	1NEQ A	1NEU A	1NFA A	1NFD A	1NFD B	1NFD E	1NFD F

1NFN A	1NFP A	1NGQ H	1NGR A	1NHK L	1NHP A	1NIF A	1NIN A	1NIP A	1NIR A	1NKL A	1NLD H
1NLO C	1NLS A	1NMB H	1NNC A	1NNT A	1NOA A	1NOR A	1NOV A	1NOV D	1NOX A	1NOY A	1NP4 A
1NPC A	1NPK A	1NPO A	1NQB A	1NRA A	1NSA A	1NSC A	1NSG B	1NSJ A	1NSO A	1NSY A	1NTN A
1NTR A	1NTX A	1NUE A	1NUL A	1NWP A	1NXB A	1NZY A	1OAC A	1OAT A	1OBP A	1OBR A	1OBW A
1OBW B	1OCP A	1OCT C	1OEF A	1OEG A	1OEN A	1OFG A	1OFV A	1OIS A	1OJT A	1OMN A	1ONC A
1ONE A	1ONR A	1OPB A	1OPC A	1OPG H	1OPR A	1ORC A	1ORD A	1ORT A	1OSA A	1OSP H	1OSP L
1OSP O	1OTF A	1OTG A	1OUN A	1OUT A	1OUT B	1OVA A	1OVW A	1OXA A	1OYC A	1P1P A	1P38 A
1PAA A	1PAF A	1PAL A	1PAM A	1PAX A	1PAZ A	1PBE A	1PBG A	1PBK A	1PBN A	1PBW A	1PCA A
1PCE A	1PCF A	1PCH A	1PCR M	1PCS A	1PDA A	1PDC A	1PDG A	1PDN C	1PDO A	1PDR A	1PDZ A
1PEA A	1PEH A	1PEI A	1PEX A	1PFC A	1PFI A	1PFK A	1PFS A	1PFT A	1PFX C	1PFX L	1PGB A
1PGS A	1PGT A	1PHB A	1PHK A	1PHN A	1PHN B	1PHO A	1PHP A	1PHR A	1PHT A	1PI2 A	1PID A
1PID B	1PII A	1PJR A	1PK4 A	1PKM A	1PKP A	1PKY A	1PLA A	1PLC A	1PLF A	1PLG H	1PLP A
1PLO A	1PLS A	1PMA A	1PMA B	1PMC A	1PMI A	1PML A	1PMP A	1PMY A	1PNB A	1PNB B	1PNE A
1PNH A	1PNK A	1PNK B	1POA A	1POC A	1POI A	1POI B	1PON B	1POT A	1POV 1	1POV 3	1POX A
1PP2 R	1PPA A	1PPE I	1PPF E	1PPN A	1PPO A	1PPR M	1PPT A	1PRC C	1PRE A	1PRN A	1PRR A
1PRU A	1PS1 A	1PS2 A	1PSC A	1PSD A	1PSE A	1PSJ A	1PSK H	1PSK L	1PSM A	1PSO E	1PSV A
1PTF A	1PTH A	1PTO A	1PTY A	1PUC A	1PUD A	1PUE E	1PUT A	1PVA A	1PVC 1	1PVC 2	1PVC 3
1PVC 4	1PVD A	1PVU A	1PYA A	1PYA B	1PYC A	1PYI A	1PYS A	1PYS B	1PYT A	1PYT D	1OAP A
