Analiza molekulskih deskriptora u kvantitativnim odnosima strukture i topljivosti tvari

Sindičić Đuretec, Valnea

SVEUČILIŠTE U ZAGREBU
FAKULTET KEMIJSKOG INŽENJERSTVA I TEHNOLOGIJE
SVEUČILIŠNI DIPLOMSKI STUDIJ

Valnea Sindičić Đuretec

DIPLOMSKI RAD

Zagreb, rujan 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET KEMIJSKOG INŽENJERSTVA I TEHNOLOGIJE
SVEUČILIŠNI DIPLOMSKI STUDIJ

Valnea Sindičić Đuretec

ANALIZA MOLEKULSKIH DESKRIPTORA U KVANTITATIVNIM ODNOSIMA STRUKTURE I TOPLJIVOSTI TVARI

DIPLOMSKI RAD

Voditelj rada: izv. prof. dr. sc. Šime Ukić

Članovi ispitnog povjerenstva:

izv. prof. dr. sc. Šime Ukić
dr. sc. Mirjana Novak Stankov
izv. prof. dr. sc. Marijana Kraljić Roković

Zagreb, rujan 2019.

Ovaj diplomski rad izrađen je na istraživačkom institutu KNOW-CENTER, Austrija pod vodstvom Marija Lovrića, mag. appl. chem, te Zavodu za analitičku kemiju Fakulteta kemijskog inženjerstva i tehnologije Sveučilišta u Zagrebu pod mentorstvom izv. prof. Šime Ukića.

Htjela bih se zahvaliti svojem mentoru izv. prof. Šimi Ukiću, na pomoći oko diplomskog rada, razumijevanju i prihvaćanju mojih želja istraživanja, stručnim savjetima i brzim odgovorima na sva moja pitanja.

Hvala mojem neposrednom voditelju Mariju Lovriću, mag. appl. chem., na pomoći i savjetima tijekom pisanja ovog rada. Također hvala na pruženoj prilici i svakom dobronamjernom prijedlogu.

Hvala mojim roditeljima, Mami i Tati, što su bili uvijek uz mene. Vjerovali ste u mene uvijek, čak i kada sama nisam.

Hvala svim mojim prijateljima na podršci, slušanju i razumijevanju kroz zajedničke godine studiranja.

Također veliko hvala mojem mužu što je uvijek vjerovao u mene i bio iznimna podrška.

SAŽETAK

Upotreba kvantitativnih odnosa strukture i svojstava tvari (QSPR) jedna je od glavnih ekonomskih alternativa u eksperimentalnom testiranju molekulskih svojstava, kao što je topljivost tvari u vodi, log *S*. U ovom radu, izrađeni su QSPR modeli primjenjujući algoritme strojnog učenja, stabala odluke i nasumičnih šuma, koristeći programski jezik Python. Modeli su rađeni na setu od 1311 molekula; koristili su se različiti molekulski deskriptori i molekulski otisci. Usporedbom modela pokazalo se kako algoritam nasumičnih šuma ima bolju moć predviđanja od algoritma stabla odluke. Također se pokazalo da modeli koji koriste molekulske deskriptore imaju bolju moć predviđanja od onih koji koriste molekulske otiske. Kao najbolji izabran je model nasumičnih šuma uz uporabu molekulskih deskriptora.

Razvijeni model mogao bi poslužiti za predviđanje topljivosti molekula kandidata prilikom razvoja novih lijekova kako bi se smanjili troškovi eksperimentalnih ispitivanja.

Ključne riječi: QSPR, molekulski deskriptori, molekulski otisci, topljivost

SUMMARY

The use of quantitative structure-property relationship (QSPR) is one of the major economic alternatives in experimental testing of molecular properties, such as the aqueous solubility of a substance, log *S*. In this master thesis, QSPR models were developed using machine learning algorithms, decision trees and random forests. The modeling was performed using Python programming language. The models were developed on a set of 1311 molecules; different molecular descriptors and molecular fingerprints were used. Model comparisons showed that the random forest algorithm had better prediction power than the decision tree one. Models using molecular descriptors indicated better prediction power than those using molecular fingerprints. The model of random forests using molecular descriptors was the one with the best characteristics.

Once developed, the QSPR solubility model could be applied in developing process of new drugs in order to reduce the cost of experimental trials.

Key-words: QSPR, molecular descriptors, molecular fingerprints, solubility

SADRŽAJ

1. UVOD 1

2. TEORIJSKI DIO 2
2.1. TOPLJIVOST TVARI 2
2.1.1. Klasifikacija tvari prema topljivosti i permeabilnosti 3
2.1.2. Određivanje topljivosti 3
2.1.3. Računalno modeliranje topljivosti 4
2.2. QSPR MODELIRANJE 4
2.3. STANDARDIZACIJA MOLEKULSKIH STRUKTURA 6
2.4. MOLEKULSKI DESKRIPTORI 8
2.5. MOLEKULSKI OTISCI 10
2.6. METODE UMJETNE INTELIGENCIJE 11
2.6.1. Umjetne neuronske mreže 12
2.6.2. Neizravna logika 13
2.6.3. Strojno učenje 14
2.6.4. Izrada modela 14
2.6.4.1. Nadzirano učenje 14
2.6.4.2. Učenje bez nadzora 15
2.6.4.3. Regresijski algoritmi 16
2.7. PYTHON 18
2.7.1. Programske biblioteke 19
2.7.1.1. Pandas 19
2.7.1.2. NumPy 19
2.7.1.3. Scikit-Learn 20
2.7.1.4. Matplotlib 20
2.7.1.5. Seaborn 20
2.7.1.6. RDKit 20

3. MODELIRANJE 21
3.1. POSTUPAK STANDARDIZACIJE 21
3.2. RAČUNANJE MOLEKULSKIH DESKRIPTORA 23

3.3. RAČUNANJE MOLEKULSKIH OTISAKA 24
3.4. IZRADA MODELA 25

4. REZULTATI I RASPRAVA 28
4.1. MOLEKULSKI DESKRIPTORI 28
4.2. MOLEKULSKI OTISCI 35

5. ZAKLJUČAK 41

6. LITERATURA 42

7. ŽIVOTOPIS 46

8. PRILOZI 47
8.1. PRILOG 1. PROGRAMSKI KOD ZA MODELIRANJE POMOĆU MOLEKULSKIH DESKRIPTORA 47
8.2. PRILOG 2. PROGRAMSKI KOD ZA MODELIRANJE POMOĆU MOLEKULSKIH OTISAKA 51

1. UVOD

Postupak otkrivanja i razvoja lijekova je izazovan, dugotrajan i skup proces. Prihvatljivije rješenje ovog problema je računalno dizajniranje lijekova koje se koristi različitim metodama i postupcima, a koje omogućuje identifikaciju odgovarajuće molekule prije njezine sinteze. Kroz povijest su se predlagali i razvijali brojni pristupi i različite metode sa svrhom olakšanog pronalaska ljekovite supstancije. QSAR metoda (kvantitativni odnosi strukture tvari i njene aktivnosti, engl. *Quantitative Structure-Activity Relationship*) je jedna od najpopularnijih metoda kojom se farmaceutska industrija danas služi u pronalasku novih ljekovitih tvari željenih karakteristika. Metode se zasniva na određivanju povezanosti kemijske strukture spoja i njegova određenog svojstva. Stoga je potrebno strukturu molekule opisati matematički: kroz niz brojčanih veličina koje nazivamo deskriptorima.[1]

Topljivost je jedan od najznačajnijih parametara koji utječe na postizanje željene koncentracije oralnog lijeka u sistemskoj cirkulaciji.

2. TEORIJSKI DIO

2.1. TOPLJIVOST TVARI

Topljivost je svojstvo krute, tekuće ili plinovite kemijske tvari, koja se naziva otopljena tvar, da se otopi u krutom, tekućem ili plinovitom otapalu te da se dobije homogena otopina. Ovisi o korištenom otapalu, temperaturi i tlaku. Stupanj topljivosti tvari u otapalu se mjeri kao koncentracija zasićenja kada daljnje dodavanje otopljene tvari ne povećava njegovu koncentraciju u otopini.[2]

Otapalo je najčešće tekućina, koja može biti čista tvar ili smjesa sviju tekućina. Također postoje i čvrste otopine, ali rijetko postoje i plinske otopine. Stupanj topljivosti ima širok raspon, od beskrajno topljivog (potpuno miješan), kao što se etanol miješa u vodi, do slabo topljivog, kao što je srebro klorid u vodi. Izraz netopljiv često se primjenjuje na slabo ili vrlo slabo topljive spojeve.[3]

U farmaceutskoj industriji je topljivost vrlo važan fizikalno-kemijski parametar, te je topljivost tvari u vodi veoma važan parametar kada se odabiru spojevi u razvoju lijekova. Igra veliku ulogu u ADMET[1] svojstvima, kao što su oralna apsorpcija, distribucija lijeka u tijelu i bio raspoloživost lijeka. Štoviše, topljivost spojeva je posebno relevantna u probnim testiranjima budući da je netopljivost i agregacija molekula glavni krivac za neuspjeh u prvim istraživanjima lijeka kandidata. Obično se topljivost procjenjuje u vrlo ranim fazama otkrivanja lijeka.[4]

Otkriće novih lijekova zahtijeva identificiranje visokokvalitetnih kandidata za lijekove s odgovarajućim fizikalno-kemijskim svojstvima. Lijekovi koji se primjenjuju oralno općenito zahtijevaju dovoljnu topljivost u vodi da bi bili bio-raspoloživi. Nepoznate molekule lijeka koje su netopljive u vodi ne mogu se testirati u biološkim testovima, imaju loše farmakološke profile i mogu izaći iz lijeka pod uvjetima skladištenja istog. Loša topljivost i slaba farmakokinetika u fiziološkim uvjetima često mogu rezultirati neuspjehom u otkrivanju lijeka sa skupim posljedicama.[5]

[1] ADMET je kratica u farmakokinetici i farmakologiji za apsorpciju, distribuciju, metabolizam, izlučivanje i toksičnost te opisuje dispoziciju farmaceutskog spoja unutar organizma

2.1.1. Klasifikacija tvari prema topljivosti i permeabilnosti

Biofarmaceutski sustav klasifikacije djelatnih tvari (engl. *Biopharmaceutics Classification System*; BCS) omogućio je lakšu regulaciju oralnih lijekova. Smatra se vodičem za predviđanje intestinalne apsorpcije djelatne tvari uz korištenje dvaju parametara, topljivosti i permeabilnosti. U BCS sustavu djelatne tvari su klasificirane u četiri kategorije odnosno četiri biofarmaceutske skupine na temelju njihove permeabilnosti kroz membrane probavnog sustava i topljivosti u vodi; I. BCS skupina (dobro permeabilne i dobro topljive); II. BCS skupina (dobro permeabilne i slabo topljive); III. BCS skupina (slabo permeabilne i dobro topljive); IV. BCS skupina (slabo permeabilne i slabo topljive). Djelatna tvar se smatra dobro topljivom ukoliko se najveća doza djelatne tvari otapa u 250 ml ili manje vodenog medija u pH području 1,2-6,8 pri 37 °C.[6]

2.1.2. Određivanje topljivosti

Na topljivost tvari utječe mnogo faktora. Osim prirode otapala, temperature i tlaka postoji nekoliko drugih čimbenika koji negativno utječu na topljivost tvari. Neki od njih su močivost, stvaranje agregata i micela, polimorfnost i vrijeme dostizanja ravnoteže.

Kod slabo topljivih tvari može doći do loše močivosti što se očituje na način da tvar ″lebdi″ na površini otapala ili se prima za stijenke posuđa što dovodi do lošijeg i sporijeg otapanja istih. Stvaranje agregata i/ili micela uočava se kod vrlo slabo topivih krutina te komplicira interpretaciju mjerenja topljivosti, pogotovo ako su molekule ionizirane. Tijekom mjerenja topljivosti, tvar kojoj se određuje topljivost može preći u stabilniji polimorf stoga je važno izolirati krutinu na kraju provedbe mjerenja kako bi se utvrdilo da je isti oblik prisutan na početku i na kraju mjerenja.

Za određivanje pH-zavisne topljivosti u vodenim otopinama postoji nekoliko metoda. Najpoznatija i najjednostavnija je metoda tresilicom (engl. *Saturation Shake-Flask*, SSF) u kojoj se uzorak dodaje u tikvice u suvišku i miješa do postizanja ravnotežnog stanja odnosno do zasićenja. Topljivost uzorka u otopini određuje se nakon filtracije otopine pomoću tekućinske kromatografije. Također zbog jednostavnosti u praksi se često koristi i potenciometrijska metoda u kojem se uzorak titrira kiselinom ili bazom 0,15 M otopinom KCl uz propuhivanje argonom zbog smanjenja utjecaja CO_2 na pH otopine. pH se mjeri staklenom elektrodom. Prilikom mjerenja dolazi do taloženja uzorka što se očituje pomakom titracijske

krivulje. Razlika u pomaku krivulje između zasićenog i nezasićenog analita predstavlja topljivost.

Osim navedenih metoda koristi se još nekoliko njih, kao što su: minijaturizirana tresilica (engl. *Miniaturized shake-flask,* MSF), UV-mikropokretljivost (engl. *Self-Calibrating direct UV microsolubility*, μSOL), turbidimetrija, metode sa ili bez DMSO-a i metoda mikro otapanja (engl. *micro dissolution apparatus for solubility measurement*, μDISS).[7]

2.1.3. Računalno modeliranje topljivosti

Pošto su eksperimentalna ispitivanja svojstava molekula dugotrajan i veoma skup proces nastala je potreba za računalnim modeliranjem istih. Kako bi se izbjegli neupotrebljivi kandidati među molekulama s potencijalnim djelovanjem korisno je unaprijed predvidjeti fizikalno – kemijske parametre.

Cilj modeliranja je na temelju poznatih strukturalnih parametara i/ili mjerenih fizikalnih veličina predvidjeti tražena svojstva. Kao podloga modeliranja služi teorijia da je otapanje posljedica dvaju svojstava:

1. jakosti interakcije između otapala i otopljene tvari (npr. polarnost, vodikove veze i slično)
2. afinitetu topljive tvari prema samoj sebi (npr. sile unutar kristalne rešetke).

U računalnom modeliranju postoji nekoliko različitih metoda kao što su: empirijske i semiempirijske metode, simulacijske metode i kvantno – mehaničke metode.

2.2. QSPR MODELIRANJE

QSPR modeliranje dolazi od engleskog termina *Quantitative Structure-Property Relationship* (kvantitativni odnos strukture i svojstva). To je pristup koji strukturna svojstva molekula želi dovesti u kvantitativni odnos sa svojstvima molekula. Danas se u velikoj mjeri primjenjuje u širokom rasponu znanstvenih disciplina, uključujući kemiju, biologiju i toksikologiju, a najširu primjenu ima u medicinskoj kemiji i prediktivnoj toksikologiji.[8] Posljednjih deset godina QSPR je neizostavan alat u farmaceutskoj industriji, a rastući trend je rana primjena QSPR-a kao sredstva provjere u postupku otkrivanja ljekovite supstance te kao alata za eliminaciju molekula kojima nedostaje određeno svojstvo ili onih za koje se predviđa da će izazvati toksičan odgovor iz daljnjeg razvoja. Takav postupak razvoja QSPR-a

omogućuje njegovo širenje izvan okvira farmaceutske industrije do ekoloških i ljudskih regulatornih tijela za upotrebu u toksikologiji.[9,10]

Prvi korak korištenja metode QSPR podrazumijeva pronalazak grupe kemijskih spojeva ili vodećeg spoja koji pokazuje željeno svojstvo. Zatim se uspostavlja kvantitativni odnos između fizičko-kemijskih parametara i svojstva nakon čega se, pomoću QSPR modela, provodi optimizacija aktivnih spojeva. Odgovarajući se spojevi nakon uspostavljanja kvalitetnog matematičkog modela, eksperimentalno testiraju na željeno svojstvo. Eksperiment je nužan za potvrđivanje ispravnosti i točnosti dobivenog matematičkog modela. Na ovaj se način QSPR metoda može upotrijebiti kao vodeće sredstvo za određivanje odgovarajućih izmjena u vodećem spoju kojima bi se poboljšala njegova svojstva. Primijenjena promišljeno, ova metoda može uštedjeti znatnu količinu vremena, novca i ljudskih resursa.[11]

Kvaliteta QSPR modela ovisi o:

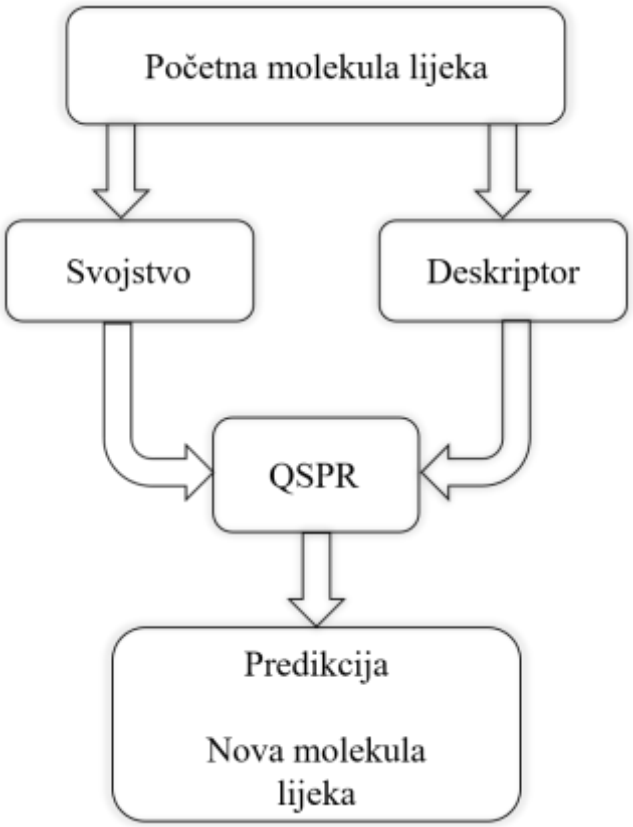
- kvaliteti podataka bioloških mjerenja
- odabiru kemijskih spojeva za izgradnju modela
- odabiru strukturnih deskriptora molekula

Statistički model je vjerodostojniji ako je što veći broj molekula uključen u izgradnju samog modela. Ponekad je bolje izgraditi model bez nekolicine molekula koje ne zadovoljavaju navedene faktore kako bi se zadržala mogućnost predviđanja, odnosno olakšala mogućnost nalaženja korelacije strukturnih svojstava. Kako bi model bio vjerodostojniji, poželjno je izgraditi model od skupa molekula koje imaju isti kostur, a razlikuju se samo u određenim fragmentima. Ipak, problem u tom slučaju je što model nije robustan te je primjenjiv samo za sličan skup molekula. Ukoliko se želi izgraditi robusniji model, potrebno je uzeti različite skupove molekula te dovoljan broj molekula iz svakog skupa kako bi se zadržala smislenost modela.[12]

Osnovni koraci QSPR modeliranja mogu se sažeti na sljedeći način:

1. Aktivne molekule koje se vežu na željeni cilj lijeka i njihova aktivnost identificiraju se pretraživanjem baze podatka, pretraživanjem literature ili eksperimentalnim radom
2. Identifikacija strukturnih ili fizikalno-kemijskih karakteristika molekula (molekulski otisak prsta) koji utječu na biološku aktivnost (npr. Vrsta veze, vrsta atoma, broj funkcionalnih skupina, površina itd.)
3. Izgradnja QSPR-a između biološke aktivnosti i identificiranih svojstava molekula lijeka

4. Provjeravanje prediktivne snage za QSPR biološku aktivnost
5. Korištenje QSPR modela za optimiziranje poznatih aktivnih spojeva kako bi se maksimizirala biološka aktivnost
6. Nove optimizirane molekule lijeka testiraju se eksperimentalno.[13]



Slika 1. Shematski dijagram koji prikazuje korake uključene u QSPR.[13]

2.3. STANDARDIZACIJA MOLEKULSKIH STRUKTURA

Kemijske strukture su gotovo uvijek prikazane kao slike te su u takvom obliku informatički nečitljive. Takav način otežava njihovo uvrštavanje u baze podataka. Kako bi se kemijska struktura, koja je u slikovitom prikazu, mogla uvrstiti u bazu podataka potrebno je strukturu pretvoriti u SMILES zapis. SMILES (engl. *Simplified molecular input line entry system*)[14] je specifikacija u obliku linijske notacije za opisivanje strukture kemijske

molekule koristeći kratke ASCII[2] kodove. SMILES kodovi se mogu koristiti u većini računalnih programa za lakše pretvaranje molekula u dvodimenzionalne crteže ili trodimenzionalne modele molekula. Glavna pravila za generiranje ispravnog SMILES zapisa su sljedeća:

1. Atomi su predstavljeni njihovim simbolima u uglastim zagradama. Neki od elemenata, kao što su B, C, N, P, S, F, Cl, Br, i I, mogu biti napisani i bez zagrada.
2. Pretpostavlja se da su susjedni atomi međusobno povezani pa su jednostruke, dvostruke, trostruke“ i aromatske veze predstavljene znakovima „-“, „=“, „#“ te „:“.
3. Aromatičnost se može prikazati malim slovima.
4. Grane molekule prikazuju se u zagradama. Kao dobar primjer može poslužiti molekula acetona koja se prikazuje kao CC(=O)C.
5. Ciklične strukture se prikazuju razbijanjem jedne veze u svakom prstenu. Veze su numerirane u bilo kojem redoslijedu, označavajući veze otvaranja (ili zatvaranja prstena) pomoću brojeva koji odmah slijedi atomski simbol na svakoj prstenovoj površini. Dobar primjer je benzen koji se prikazuje kao c1ccccc1.
6. Konfiguracija oko dvostruke veze prikazuje se uz pomoć znakova „/“ i „\“.
7. Konfiguracija oko tetraedarskih centara prikazuje se sa @ ili @@ kao atomsko svojstvo slijedeći atomski simbol kiralnog atoma unutar uglatih zagrada, a temelji se na redoslijedu u kojem se susjedni atomi pojavljuju u kiralnom centru, simbol @ označava da se druga tri susjedna atoma pojavljuju u smjeru suprotnom od kazaljke na satu redoslijedom kojim su navedeni u zapisu. @@ označava da su susjedni atomi zapisani u smjeru kazaljke na satu.[15]

Ovakav zapis molekula može generirati mnogo pogrešaka zbog čega je obavezno provesti proces standardizacije. Sam postupak standardizacije ovisi o softveru koji će se koristiti u obradi podataka.

[2] ASCII – Američki standardni kod za razmjenu informacija (engl. *American Standard Code for Information Interchange*)

2.4. MOLEKULSKI DESKRIPTORI

Molekulski deskriptori su različiti kompjuterski ili eksperimentalno izvedeni kvantitativni parametri uz pomoć kojih se QSAR model može opisati matematičkim jednadžbama koje mogu povezivati svojstva molekula kao što su fizikalno-kemijska svojstva, biološka svojstva ili toksikološka svojstva. Deskriptori su povezani s eksperimentalnim svojstvima koristeći različite kemometrijske alate da bismo dobili statistički značajan QSAR model.[16] Oni su rezultat numeričkih i logičkih transformacija kemijske informacije povezane s kemijskim sastavom za usporedbu kemijske strukture s različitim fizikalnim svojstvima, kemijskom reaktivnošću ili biološkom aktivnošću. Prema tome svaki odziv kemijskog spoja (aktivnost/ svojstvo/ toksičnost) može biti matematički izveden kao funkcija deskriptora.[17]

Tablica 1. Podjela deskriptora s obzirom na opisujuća svojstva[18]

Tip deskriptora	Svojstvo
Konstitucijski deskriptori	molekulska masa, broj atoma, broj veza, broj prstenova, lipofilnost
Topološki deskriptori	Weinerov indeks, Randićev indeks, Kierove i Hallove značajke, informacijski sadržaj, indeks povezanosti, Balabanov indeks
Elektrostatski deskriptori	parcijalni naboji, indeks polarnosti; topološki elektronski indeks, multipolovi, djelomično elektronski nabijena molekulska površina, polarnost, anizotropija polarnosti
Geometrijski deskriptori	moment inercije, molekulski volumen, molekulska površina, indeksi zasjenjivanja, Taftova konstanta steričnosti, parametri duljine, širine i visine, faktor oblika
Kvantno-mehanički deskriptori	mreža atomskih naboja, red veze, HOMO i LUMO energije, FMO indeksi reaktivnosti, refrakcije, ukupna energija, ionizacijski potencijal, elektronski afinitet, energija protoniranja, orbitalna populacija, granica orbitalne gustoće, superdelokalizacije

Mogu se podijeliti prema načinu dobivanja na eksperimentalne i teorijske deskriptore. Eksperimentalni molekularni deskriptori predstavljaju rezultat nekog standardiziranog pokusa, a opisuju fizička svojstva molekula kao što su molekulska masa, lipofilnost (log *P*), dipolni moment, topljivost… Teorijski molekularni deskriptori pretvaraju strukturu u koristan broj te se temelje na simboličkom prikazu molekule, te su oni dobivali imena prema autoru koji ih je predložio. Neki od primjera teorijskih deskriptora su Balabanov indeks, Radićev indeks, Zagrebački indeks i dr. Izračunavaju se upotrebom precizno određenih algoritama,

dok se izvode primjenom načela različitih znanstvenih disciplina kao što su kvantna kemija, organska kemija, informacijske teorije i dr. Glavna razlika između ove dvije vrste molekularnih deskriptora je ta što teorijski deskriptori ne sadrže statističku pogrešku uzrokovanu šumom eksperimentalnih postupaka.[18] U tablici 1 prikazana je podjela molekularnih deskriptora s obzirom na opisujuća svojstva.

Konstitucijski deskriptori predstavljaju fizička i kemijska svojstva nekog spoja. Posebno treba istaknuti lipofilnost (log *P*) između oktan-1-ola i vode (fosfatni pufer, pH = 7,4). Njegovim otkrićem je započela intenzivna primjena metode QSAR, a i danas je jedan od najvažnijih molekularnih, fizičko-kemijskih deskriptora u QSAR-u.