1OBA A	1OBE A	1ODP A	1OLI A	1ONF A	1OOA A	1OOR A	1OPA A	1OPG A	1ORD A	1OUE A	1OYP A
1R09 1	1R09 2	1R09 3	1R09 4	1R1A 1	1R1A 2	1R1A 3	1R69 A	1RA9 A	1RAI A	1RAI B	1RBL A
1RBL M	1RCB A	1RCD A	1RCF A	1RCY A	1RDG A	1RDO 1	1RDS A	1REC A	1REG X	1REI A	1REO A
1REO B	1RES A	1RFB A	1RFS A	1RGE A	1RGP A	1RGS A	1RHG A	1RHI 1	1RHI 2	1RHI 3	1RHO A
1RHP A	1RHS A	1RIE A	1RIL A	1RIP A	1RIS A	1RKD A	1RLA A	1RLR A	1RLW A	1RMD A	1RMF H
1RMG A	1RMV A	1ROD A	1ROE A	1ROM A	1RON A	1ROO A	1ROT A	1RPA A	1RPB A	1RPM A	1RPO A
1RRO A	1RSS A	1RSY A	1RTF B	1RTP 1	1RTU A	1RVA A	1RVV A	1RYT A	1SAC A	1SAP A	1SAT A
1SBP A	1SCH A	1SCM A	1SCO A	1SCT A	1SCT B	1SCU A	1SCU B	1SCY A	1SDF A	1SE4 A	1SEB A
1SEI A	1SEM A	1SES A	1SFE A	1SFT A	1SGP E	1SGP I	1SGT A	1SH1 A	1SHA A	1SHC A	1SHF A
1SHG A	1SHP A	1SIG A	1SIS A	1SJU A	1SKY B	1SKY E	1SKZ A	1SLT A	1SLU A	1SLY A	1SMD A
1SME A	1SMN A	1SMP I	1SMR A	1SMT A	1SMV A	1SNB A	1SOL A	1SP1 A	1SP2 A	1SPB P	1SPF A
1SPG A	1SPG B	1SPH A	1SPI A	1SOC A	1SRA A	1SRB A	1SRD A	1SRI A	1SRO A	1SRR A	1SRS A
1SSO A	1STD A	1STE A	1STF I	1STM A	1STU A	1SUP A	1SVA 1	1SVB A	1SVN A	1SVP A	1SVO A
1SXC A	1SXL A	1SXM A	1T7P A	1TAB I	1TAD A	1TAF A	1TAF B	1TAL A	1TAP A	1TAO A	1TBD A
1TBR R	1TC3 C	1TCA A	1TCG A	1TCR A	1TDT A	1TEH A	1TEN A	1TER A	1TET H	1TF3 A	1TF4 A
1TFD A	1TFE A	1TFI A	1TFP A	1TFR A	1TFS A	1TFX C	1TGJ A	1TGS I	1TGX A	1THE A	1THG A
1THJ A	1THM A	1THT A	1THV A	1THX A	1TIB A	1TIE A	1TIF A	1TIG A	1TIH A	1TII A	1TII C
1TII D	1TIS A	1TIT A	1TIV A	1TLF A	1TLK A	1TME 1	1TME 2	1TME 3	1TME 4	1TML A	1TMY A