Topološki deskriptori u strukturi molekule vide samo mogućnost povezivanja atoma. Oni numerički izražavaju topološku informaciju za neki spoj te se mogu direktno upotrijebiti kao jednostavni deskriptori za uspoređivanje s fizičkim, kemijskim ili biološkim parametrima molekule u QSAR-u. Njih čine tzv. topološki indeksi: Wienerov, Randićev, Balabanov i mnogi drugi.[19]

Geometrijski i elektronski deskriptori opisuju raspodjelu naboja u molekuli. Mogu se izračunati molekularno-mehaničkim i kvantno-kemijskim proračunima. Nakon otkrića tih deskriptora pojavila se i nova inačica metode QSAR – 3D QSAR, koja je utemeljena na tim deskriptorima.[20]

Potrebno je napomenuti da se odabir određenog molekularnog deskriptora ili većeg broja deskriptora temelji na njihovoj mogućnosti, odnosno jednostavnosti interpretacije. Pri odabiru treba voditi računa da odabrani skup molekularnih deskriptora sadrži one koji ne nose istu informaciju i koji su međusobni slabo korelirani.

Prema dimenzionalnosti strukturne informacije molekularni deskriptori dijele se na:

- 0D – opisuju svojstva molekula temeljem sažete molekulske formule, npr. broj atoma,
- 1D – opisuju svojstva supstituenata, fragmenata, funkcionalnih skupina,
- 2D – opisuju veze između atoma u molekuli – topološki indeksi,
- 3D – opisuju konfiguraciju molekule – geometrijski, sterički i volumni deskriptori.[21]

2.5. MOLEKULSKI OTISCI

Pod pojmom molekulskih otisaka (engl. *fingerprints*) krije se prikaz kemijskih struktura izvorno osmišljen kako bi se pomoglo u pretraživanju velikih baza kemijskih podataka, a koristi se i za različite analize kao što su usporedba sličnosti molekula, grupiranje i klasifikaciju.[22] To su nizovi bitova (sekvence nula i jedinica) kojima se kodiraju prisutne pod-strukture molekula. Svaka sekvenca atoma i njihovih međusobnih kemijskih veza aktivira određeni broj bitova. Broj bitova koji su aktivirani svakom strukturom su parametar molekulskog otiska. Uz pomoć algoritama moguće je odrediti koji bitovi su aktivirani određenim sekvencama atoma ili kemijskih veza.[15]

Generiranje molekulskog otiska uključuje nekoliko koraka:

1. Inicijalizacija svih bitova otiska s vrijednošću 0
2. Nabrajanje svih uzoraka koji predstavljaju svaku skupinu atoma i veza povezanih do n broja veza (broj veza određuje sam korisnik)
3. Svaki uzorak služi za generiranje pseudo-slučajnih brojeva čiji je izlaz položaj bita u otisku. Bitovi zatim dobivaju vrijednost 1.

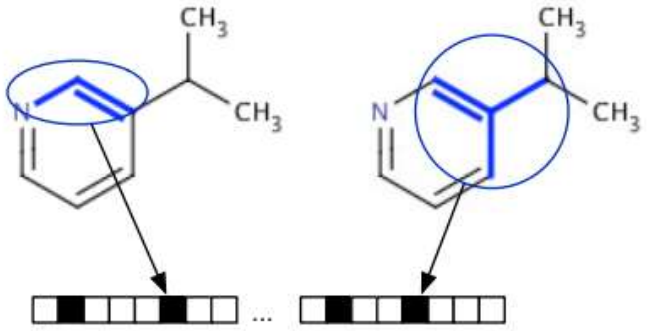
Kao primjer možemo uzeti molekulu alanina. Potrebno je poznavati SMILES zapis alanina koji glasi: CC(N)C(O)=O. Uzorci koji će dati vrijednost bita 0 su C, N i O, dok su uzorci koji će dati vrijednosti bita 1 su CC, CN, CO i C=O.

Isti uzorak uvijek aktivira isti set bitova u bilo kojoj molekuli, a algoritam može prepoznati svaki od njih. Unatoč svemu može doći do kolizije, tj. isti bit može biti aktiviran različitim uzorcima te struktura ne može biti prepoznata iz takvih otisaka. Općenito su atomi vodika izostavljeni iz generiranja otiska, a i stereokemija se ne uzima u obzir.

Prije generiranja otisaka za skupinu molekula , krajnji korisnik mora definirati veličinu otiska, maksimalnu veličinu uzoraka koje treba nabrojati i broj bitova koji se aktiviraju uz pomoć svakog uzorka. Ovi parametri utječu na potrebnu memoriju za pohranu otiska i njihovu sposobnost razlikovanja različitih molekula.[15]

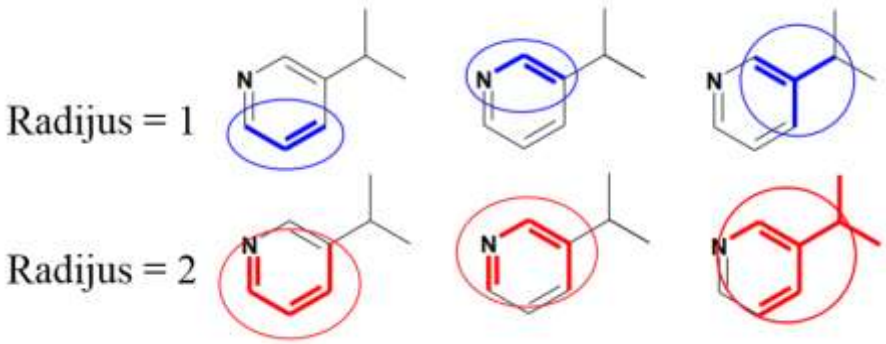
Ideja generiranja molekulskih otisaka je primijeniti softver na molekuli kako bi se generirao bit vektor. Bit vektor je linearna sekvenca numeričkih vrijednosti u kojoj je svaka vrijednost bit, odnosno vrijednosti 0 ili 1. Kako bi došlo do generacije ekstrahiraju se glavne značajke svake molekule, izdvajaju se i zatim se izdvojeni dijelovi molekule koriste za određivanje bitova koji će biti u setu. Svaki otisak prsta odgovara fragmentu molekule kao što

je prikazano na slici 2. Pretpostavka je da molekule koje su slične imaju dosta zajedničkih otisaka prsta.



Slika 2. Prikaz značajki molekule kao otisak prsta u obliku bit vektora.[23] Plavo označeni dijelovi molekula su aktivirale jedinice (1) u bit vektoru.

Tipična veličina molekulskih otisaka je 1000 do 4000 bitova. Jedan od parametara generiranja otisaka prstiju je radijus kojim se određuje preko koliko veza od početnog atoma će se gledati neka značajka molekule. Tipičan radijus je od 1 do 5 veza. Na slici 3 prikazan je primjer kada su vrijednosti radijusa 1 i 2.[23]



Slika 3. Prikaz radijusa otiska prsta molekule[23]

2.6. METODE UMJETNE INTELIGENCIJE

Kemometrija je primjena statistike u analizi kemijskih podataka. To je dio kemije koji proučava i provodi primjenu matematičkih, statističkih i ostalih metoda zasnovanih na formalnoj logici.[24] Naziv je osmišljen 1970-ih, a u svojim se početcima uglavnom bavila obradbom podataka dobivenih instrumentalnim analitičkim tehnikama uključujući jednostavne izračune.[25] Danas su kemometrijski alati kudikamo napredniji, ponajviše

zahvaljujući naprednim mogućnostima koje je donio razvoj računarstva. Među najnaprednijim pristupima nalaze se metode umjetne inteligencije.

Umjetna inteligencija (engl. *artificial intlligence*, AI) je znanstveno polje koje obuhvaća računalne tehnike za izvršavanje zadataka za koje je potrebna inteligencija kada ih rješavaju ljudi. Takvi zadaci mogu biti: dijagnosticiranje bolesti, pisanje priča ili skladanje glazbenih djela, pronalaženje matematičkih teorema, sastavljanje i analiza proizvoda u tvornicama, pregovaranje i sklapanje međunarodnih ugovora te još puno toga.[26]

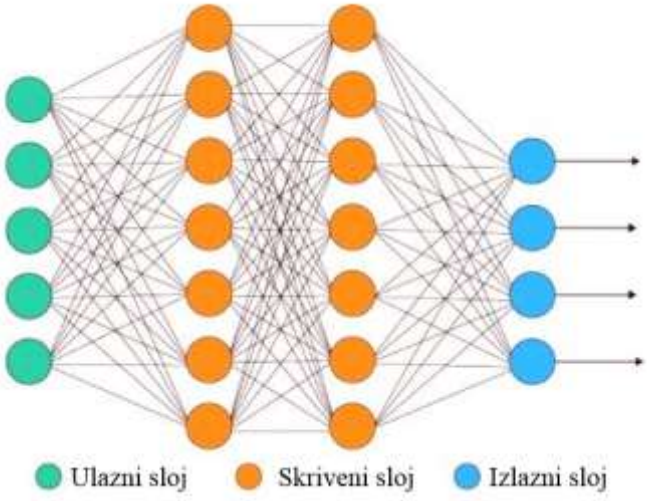
2.6.1. Umjetne neuronske mreže

Umjetne neuronske mreže (engl. *artificial neural networks*, ANN) su računalne mreže koje pokušavaju simulirati rad ljudskog živčanog sustava. Istraživanja i razvoj umjetnih neuronskih mreža motivirani su spoznajama o građi i načinu funkcioniranja ljudskog mozga te njegovim nevjerojatno velikim sposobnostima u rješavanju složenih problema. ANN omogućava rješavanje jednostavnih računalnih operacija poput zbrajanja, množenja, ali i složenih ili matematički loše definiranih problema. Konvencionalni algoritmi sastoje se od složenih setova jednadžbi koje se mogu primijeniti samo za zadani problem. ANN će računalno i algoritamski nerijetko biti jednostavnije, a imati će mogućnost bavljenja znatno širim spektrom problema. Primjerice, ako muha izbjegava prepreke u letu ili kada miš izbjegava mačku, oni zasigurno putem ne rješavaju diferencijalne jednadžbe niti koriste složene uzorke algoritama. Njihov mozak je jednostavan i zato ANN teži k toj jednostavnosti u rješavanju problema. Još jedan aspekt ANN-a koji ih razlikuje od običnih računala je njihova visoka paralelnost.[27]

Živčane stanice organizirane su u module (slojeve, engl. *layers*) i međusobno povezane u složenu mrežu s otprilike 10 međusobnih veza. Tako gusto povezana mreža živaca osigurava izuzetno veliku računsku i memorijsku moć ljudskog mozga. Živčana stanica, kao osnovna gradivna jedinica biološke neuronske mreže, prima i obrađuje informacije (signale) od drugih živčanih stanica i/ili osjetilnih organa.[28]

Analogno biološkim umjetnim mrežama, mogu se izraditi i ANN. Umjetna mreža se sastoji od jednostavnih procesnih jedinica koje komuniciraju slanjem signala jedna drugoj preko jedne (ili više) veza definiranih određenim težinskim koeficijentom (w_{nj}). Unutar neuronskog sustava korisno je razlikovati tri vrste slojeva neurona: ulazni sloj (engl. *input leyer*) koji sadrži neurone čija je dužnost primanje podataka (signala) iz okoline neuronske

mreže (izvana), izlazni sloj (engl. *output layer*) u kojem se nalaze neuroni koji šalju podatke iz neuronske mreže, i skriveni slojevi (engl. *hidden units*) kojih može biti jedan ili više, a nalaze se između ulaznih i izlaznih jedinica. Neuronska mreža u kojoj nema skrivenih slojeva smatra se jednoslojnom, a ukoliko skriveni slojevi postoje govorimo o višeslojnoj mreži.[29]



Slika 4. Shematski prikaz višeslojne umjetne neuronske mreže

2.6.2. Neizravna logika

Neizravna logika (engl. *Fuzzy Logic*, FL) je oblik logike u kojoj vrijednosti varijabli mogu biti bilo koji stvarni broj između 0 i 1, uključujući te vrijednosti. Koristi se za rukovanje konceptom parcijalne istine, gdje se vrijednost istine može kretati između potpune istine i potpune laži.[30] FL algoritam pomaže riješiti problem nakon razmatranja svih dostupnih podataka. Tada je potrebna najbolja moguća odluka za dani unos. FL imitira način odlučivanja kod čovjeka koji razmatra sve mogućnosti između digitalnih vrijednosti istine i laži.

Neke od važnijih značajki neizravne logike su:

1. Fleksibilna i jednostavna tehnika strojnog učenja
2. Pomaže u oponašanju logike ljudske misli
3. Može imati dvije vrijednosti koje predstavljaju dva moguća rješenja
4. Vrlo pogodna metoda za nesigurno ili približno obrazloženje
5. Shvaća zaključke kao proces širenja elastičnih ograničenja
6. Omogućuje izgradnju nelinearnih funkcija proizvoljne složenosti
7. Treba biti izgrađena uz potpuno vodstvo stručnjaka[31]

2.6.3. Strojno učenje

Strojno učenje (engl. *Machine Learning*) je područje računalne znanosti koja koristi statističke tehnike kako bi računalnim sustavima omogućila "učenje" s podacima, bez izričitog programiranja.

Strojno učenje dobro je za:

- probleme za koje postojeća rješenja zahtijevaju puno ručnog podešavanja ili dugačke popise pravila: jedan algoritam strojnog učenja često može pojednostaviti kod i bolje ga izvršavati
- kompleksne probleme za koje ne postoje dobra rješenja koristeći tradicionalni pristup: najbolje tehnike strojnog učenja mogu pronaći rješenja za takve probleme
- fluktuirajuće okruženje: sistem strojnog učenja može se prilagodit novim podacima
- dobivanje uvida u složene probleme i u velike količine podataka

2.6.4. Izrada modela

Sistemi strojnog učenja mogu biti klasificirani prema količini i vrsti nadzora koju dobivaju tijekom treninga. Postoje dvije bitne kategorije: nadzirano učenje i učenje bez nadzora, no moguće su i varijacije između ovih kategorija (polu nadzirano učenje).

2.6.4.1. Nadzirano učenje

U nadziranom učenju (engl. *Supervised Learning*), podaci koji se treniraju i šalju algoritmu uključuju željena rješenja, te se nazivaju oznakama (engl. *labels*). Tipični zadatci nadziranog učenja su klasifikacija i regresija. Neki od regresijskih algoritama mogu biti korišteni u klasifikaciji i obratno. Najvažniji algoritmi nadziranog učenja su:

- *k*-najbliži susjed (engl. *k-Nearest Neighbors*)
- Linearna regresija (engl. *Linear Regression*)
- Logistička regresija (engl. *Logistic Regression*)
- Metoda potpornih vektora (engl. *Support Vector Machines, SVM*)
- Stabla odluke (engl. *Decision Trees*)
- Nasumične šume (engl. *Random Forest*) (ansambl stabala odluke) i

- Neuronske mreže (engl. *Neural Networks*)

2.6.4.2. Učenje bez nadzora

U učenju bez nadzora (engl. *Unsupervised Learning*) svi podaci koji se treniraju nisu označeni. Cijeli sistem pokušava učiti bez učitelja.[32]

Analiza glavnih komponenti

Analiza glavnih komponenti (engl. *Principal Components Analysis*, PCA) predstavlja jednu od najjednostavnijih multivarijantnih tehnika. Ona se primjenjuje kada je veliki broj varijabli u skupu sličan, odnosno kada se dvije ili više varijabli odnose na istu dimenziju i kada ne pružaju nikakvu informaciju koja već nije obuhvaćena nekom drugom varijablom.[33] Osnovna ideja metode je redukcija dimenzionalnosti seta podataka koji sadrži velik broj međusobno povezanih varijabli, zadržavajući pri tome što je više moguće varijacija koje su prisutne unutar izvornog seta podataka. To je moguće postići transformiranjem izvornog seta podataka u set novih varijabli, tzv. glavnih komponenata (engl. *Principal Components*, PC) koje nisu međusobno korelirane. Glavne komponente su poredane na način da je u prvih ″nekoliko″ sadržano najviše varijanci prisutnih unutar svih izvornih varijabli.[34]

Analiza glavnih komponenti se izvodi u sljedećim koracima:

1. Vrši se standardizacija originalnih podataka tako da originalne varijable imaju aritmetičku sredinu jednaku nuli i varijancu jednaku jedinici. Ovaj korak se najčešće ne preskače iako ima slučajeva da se to čini kada se vjeruje da je važnost originalnih varijabli dobro iskazana kroz varijance.
2. Izračunava se matrica kovarijanci *C*.
3. Izračunavaju se svojstvene vrijednosti λ_1, λ_2, ..., λ_p i odgovarajući vektori $\alpha_1,\alpha_2,...,\alpha_p$. Glavna komponenta je tako iskazana preko koeficijenta α_i i varijance λ_i.
4. Komponente koje se u modelu odnose na malu proporciju varijacija podataka se eliminiraju. Na primjer, ako prve dvije komponente objašnjavaju 95% varijance, onda se sve ostale eliminiraju. Tada su prve dvije komponente zapravo glavne komponente.[35]

Klasteriranje

Klasteriranje (engl. *clustering*) je tehnika strojnog učenja koja uključuje grupiranje podataka. S obzirom na skup podataka, možemo koristiti algoritam klasteriranja za razvrstavanje svake podatkovne točke u određenu skupinu. Teoretski, podatkovne točke koje su u istoj skupini trebaju imati slična svojstva i/ili značajke, dok bi podatkovne točke u različitim skupinama trebale imati vrlo različita svojstva i/ili značajke. Klasteriranje je metoda nenadziranog učenja i uobičajena je tehnika statističke analize podataka koja se koristi u mnogim područjima.[36]

Klasteriranje je vrlo važno jer određuje unutarnje grupiranje među neobilježenim podacima. Ne postoje kriteriji za dobro grupiranje podataka nego ovisi o osobi koja provodi samo grupiranje i kriterijima koji zadovoljavaju potrebe zbog koje se grupiranje i provodi. Ovaj algoritam mora dati neke pretpostavke koje čine sličnost točaka, a svaka pretpostavka čini različite i podjednako valjane skupine.[37]

2.6.4.3. Regresijski algoritmi

Regresijska analiza neizbježan je korak u izradi modela. Regresija traži odnose među varijablama. Obično se razmatra neki fenomen od interesa koji ima brojna opažanja te svako opažanje ima dvije ili više značajki. Pretpostavlja se da barem jedna značajka ovisi o drugima i pokušava se uspostaviti odnos među njima. Drugim riječima, treba pronaći funkciju koja će dovoljno dobro prikazati neke značajke ili varijable.

Dostupne su mnoge regresijske metode, a u ovom radu biti će detaljno spomenute one koje su korištene tijekom modeliranja: stabla odluke i metoda nasumičnih šuma.

Stabla odluke

Stabla odluke (engl. *Decision Trees*) je svestrani algoritam strojnog učenja koji može izvesti i regresijske i klasifikacijske zadatke pa čak i zadatke sa višestrukim izlaznim informacijama. Veoma su jaki algoritmi koji omogućuju rješavanje kompleksnih setova podataka. Ključan su dio algoritma nasumičnih šuma (engl. *Random Forests*) o kojima će biti nešto više u nastavku rada.[32]

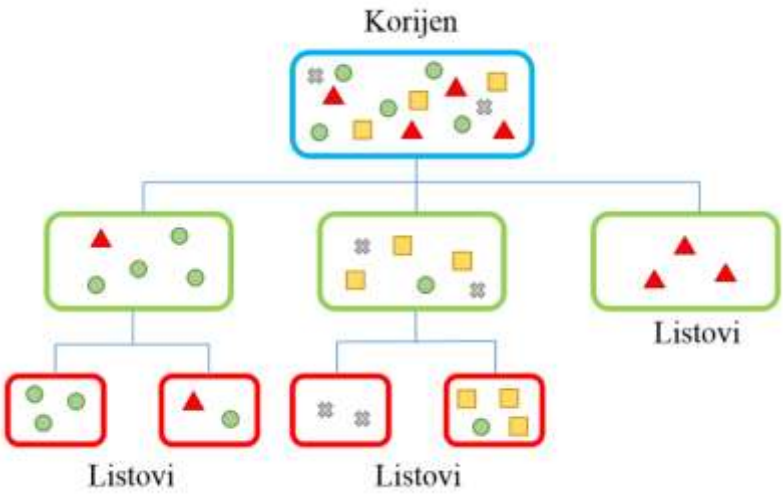
Opći motiv ovog algoritma je stvoriti model odluke koji može koristiti za predviđanje klase ili vrijednosti ciljnih varijabli učenjem pravila odlučivanja iz prethodnih podataka.

Glavni koraci prilikom izrade drva odluke su:

1. Postaviti najbolji atribut seta podataka u korijen (engl. *root*) stabla

2. Podijeliti set za treniranje u više pod-skupova (engl. *subsets*). Pod-skupovi bi trebali biti podijeljeni tako da svaki pod-skup sadrži podatke s istom vrijednošću kao i atribut
3. Ponavljati korake 1 i 2 na svakom pod skupu dok se ne pronađu čvorišta listova (engl. *leaf nodes*) na svim granama stabla

U stablima odluke, za predviđanje oznake klase počinje se od korijena stabla. Zatim se uspoređuju vrijednosti atributa u korijenu sa atributima u pod-skupovima. Na temelju usporedbe tih atributa slijedi se grana koja odgovara toj vrijednosti i prelazi se na sljedeće čvorište. Nastavljaju se uspoređivati vrijednosti zapisa sa drugim čvorištima stabla sve dok se ne dođe do lista s predviđenom vrijednošću klase. Glavna zadaća stabala je smanjiti srednje kvadratno odstupanje.[38] Na slici 5: prikazan je shematski prikaz stabla odluke.



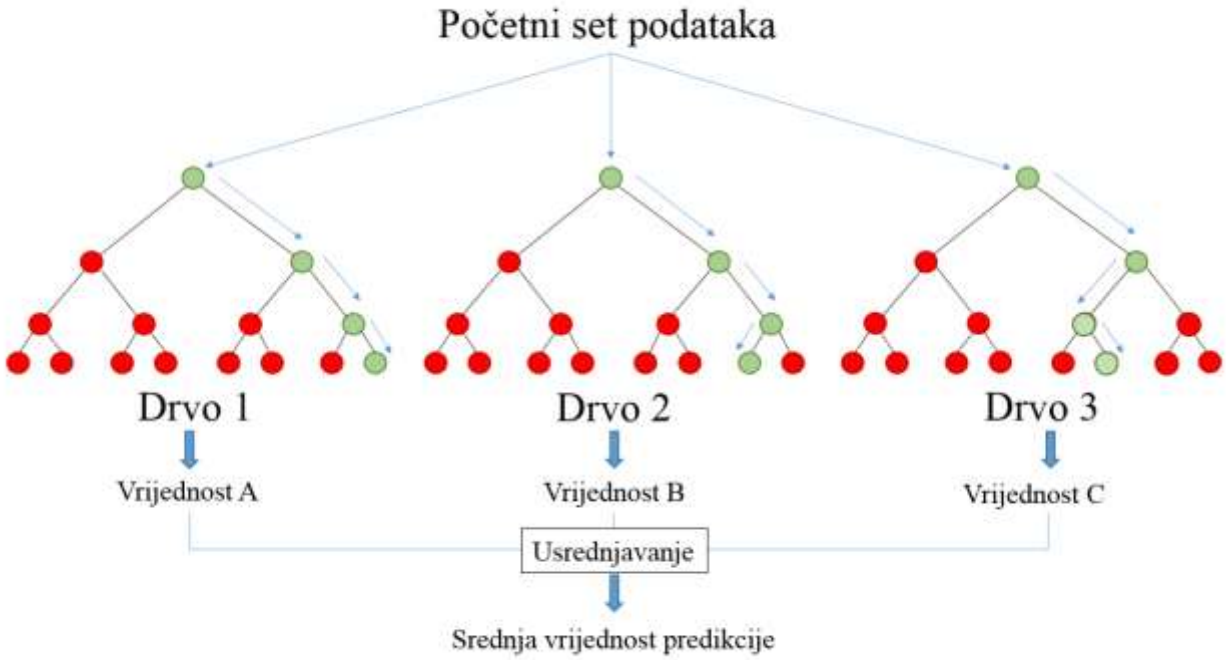
Slika 5. Shematski prikaz stabla odluke

Algoritam nasumičnih šuma

Nasumične šume (engl. *Random Forest*) su fleksibilan, jednostavan za korištenje algoritam strojnog učenja koji proizvodi, čak i bez podešavanja parametara, odličan rezultat većinu vremena. To je također jedan od najčešće korištenih algoritama, jer je jednostavan i može se koristiti za klasifikacijske i regresijske zadatke.

"Šuma" je sastavljena od više stabala odluke. Opća zamisao metode je da kombinacija modela učenja povećava ukupni rezultat. Jednostavnijim rječnikom, algoritam gradi više stabala odluke i spaja ih kako bi se dobilo što točnije i stabilnije predviđanje, odnosno krajnje predviđanje ovog algoritma je srednja vrijednost svih stabala unutar algoritma.

Algoritam dodaje modelu dodatnu slučajnost kako povećava broj drva odluke. Umjesto traženja najvažnije značajke prilikom stvaranja čvorišta, on traži najbolju značajku među slučajnim podskupom značajki. To rezultira velikom raznolikošću koja obično rezultira boljim modelom.[39]



Slika 6. Shematski prikaz nasumičnih šuma

2.7. Python

Python™ je skriptni objektno orijentirani programski jezik visoke razine i dinamičke semantike. Njegova visoka razina izgrađena u podatkovnim strukturama, u kombinaciji s dinamičkim pisanjem i dinamičkim vezivanjem, čine ga vrlo atraktivnim za razvoj brzih aplikacija, kao i za upotrebu povezivanja postojećih komponenata. Njegova velika prednost je što podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Sadrži opsežne biblioteke koje su dostupne u izvornom ili binarnom obliku bez naknada za sve bitnije platforme te se mogu slobodno distribuirati što ga čini lako dostupnim programskim jezikom.[40] Bitan koncept u Pythonu jest da je sve objekt. Objekti imaju metode (funkcije koje se vrše nad objektima), klase (skupine metoda) i atribute (svojstva).[41]

2.7.1. Programske biblioteke

Programske biblioteke poput Pandas[42], NumPy[43] i sckit-learn[44] (SKLearn) namijenjene su prvenstveno za obradu podataka i statističku analizu. Matplotlib[45] i Seaborn[**Error! Reference source not found.**] su programske biblioteke koje se ponajviše koriste za grafičke prikaze, dok se u svrhu kemoinformatike i strojnog učenja koristi biblioteka RDKit.[47]

2.7.1.1. Pandas

Ime ove biblioteke dolazi kombinacijom dvaju termina, *panel data*, ekonometrijskog termina za multidimenzionalni strukturirani set podataka, i *Python data analysis*. Uz prisutnost ove biblioteke omogućeno je da Python bude moćna i produktivna okolina za analizu podatka. Primarni objekti u pandas biblioteci su podatkovni okvir (engl. *Data Frame*) i podatkovni vektor (engl. *Series*), jednodimenzionalni matrični zapis. Podatkovni okvir je struktura za tablično skladištenje svih vrsta podataka, ne samo numeričkih, i sastoji se od podataka, indeksa i stupaca (engl. *column*). Pruža sofisticiranu funkcionalnost indeksiranja kako bi se olakšalo preoblikovanje, združivanje, razdvajanje i odabir podskupa podataka. Ima veliku važnost u manipulaciji, pripremi i čišćenju seta podataka.[48, 49]

2.7.1.2. NumPy

NumPy je skraćenica koja dolazi od *Numerical Python* i odavno je kamen temeljac numeričkog računanja u Pythonu. Pruža strukturiranje podataka, algoritama i koristi se kao poveznica potrebna za većinu znanstvenih aplikacija koje uključuju numeričke podatke u Pythonu. NumPy sadrži, između ostalog:

- brz i učinkovit višedimenzionalni matrični objekt, *ndarray*
- funkcije za izvođenje elementarnih proračuna s matricama ili matematičkih operacija između njih
- alate za čitanje i pisanje baza podataka na bazi matrica
- operacije linearne algebre, Fourierove transformacije i nasumično generiranje brojeva

Osim brzih mogućnosti obrade matrica koje NumPy pruža Pythonu, jedna od njegovih primarnih svrha za analizu podataka je prijenos podataka između algoritama i biblioteka.