1TNF A	1TNR A	1TNS A	1TOC R	1TOF A	1TON A	1TPF A	1TPG A	1TPH 1	1TPL A	1TPM A	1TRB A
1TRE A	1TRK A	1TRL A	1TRN A	1TRY A	1TSK A	1TSY A	1TTB A	1TUC A	1TUD A	1TUL A	1TUP A
1TVD A	1TVS A	1TVX A	1TX4 A	1TXA A	1TXM A	1TYS A	1TYV A	1TZE E	1U9A A	1UAE A	1UAG A
1UBD C	1UBI A	1UBS B	1UBY A	1UCB H	1UCB L	1UCH A	1UCY H	1UCY J	1UDC A	1UDG A	1UDH A
1UDI I	1UKR A	1UKZ A	1ULA A	1ULO A	1UMU A	1UNA A	1UNK A	1URN A	1UTG A	1UXC A	1UXY A
1V39 A	1VAO A	1VAP A	1VCA A	1VCC A	1VCP A	1VDC A	1VDF A	1VDR A	1VFA A	1VFA B	1VGE H
1VGE L	1VHB A	1VHH A	1VHI A	1VHP A	1VHR A	1VIB A	1VID A	1VIE A	1VIG A	1VII A	1VIN A
1VIP A	1VJS A	1VJW A	1VKT A	1VLS A	1VLX A	1VMO A	1VNC A	1VND A	1VOK A	1VOL A	1VOM A
1VPI A	1VPS A	1VPT A	1VPU A	1VRT A	1VSD A	1VSG A	1VTM P	1VTP A	1VTX A	1VVC A	1VWL B
1WAB A	1WAD A	1WAJ A	1WAP A	1WBA A	1WDC A	1WDC B	1WDC C	1WER A	1WFB A	1WGJ A	1WHI A
1WHO A	1WHT A	1WHT B	1WIT A	1WJD B	1WKT A	1WPO A	1WSA A	1WTL A	1WTU A	1XAA A	1XBL A
1XBR A	1XDT R	1XER A	1XGS A	1XIB A	1XIK A	1XIM A	1XJO A	1XLA A	1XNB A	1XSM A	1XSO A
1XTC A	1XTC C	1XVA A	1XXB A	1XYN A	1XYP A	1XYZ A	1YAI A	1YAL A	1YAS A	1YAT A	1YBV A
1YCC A	1YCO A	1YCR A	1YCS B	1YDV A	1YEC H	1YEC L	1YED H	1YFO A	1YGE A	1YGP A	1YKF A
1YNA A	1YPC I	1YPP A	1YPR A	1YRN A	1YRN B	1YTB A	1YTC A	1YTF B	1YTF C	1YTF D	1YTI A
1YTW A	1YUA A	1YUB A	1YUF A	1YUH H	1YUI A	1YVE I	1ZAO A	1ZDA A	1ZEC A	1ZFD A	1ZFO A
1ZIA A	1ZII A	1ZIN A	1ZNB A	1ZNC A	1ZNF A	1ZTN A	1ZTO A	1ZXO A	1ZYM A	256B A	2AAA A
2AAI B	2AAK A	2ABK A	2ABX A	2ACG A	2ACH B	2ACK A	2ACT A	2ACY A	2ADM A	2AFG A	2AK3 A