Najučinkovitiji za pohranjivanje i manipulaciju numeričkih podataka za razliku od drugih struktura podataka koji su ugrađeni u Python.[43,48]

2.7.1.3. Scikit-Learn

Glavni alat opće namjene za strojno učenje koji se koristi prilikom programiranja u Pythonu. Uključuje nekoliko podmodela za modele kao što su:

- Klasifikacija
- Regresija
- Klasteriranje
- Redukcija dimenzija
- Selekcija modela i
- Predprocesiranje

Scikit-Learn otkriva širok raspon algoritama strojnog učenja, kako pod nadzorom tako i bez nadzora, koristeći dosljedno, orijentirano na zadatak sučelje, što omogućuje jednostavnu usporedbu metoda.[48, 50]

2.7.1.4. Matplotlib

Matplotlib je najpopularnija Pythonova biblioteka za izradu grafičkih prikaza i ostalih dvodimenzionalnih vizualizacija podataka. Dizajniran je za izradu grafičkih prikaza koji su pogodni za publikacije. Iako postoje i druge vizualizacijske biblioteke, Matplotlib ima najširu primjenu i kao takav može se vrlo lako integrirati s ostalim bibliotekama unutar Pythona.[48,51]

2.7.1.5. Seaborn

Seaborn je Pythonova vizualzacijska biblioteka bazirana na matplotlib biblioteci. Pruža sučelje visoke razine za prikaz atraktivnih i informativnih statističkih grafova.[52]

2.7.1.6. RDKit

RDKit je softver koji se koristi u svrhu kemoinformatike i strojnog učenja, a pogodan je za korištenje kako u Pythonu tako i u programu C++.

3. MODELIRANJE

Cilj ovog rada bio je razviti model koji najbolje predviđa topljivost tvari u vodi na temelju njihove strukture na skupu organskih kemijskih spojeva pomoću dva različita regresijska algoritma, stabla odluke i nasumičnih šuma. Za razvijanje i testiranje modela koristio se set podataka preuzet iz literature[53,54] koji je prethodno bio pročišćen.[41] Konačan set podataka sačinjen je od 1311 molekula.

Prije samog razvoja modela bilo je potrebno pregledati set podataka kako bi se uklonile eventualne pogreške, ponavljajuće strukture ili strukture koje su u obliku soli. Osim manualnog pregleda seta podataka proveden je i postupak standardizacije na istom.

Kada je set podataka bio u potpunosti pročišćen za svaku molekulu izračunati su molekulski deskriptori i molekulski otisci nakon čega je slijedila izrada modela. Za obradu podataka i izradu modela korišten je programski jezik Python. Biblioteke koje su se koristile u ovom radu su:

1. Pandas za rukovanje i obradu svih vrsta podataka,
2. NumPy i SKLearn za matematičke i statističke operacije i
3. Matplotlib i Seaborn za grafičke prikaze.

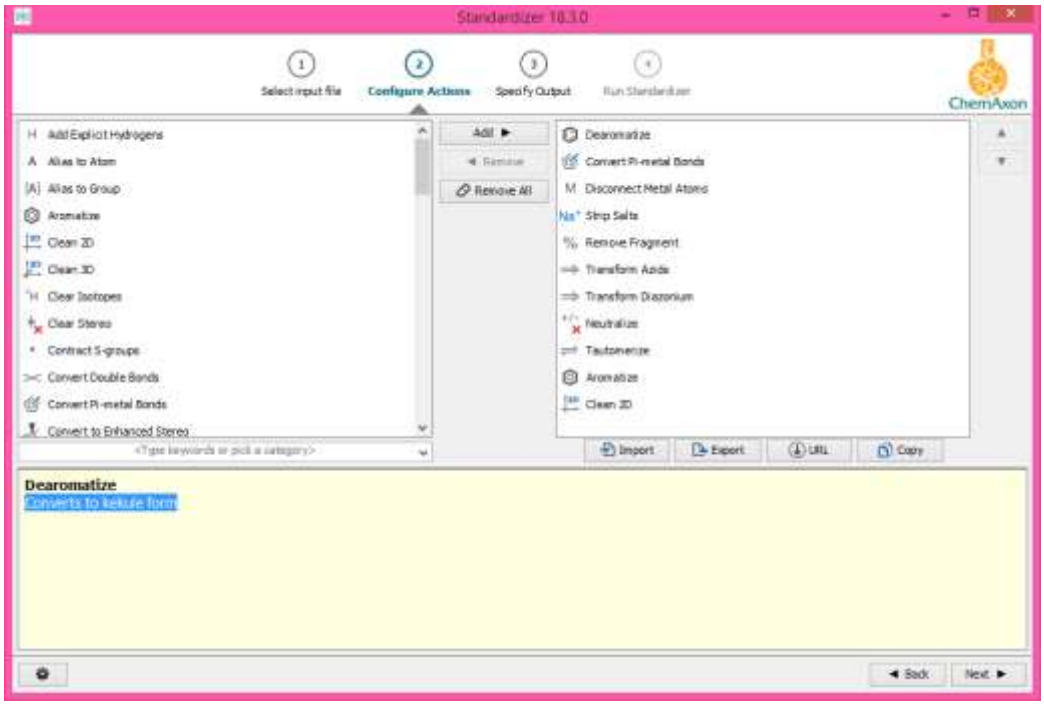
Sve navedene biblioteke integrirane su unutar programa Anaconda Navigator v.2.0[55] koja omogućuje primjenu i upravljanje bibliotekama te upotrebu sučelja Jupyter Notebook za programiranje i ispis rezultata.

3.1. Postupak standardizacije

U ovom radu za standardizaciju korišten je program ChemAxon Standardizer.[56]

Za kreiranje dokumenta koji se koristio za standardizaciju bilo je potrebno SMILES-e struktura spremiti u dokument s .txt ekstenzijom. Budući da softver koji se koristio u ovom radu ne podržava .txt format, polazni dokument je preveden u dokument s ekstenzijom .smiles.

Kada je datoteka bila spremna za provedbu standardizacije učitala se u sučelje i potom su se odabrale željene radnje za standardizaciju struktura. Budući da ne postoji univerzalni način na koji se mogu strukture standardizirati potrebno je poznavati softver koji će se koristiti u daljnjem modeliranju.



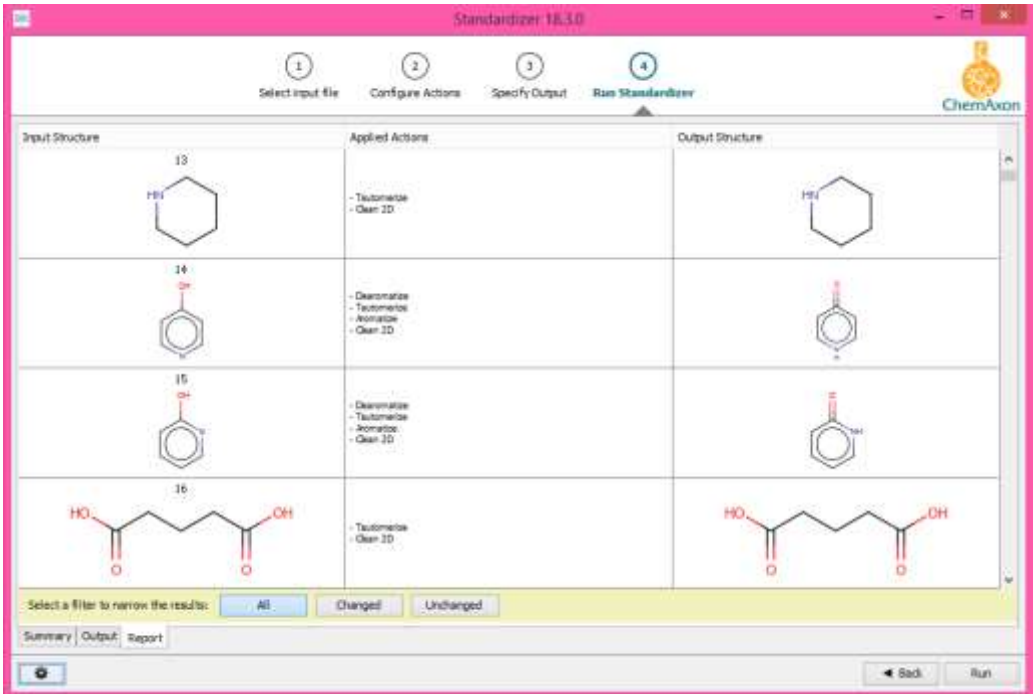
Slika 7. Prikaz odabranih radnji za standardizaciju

Kao što je prikazano na slici 7 s lijeve strane sučelja nalazi se popis radnji, a s desne strane prazan prozor u kojeg dodajemo radnje koje želimo izvršiti nad strukturama. Za ovaj rad izabrane su sljedeće radnje:

1. Dearomatizacija
2. Pretvaranje π-metalnih veza
3. Isključivanje metalnih atoma
4. Uklanjanje soli
5. Uklanjanje fragmenata
6. Transformacija azida
7. Transformacija diazona
8. Neutralizacija
9. Tautomerizacija
10. Aromatizacija
11. 2D čišćenje

Nakon odabira svih željenih radnji bilo je potrebno odredit mjesto gdje će se spremiti podaci s ekstenzijom .mrv, nakon čega je standardizacija bila pokrenuta. Po završetku otvara se sučelje prikazano na slici 8 u kojem je prikazan izvještaj sa svim standardiziranim

strukturama. Poželjno je po završetku standardizacije provjeriti vizualno što veći broj struktura kako bi se uklonile eventualno zaostale pogreške.



Slika 8. Finalni izvještaj standardizacije

Podaci spremljeni nakon provedbe standardizacije korišteni su za daljnje računanje molekularnih deskriptora i otisaka prsta molekula.

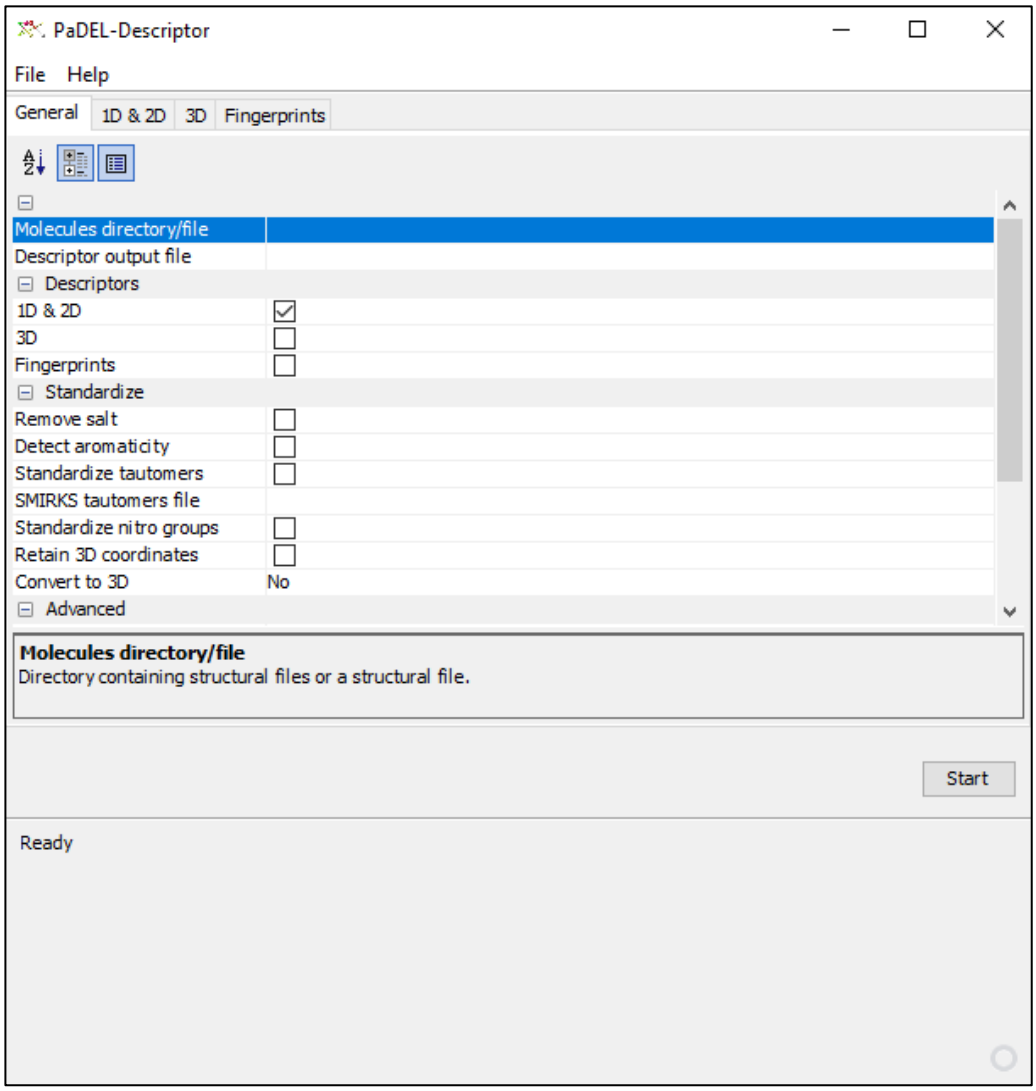
3.2. Računanje molekulskih deskriptora

Za računanje molekulskih deskriptora korišten je PaDEL Descriptor, v. 2.21[57]. Za računanje molekulskih deskriptora korišten je set podataka koji je u prijašnjem koraku standardiziran i pročišćen.