2ALR A	2ANT I	2APR A	2ARC A	2ASI A	2ASR A	2ATC B	2ATJ A	2AYH A	2BAA A	2BB2 A	2BBK H
2BBK L	2BBM B	2BBV A	2BGU A	2BNH A	2BOP A	2BPA 1	2BPA 2	2BPA 3	2BTF A	2CBA A	2CBP A
2CCY A	2CDX A	2CGR H	2CHB D	2CHR A	2CHS A	2CMD A	2CND A	2CRO A	2CST A	2CTC A	2CTX A
2CY3 A	2CYP A	2DGC A	2DKB A	2DLD A	2DNJ A	2DRI A	2DRP A	2DTR A	2EBN A	2ECH A	2END A
2ENG A	2EOL A	2ERL A	2ETI A	2EZD A	2EZH A	2EZK A	2FB4 H	2FB4 L	2FBJ H	2FBJ L	2FCR A
2FGW H	2FHA A	2FX2 A	2GDM A	2GF1 A	2GLI A	2GMF A	2GSA A	2GSO A	2GSR A	2H1P H	2HFT A
2HHM A	2HIP A	2HLC A	2HMO A	2HNT C	2HPD A	2HPP P	2HPO P	2HRP H	2HRP L	2HTS A	2HVM A
2IFO A	2ILK A	2IMN A	2JXR A	2KNT A	2LBD A	2LBP A	2LDX A	2LEU A	2LHB A	2LIV A	2LTN A
2LTN B	2MAS A	2MCG 1	2MCM A	2MEV 1	2MEV 2	2MEV 3	2MHR A	2MHU A	2MIN A	2MLT A	2MPR A
2MRB A	2MSB A	2MTA C	2MYS A	2NAC A	2NCM A	2NLL A	2NLL B	2OHX A	2OMF A	2PEL A	2PF2 A
2PGD A	2PGH A	2PGH B	2PHL A	2PHY A	2PIA A	2PII A	2PKA A	2PKA B	2PKC A	2PLC A	2PLD A
2PLH A	2PLT A	2PNA A	2POL A	2POR A	2PRD A	2PSP A	2PTA A	2PTD A	2PTH A	2PTL A	2RAN A
2RBI A	2REB A	2RHE A	2RMC A	2RN2 A	2RSL A	2RSP A	2SAK A	2SAS A	2SCP A	2SEB B	2SFA A
2SGA A	2SIC I	2SIL A	2SN3 A	2SNS A	2SPC A	2STT A	2TBS A	2TBV A	2TCT A	2TGI A	2TMD A
2TMV P	2TRC P	2TRX A	2TS1 A	2TYS A	2TYS B	2U1A A	2UCZ A	2VAA A	2VHB A	2VIK A	2VPF B
2WBC A	2YHX A	351C A	3ADK A	3BCT A	3BTO A	3C2C A	3CHY A	3CLA A	3CMS A	3COX A	3CYR A
3DFR A	3FIB A	3FIV A	3GAR A	3GEO A	3GPD R	3GRS A	3IL8 A	3INK C	3LAD A	3LCK A	3LDH A
3LIP A	3LZT A	3MDD A	3MIN A	3MIN B	3MRA A	3NUL A	3OVO A	3P2P A	3PCH A	3PCH M	3PFK A
3PMG A	3PSG A	3PTE A	3RNT A	3RP2 A	3RUB S	3SDH A	3SDP A	3SIC I	3TGL A	3TSS A	3ULL A
4AAH A	4AAH B	4BP2 A	4CPA I	4CPV A	4DPV Z	4ECA A	4FXC A	4GAT A	4GPD 1	4HB1 A	4HTC I
4KBP A	4MDH A	4MT2 A	4PGA A	4PGM A	4RHN A	4SBV A	4SGB I	5CSM A	5CYT R	5EAS A	5HPG A
5LDH A	5NUL A	5P21 A	5PAL A	5PTI A	5PTP A	5RUB A	5ZNF A	6CEL A	6FAB H	6FAB L	6FD1 A
6GSV A	6LDH A	6RLX A	6RLX B	6RXN A	6TAA A	7AAT A	7AHL A	7CAT A	7FAB H	7PCY A	7RSA A
7TIM A	8ABP A	8ACN A	8DFR A	8FAB A	8FAB B	8I1B A	8RUC A	8RUC I	8RUC K	8RXN A	8TLN E
9LDT A	9PCY A	9RNT A	9WGA A	1VLB A	1A8O A	1DYZ A	1XWL A	1BVT A	1IR1 S	1BKJ A	1EU1 A
1P2Z A	1R36 A	1DUR A	1XFF A	2DOT H	2DOT L	2IFE A	1B1U A	3ENI A	1N4Y A	2LEF A	1GH1 A
2SBA A	1O7B T	1KA5 A	1XX2 A	1M85 A	1IOZ A	1E70 M	2BUK A	1OCO A	1CYO A	3EOJ A	1IG5 A

7.3 Postupak

Na DVD mediju priloženom uz rad nalaze se korišteni programi, baze podataka, podaci o proteinskim lancima te skripte izrađene u Perl programskom jeziku sa pripadnim objašnjenjima. U nastavku će ukratko biti opisan postupak kojim su ostvareni dobiveni rezultati. Korišteni programi nalaze se na DVD-u u direktoriju *'Programi*' a korištene skripte u direktoriju '*Perl'*.

1. Zapisi o proteinima pojedinih skupova, dohvaćeni sa PDB poslužitelja, nalaze se na DVD-u u *PDB* direktoriju u obliku *.pdb* datoteke za svaki od proteina. DSSP zapisi o sekundarnoj strukturi dobivaju se korištenjem DSSP programa te *pdb2dssp.pl* skripte; navedenu skriptu i *DsspSMBI.exe* izvršnu datoteku DSSP programa potrebno je postaviti u *PDB* direktorij te zatim pokrenuti skriptu. Po izvršavanju skripte dobiven je *'DSSP*' pod-direktorij sa pripadnim *.dssp* definicijama proteina pripadnih skupova.
2. Potrebne informacije o aminokiselinskim ostacima proteina sadržane su u PDB i DSSP zapisima, pa se za njihovo pročišćavanje koristi *parseDSSP.pl* skripta koju je potrebno smjestiti na isto mjesto kao u prethodnom koraku, *PDB* direktorij. Nakon izvršavanja skripte, pročišćeni podaci nalaze se u *'sekvence'* pod-direktoriju za svaki od skupova podataka u obliku tekstualnih datoteka sa *.txt* nastavkom.
   DSSP zapisi sadrže informacije o pojedinim proteinima koji se mogu sastojati od više proteinskih lanaca. Zato je broj generiranih tekstualnih datoteka veći od broja DSSP zapisa i svakoj je pridijeljeno ime koje se sastoji od izvorne oznake proteina uz dodanu informaciju oznake lanca. Npr. protein *1A0T* definiran istoimenim PDB (*1A0T.pdb*) odnosno DSSP zapisom (*1A0T.dssp*) sastoji se od tri lanca, pa se dobivaju tri datoteke za svaki lanac – *1a0T_P.txt, 1a0T_Q.txt, 1a0T_R.txt*.
3. Dobiveni skup tekstualnih datoteka čine svi proteinski lanci definirani DSSP zapisima. Pojedini skupovi proteina, međutim, ne sadrže sve proteinske lance, od kojih su neki i isti, pa je suvišne potrebno izdvojiti. Popis proteinskih lanaca uključenih u pojedini skup nalazi se u tekstualnoj datoteci oblika `'naziv_skupa.txt'` (npr. *RS126.txt*), u istom direktoriju gdje se nalaze i Perl skripte. Istoimene Perl skripte (npr. *RS126.pl*) iz skupa generiranih pročišćenih zapisa , a na temelju spomenutih tekstualnih datoteka izdvajaju potrebne proteinske lance u zaseban direktorij. Datoteku sa popisom proteina i istoimenu Perl skriptu potrebno je premjestiti u direktorij u kome se nalaze pročišćeni zapisi (u ovom slučaju */PDB/sekvence/naziv_skupa*) i od tamo ih izvršiti. Konačni skupovi proteinskih lanaca nalaze se u pod-direktoriju *'izdvojeno'*.
4. Generiranje profila slijeda za svaki od proteinskih lanaca vrši se *2PSSM.pl* Perl skriptom pri čemu je prethodno potrebno instalirati PSI-BLAST program a NR bazu raspakirati u *PDB* direktorij, gdje je potrebno smjestiti i navedenu skriptu.

Po izvršenju skripte stvoren je novi direktorij '*PSSM'* koji sadrži profile slijeda svih potrebnih proteinskih lanaca.

5. Konačni korak je izrada ARFF datoteka uz pomoć *2ARFF.pl* odnosno *2ARFF_2.pl* skripti koje se također pokreću iz osnovnog, '*PDB'* direktorija. *2ARFF.pl* skripta proizvodi ARFF datoteke za klasifikaciju u tri klase, dok *2ARFF_2.pl* generira podatke za binarnu klasifikaciju. Obje skripte se pozivaju sa dodatnim argumentima na sljedeći način:

   ```
   >> 2ARFF.pl [-l]
   >> 2ARFF_2.pl [-l] -S
   ```

   '*l*' je opcionalan parametar koji definira duljinu okvira, a '*S*' nužan parametar za binarnu klasifikaciju koji postavlja željenu sekundarnu strukturu a može poprimiti jednu od sljedećih vrijednosti: *'H'*, *'E'* ili '*C*'. Ukoliko se '*l*' parametar izostavi, pretpostavljena duljina okvira iznosi 9.
6. Dobivene ARFF datoteke sadrže konačni skup informacija o pojedinim skupovima i dalje se uz proizvoljne parametre mogu koristiti sa Rattle R programom za (binarnu) klasifikaciju aminokiselinskih ostataka.