PaDEL je softver u kojem se mogu izračunati molekulski deskriptori i molekulski otisci. U verziji softvera 2.21 moguće je izračunati 1875 deskriptora, od toga 1444 1D i 2D deskriptora, 431 3D deskriptora te 12 vrsta molekulskih otisaka.

Na slici 9 prikazano je sučelje softvera. Za računanje deskriptora potrebno je odabrati mapu koja sadrži strukturne datoteke molekula u polje označeno plavom bojom. Najčešće su podržani formati datoteka MDL ili SMILES, ali preporučeni format je MDL. Također je potrebno odabrati datoteku u koju će se izračunati deskriptori pohraniti. Deskriptori se

spremaju u .csv formatu. Unutar softvera moguće je odabrati željene radnje, kao što je računanje 1D, 2D i 3D deskriptora, računanje otisaka kao i provedba standardizacije. Za ovaj rad izračunati su samo 1D i 2D deskriptori. Standardizaciju nije bilo potrebno provesti budući da je bila provedena prije računanja deskriptora.



Slika 9. Prikaz PaDEL sučelja

3.3. Računanje molekulskih otisaka

Za izračun molekulskih otisaka korišten je programski jezik Python, odnosno biblioteka za molekulsko modeliranje RDKit. Izračunati su Morganovi molekulski otisci koji se računaju i .mol zapisa molekula.[23] Morganovi molekulski otisci poznati su i kao kružni

molekulski otisci i stvoreni su primjenom Morganovog algoritma. Prilikom generiranja Morganovih otisaka potrebno je definirati radijus otiska.[58]

Prilikom izrade modela korištene su različite duljine bit vektora (2048, 3072, 4096 i 5120) i veličine radijusa (2–5).

3.4. Izrada modela

Prilikom izrade modela korišteno je sučelje Jupyter Notebook koje je integrirano u program Anaconda.

Za izradu dobrog modela potrebno je imati skup molekula koje već imaju traženu izmjerenu aktivnost (ciljna varijabla) kako bi model mogao učiti iz poznatih strukturnih parametara i poznate aktivnosti, tj. odrediti kakva je funkcija koja prevodi deskriptore u aktivnost. Uz skup za učenje potrebno je imati i skup za testiranje modela na kojem se testira ispravnost modela. Također je potrebno znati koja aktivnost se želi računati, tzv. ciljna varijabla. U ovom radu ciljna varijabla bila je topljivost u vodi, logS.

U modeliranju je poželjno koristiti smanjeni broj deskriptora uz model kojeg je moguće objasniti. U tu svrhu je matrica predikcijskih varijabli (deskriptora) prethodno pročišćena. Pri tome su korišteni sljedeći koraci:

1. Uklanjanje varijabli s nenumeričkim vrijednostima
2. Uklanjanje diskretnih varijabli koje imaju oko 90% unosa vrijednosti nula (0)
3. Skaliranje kontinuiranih varijabli metodom MinMax (0 – 1)
4. Uklanjanje skaliranih kontinuiranih varijabli s jako niskom varijancom (0.01)
5. Uklanjanje visoko-koreliranih varijabli (korelacija veća od 80%)

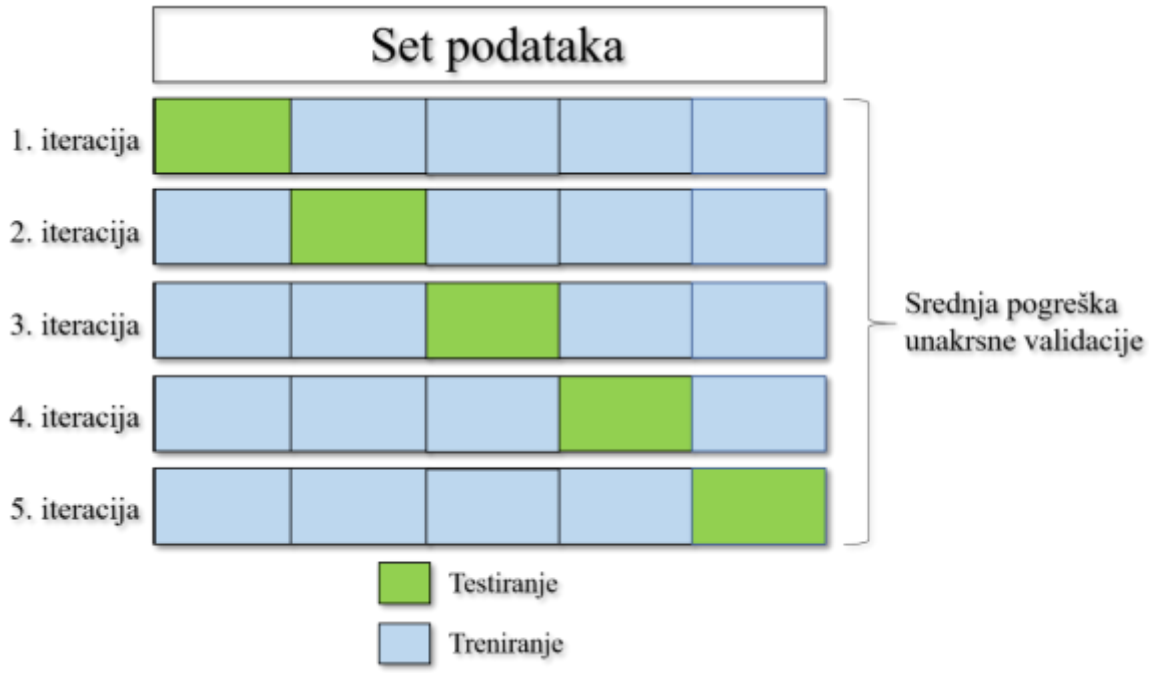
Prije izrade modela potrebno je uvesti sve potrebne biblioteke i definirati funkcije koje će biti korištene u daljnjoj izradi modela. Većina algoritama strojnog učenja ima svoje tzv. hiperparametre koje je potrebno odabrati i definirati; oni definiraju način *"fitanja"* na podatke. U ovom radu korišteni su sljedeći hiperparametri:

1. *n_estimators* određuje broj stabala u modelu nasumičnih šuma. Standardna predložena vrijednost za ovaj parametar je 10, što znači da će u model nasumičnih šuma biti izgrađeno 10 različitih stabala odluke.
2. *max_depth* određuje maksimalnu dubinu svakog stabla. Standardna predložena vrijednost je *"None"* što znači da će se svako stablo širiti dok svaki list ne bude

u potpunosti čist, a čistim listom se smatra onaj gdje svi podaci na listu potječu iz iste klase.

3. *min_samples_leaf* određuje minimalni broj uzoraka potrebnih za daljnje dijeljenje podataka. Standardna predložena vrijednost je 2 što znači da unutarnji list mora imati najmanje dva uzorka prije nego što se može podijeliti da bi imao specifičniju klasifikaciju.

Najbolji parametri određeni su unakrsnom validacijom. To je tehnika za procjenu modela strojnog učenja obučavanjem nekoliko modela na podskupovima dostupnih ulaznih podataka (iz seta za učenje) i njihovim vrednovanjem na komplementarnom podskupu podataka. Set podataka podijeljen je na 5 manja skupa podataka za. Iteracija se provodi 5 puta na način da se uvijek na 4 skupa model trenira, dok jedan skup služi za testiranje.



Slika 10. Shematski prikaz unakrsne validacije

U modeliranju je potrebno učiti modele, odnosno pokušati učiti što univerzalniji model. Kako bi model mogao učiti potrebno je set podataka razdvojiti na skup za učenje (engl. *Training set*) i na skup za validiranje (engl. *Test set*). Skup za učenje sadrži poznate vrijednosti i model se uči na tom skupu kako bi se kasnije primijenio na ostale podatke. Skup za validiranje koristi se za testiranje predviđanja našeg modela. Postotak spojeva korišten za razvoj i validaciju modela iznosio je 75%, dok je preostalih 25% spojeva korišteno kao

vanjski set podataka za testiranje samog modela. Spojevi za vanjski set podataka bili su odabrani nasumično.

Model se trenirao i testirao pomoću dva različita regresijska algoritma, stabla odluke i nasumične šume, te su analizirana dva različita pristupa, molekulski deskriptori i molekulski otisci.

Za procjenu točnosti (uspješnosti) algoritma učenja u rješavanju postavljenog zadatka potrebno je definirati tzv. mjeru (indeks) točnosti. Pomoću mjere točnosti za algoritme strojnog učenja moguće je uspoređivati primjenjivani algoritam s ostalim algoritmima učenja. Danas su najčešće u upotrebi 3 mjere točnosti algoritma učenja:

1. Srednja kvadratna pogreška(engl. *Mean Square Error*, MSE)
2. Korijen srednje kvadratne pogreške(engl. *Root Mean Square Error*, RMSE)
3. Normalizirani korijen srednje kvadratne pogreške(engl. *Normalised Root Mean Square Error*, NRMSE)

U ovom radu za predviđanje točnosti modela koristio se RMSE.

4. REZULTATI I RASPRAVA

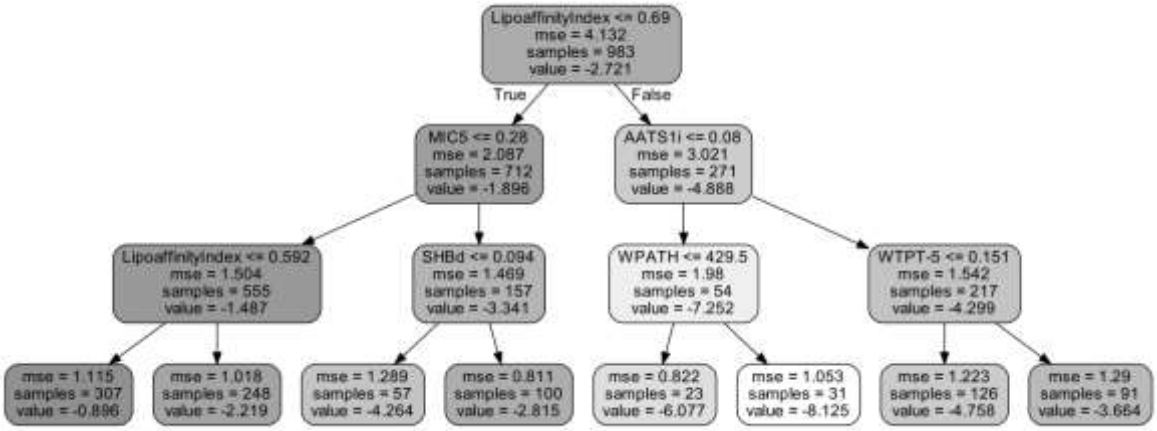
4.1. Molekulski deskriptori

Prije podjele seta podataka na skup za učenje i skup za validiranje izdvojili su se samo oni deskriptori koji koreliraju s manje od 80%. Sveukupno je izračunato 282 molekulska deskriptora s kojima se izrađivao model. Za izradu modela koristila su se dva regresijska algoritma, stabla odluke i nasumične šume.

Stabla odluke

Set podataka podijeljen je na set za treniranje i set za učenje. Model se učio, odnosno trenirao, na skupu za učenje, a zatim se provjeravao na skupu za validiranje.

Stablo odluke sastoji se od korijena, čvorišta i listova. Prilikom izrade modela stablo iterativno bira varijable koje najbolje opisuju ciljnu varijablu. Na slici 11 prikazan je algoritam stabla odluke kojem je zadan parametar *max_depth* = 3, što znači da se nakon tri grananja model prisilno zaustavlja (engl. *early stopping*). Prema slici 11 vidljivo je da je molekulski deskriptor, LipoaffinityIndex, koji je prema algoritmu stabla odluke najbolji za dijeljenje podataka na manje podskupove.



Slika 11. Shematski prikaz algoritma stabla odluke, *max_depth*=3

Učinkovitost modela prikazana je RMSE vrijednošću koja je ona iznosila 1,105.

Također je model isproban u slučaju kada se parametar *max_depth* promijeni i postavi se zadana vrijednost koja glasi *"None"*. U tom slučaju model nema rano zaustavljanje, tj. maksimalnu vrijednost do koje se može širiti već će se svako stablo širiti sve dok listovi ne

budu u potpunosti čisti, odnosno dok u svakom listu ne budu podaci koji pripadaju jednoj vrijednosti topljivosti. Takav model imao je RMSE = 1,001 što ga čini boljim modelom od prijašnjeg, koji je više generalizirao.

Analizom modela dobiveno je 5 najvažnijih molekulskih deskriptora koji utječu na topljivost molekula, a prikazani su u tablici 2.

Tablica 2. Najvažniji molekulski deskriptori dobiveni algoritmom stabla odluke

Molekulski deskriptor	Važnost / %
LipoaffinityIndex	51,64
MIC5	10,49
AATS1i	9,76
WPATH	1,89
SHBd	1,89

Nasumične šume

Algoritam nasumičnih šuma ima, kao što je već ranije napisano, hiperparametre koje je potrebno postaviti (parametarski prostor) prije izrade finalnog modela.

Tablica 3. Ispitivani uvjeti za mrežnu pretragu pomoću algoritma nasumičnih šuma

	Zadano
n_estimators	100
	200
max_depth	10
	15
min_samples_leaf	2
	3

Parametri koji su bili određivani su *n_estimators*, *max_depth* i *min_samples_leaf*, a najbolja kombinacija parametara određena je pomoću algoritma mrežne pretrage (engl. *Grid Search*) koji se koristi za pronalaženje optimalnih hiperparametara modela koji zatim rezultiraju optimalnim modelom za dane podatke i ciljnu varijablu. Zadani su bili željeni parametri i njihove vrijednosti te su oni najbolji bili izračunati unakrsnom validacijom. Optimalni parametri ispitivali su se na skupu za učenje.

U tablici 4. prikazani su rezultati optimiranih parametara unakrsnom validacijom. Kao što je vidljivo, najveći koeficijent determinacije, R^2, postignut je za *max_depth* = 15, *min_samples_leaf* = 2 i *n_estimators* = 100, što znači da se regresijski algoritam nasumičnih šuma sastoji od 100 stabala odluke, svako stablo odluke ima maksimalno 15 listova i potrebno

je minimalno 2 podatka u listu prije sljedeće podjele. Parametarski prostor je moguće i širiti van ovih gabarita, ali potrebno je sustav prilagoditi računalnim kapacitetima infrastrukture.

Tablica 4. Rezultati optimiranja hiperparametara za algoritam nasumičnih šuma.

max_depth	*min_ssmples_leaf*	*n_estimators*	R^2
15	2	100	0,8855
15	2	200	0,8853
10	2	200	0,8849
10	2	100	0,8846
15	3	100	0,8836
15	3	200	0,8833
10	3	200	0,8831
10	3	100	0,8829

Rezultati su dobiveni unakrsnom validacijom na skupu za učenje podjelom podataka na manje setove podataka. Nakon provedbe 5 iteracija model je testiran na skupu podataka za validiranje. Najbolji model imao je RMSE_CV = 0,6887.

Analizom modela dobiveno je 10 najvažnijih molekulskih deskriptora koji utječu na topljivost molekula, a prikazani su u tablici 5.

Tablica 5. 10 najvažnijih deskriptora dobivenih unakrsnom validacijom algoritma nasumičnih šuma na cijelom setu podataka

Molekulski deskriptor	Važnost / %
LipoaffinityIndex	51,22
AATS1i	8,69
MIC5	7,42
AATS5v	3,41
AATS0v	1,94
SM1_DzZ	1,91
WPATH	1,59
BCUTc-1l	0,82
BCUTw-1h	0,73
apol	0,66

Nakon odabira 10 najvažnijih deskriptora ponovno je provedeno modeliranje pomoću unakrsne validacije, ali sa smanjenim setom podataka na spomenutih 10 varijabli. Unakrsna validacija provedena je na isti način kako je ranije opisano. U tablici 6. prikazani su rezultati optimiranih parametara unakrsnom validacijom na smanjenom setu podataka sa izabranih 10 deskriptora. Vidljivo je da se najveći koeficijent determinacije, R^2, postiže za:

max_depth = 15, *min_samples_leaf* = 2 i *n_estimators* = 200, što znači da se regresijski algoritam nasumičnih šuma sastoji od 200 stabala odluke, svako stablo odluke ima maksimalno 15 listova i potrebno je minimalno 2 podatka u listu prije sljedeće podjele.

Tablica 6. Rezultati optimiranja hiperparametara za algoritam nasumičnih šuma na smanjenom setu podataka.

max_depth	*min_ssmples_leaf*	*n_estimators*	R^2
15	2	200	0,8712
10	2	200	0,8703
15	2	100	0,8699
10	2	100	0,8695
15	3	200	0,8690
10	3	200	0,8690
15	3	100	0,8684
10	3	100	0,8681

U ovom slučaju najbolji model imao je RMSE_CV = 0,7165.

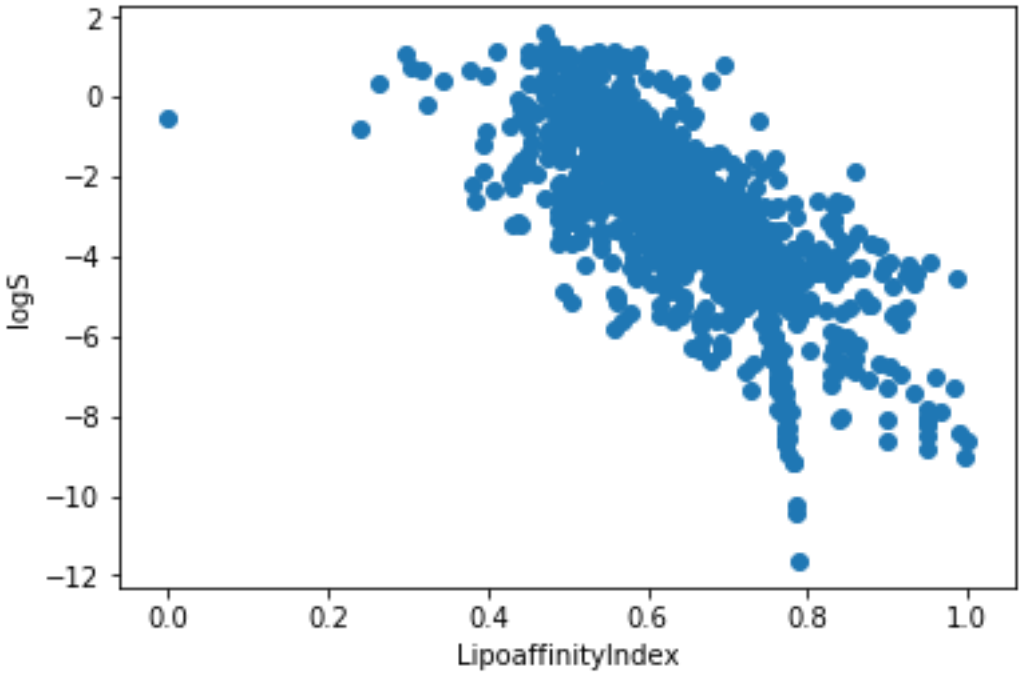
Analizom modela dobiveno je 5 najvažnijih molekulskih deskriptora koji utječu na topljivost molekula, a prikazani su u tablici 7.

Tablica 7. 5 najvažnijih deskriptora dobivenih unakrsnom validacijom algoritma nasumičnih šuma na smanjenom setu podataka

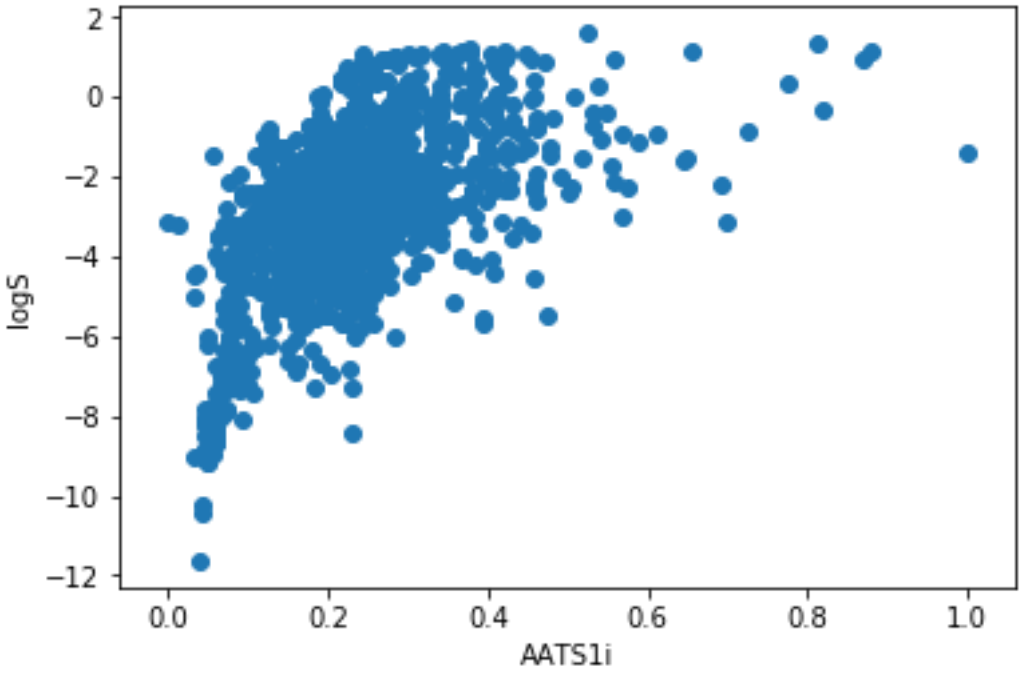
Molekulski deskriptor	Važnost / %
LipoaffinityIndex	54,21
AATS1i	11,83
MIC5	9,05
AATS5v	5,64
WPATH	4,04

Analiza pojedinačnih najvažnijih deskriptora

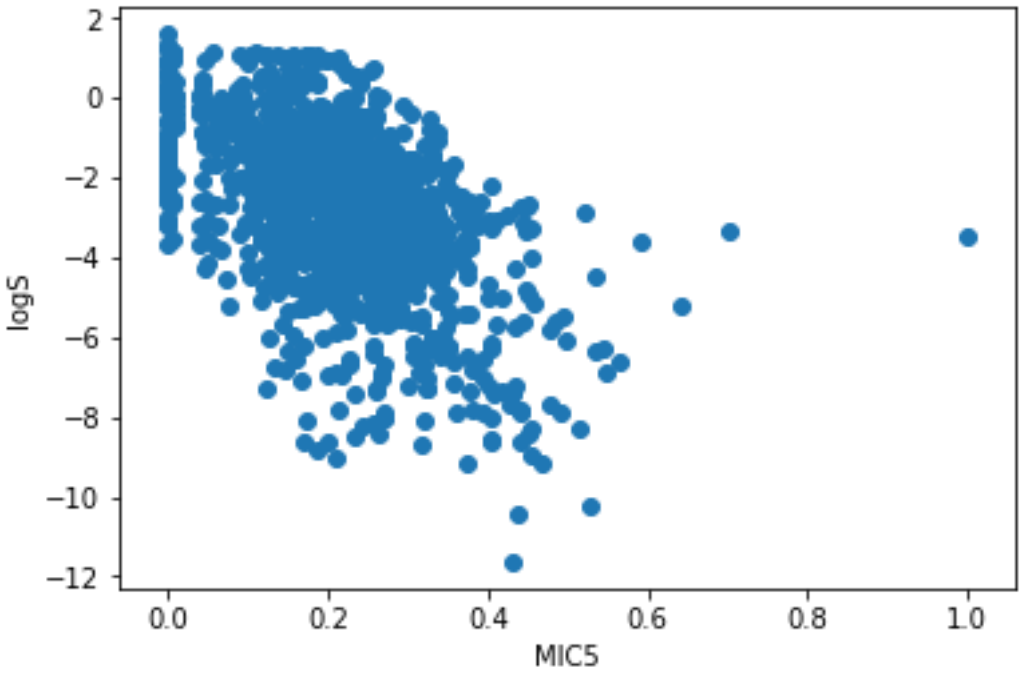
Na slikama 12 – 16 prikazane su ovisnosti 5 najvažnijih molekulskih deskriptora u ovisnosti o ciljnoj varijabli, log *S*.



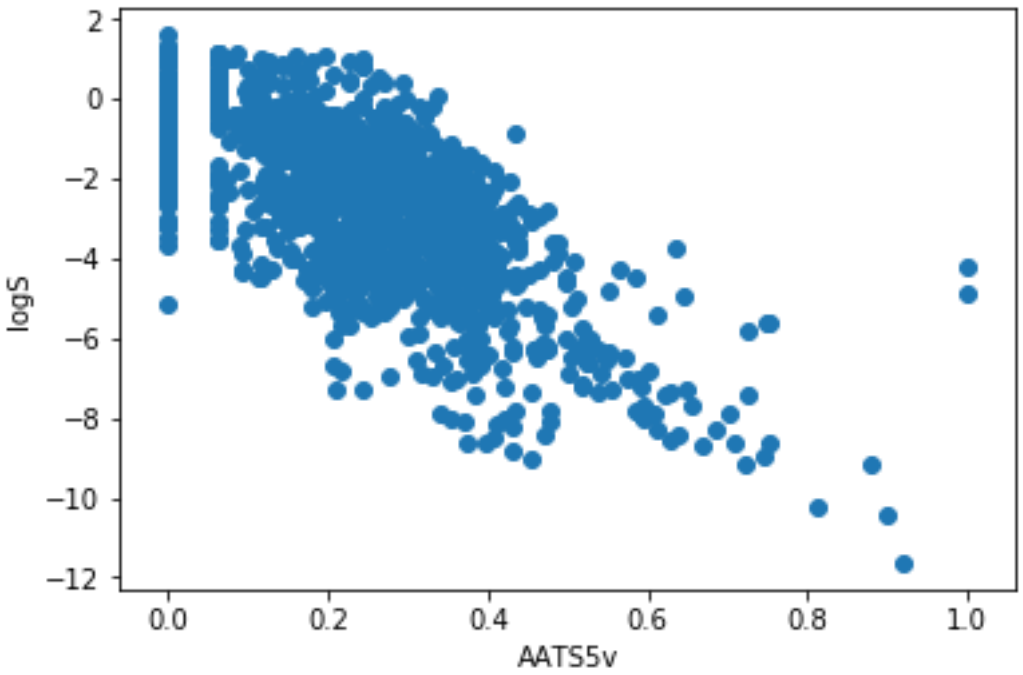
Slika 12. Ovisnost molekulskog deskriptora LipoaffinityIndex o ciljnoj varijabli, log *S*



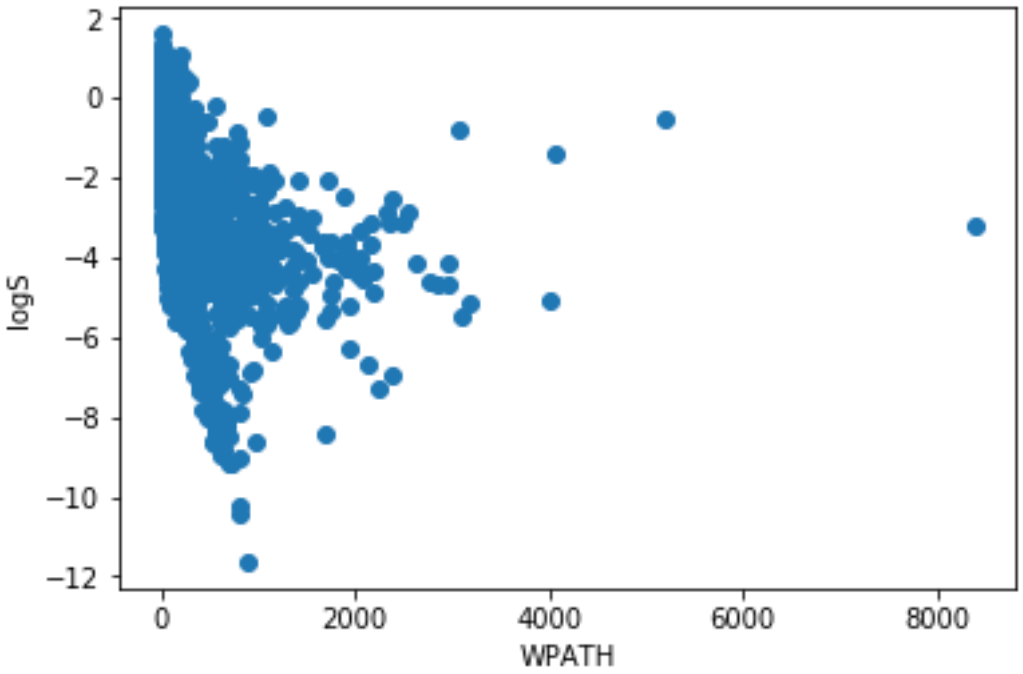
Slika 13. Ovisnost molekulskog deskriptora AATS1i o ciljnoj varijabli, log *S*



Slika 14. Ovisnost molekulskog deskriptora MIC5 o ciljnoj varijabli, log *S*



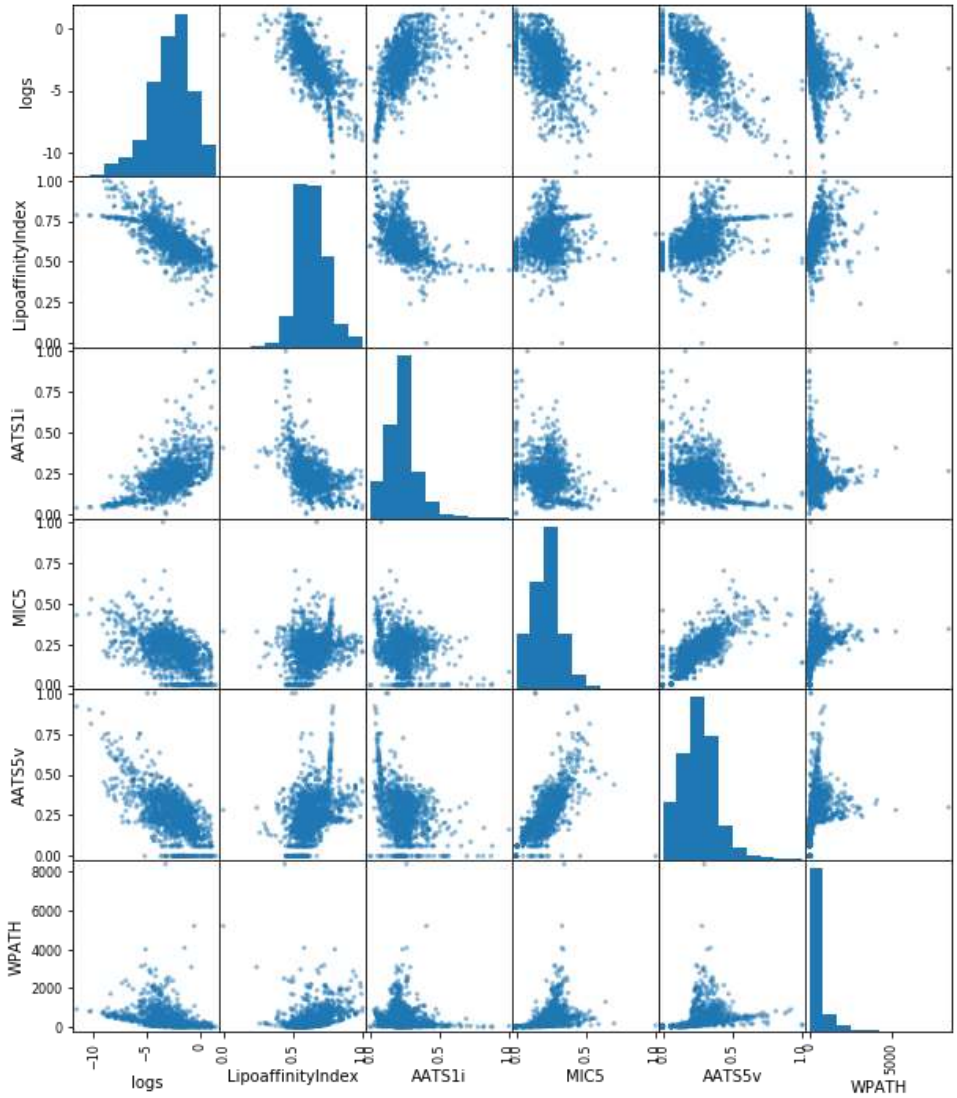
Slika 15. Ovisnost molekulskog deskriptora AATS5v o ciljnoj varijabli, log *S*



Slika 16. Ovisnost molekulskog deskriptora WPATH o ciljnoj varijabli, log *S*

Svi prikazani molekulski deskriptori izračunati su algoritmom nasumičnih šuma te su rezultat najboljeg modela dobivenog unakrsnom validacijom. Svih 5 molekulskih deskriptora pripadaju 2D skupu deskriptora. 2D deskriptori temelje se na molekularnoj topologiji poput topoloških indeksa, broj fragmenata itd. To su deskriptori koji su u širokoj primjeni jer sadrže dragocjene kemijske podatke kao što su veličina molekule, stupanj razgranatosti, susjedstvo atoma, elektronski i sterički efekti, fleksibilnost ili cjelokupni oblik molekule.

Na slici 17 prikazani su korelacijski odnosi svih 5 najvažnijih deskriptora u odnosu na ciljanu varijablu, log *S*. Vidljivo je da svih 5 najvažnijih deskriptora imaju tendenciju linearne korelacije sa ciljanom varijablom što je poželjno u modeliranju radi lakše obrade podataka.



Slika 17. Korelacijska matrica 5 najvažnijih molekulskih deskriptora sa ciljnom varijablom, log *S*

Usporedbom dvaju korištenih regresijskih algoritama uočeno je da bolju predikciju pruža model dobiven algoritmom nasumičnih šuma (RMSE = 0,7165) u odnosu na model dobiven algoritmom stabala odluke (RMSE = 1,001).

4.2. Molekulski otisci

Nakon dobivenih modela za predikciju molekulskih deskriptora, uočeno je da bolju predikciju pruža algoritam nasumičnih šuma. Sljedeći korak je bio primijeniti algoritam nasumičnih šuma na izradu modela za predikciju molekulskih otisaka, često korištenih u

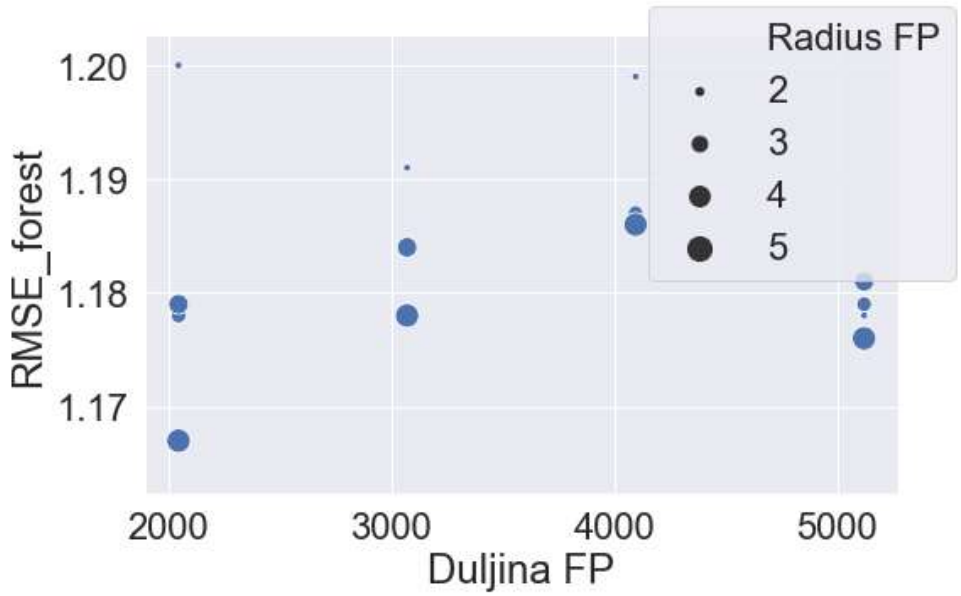
literaturi[59,60] te potom usporediti koji model ima bolju predikciju, model sa deskriptorima ili otiscima.

Prilikom izrade modela za molekulske otiske bio je zadan parametarski prostor za veličinu radijusa i duljinu bit vektora, koji je prikazan tablici 8.

Tablica 8. Parametarski prostor za veličinu radijusa i duljinu bit vektora

Veličina radijusa	Duljina bit vektora
2	2048
3	3072
4	4096
5	5120

Na slici 18 prikazani su parametri veličine radijusa i duljine bit vektora grafički. Iz grafa je vidljivo da najviše utječu na predikciju oni molekulski otisci koji imaju duljinu bit vektora 2048 i veličinu radijusa 5.



Slika 18 Grafički prikaz korištenih parametar za izradu modela algoritmom nasumičnih šuma

Da bi potvrdili tezu vidljivu na grafu, korišten je model koji je bio treniran sa zadanim parametrima na skupu za učenje, a potom testiran na skupu za validiranje. U tablici 9 prikazani su rezultati za sve izračunate kombinacije zadanih parametara. Prema dobivenim rezultatima vidljivo je da najbolju predikciju pruža model s veličinom radijusa 5 i duljinom bit vektora 2048 (RMSE = 1,1667).

Tablica 9. Prikaz rezultata dobivenih modelom za izračunate kombinacije parametara

Radijus	Duljina	RMSE
5	2048	1,1667
5	5120	1,1764
3	2048	1,1777
5	3072	1,1777
2	5120	1,1784
3	5120	1,1786
4	2048	1,1787
4	5120	1,1808
4	3072	1,1838
3	3072	1,1839
5	4096	1,1858
4	4096	1,1865
3	4096	1,1873
2	3072	1,1914
2	4096	1,1986
2	2048	1,2004

Prije izrade finalnog modela određeni su hiperparametri algoritma nasumičnih šuma pomoću algoritma mrežne pretrage. Parametri koji su bili određivani su *n_estimators, max_depth i min_samples_leaf.* Zadani su bili željeni parametri i njihove vrijednosti te su oni najbolji bili izračunati unakrsnom validacijom. Optimalni parametri ispitivali su se na skupu za učenje.

Tablica 10. Ispitivani uvjeti za mrežnu pretragu pomoću algoritma nasumičnih šuma

	Zadano
n_estimators	200
	300
max_depth	15
	20
min_samples_leaf	2
	3

U tablici 11. prikazani su rezultati optimiranih parametara unakrsnom validacijom. Kao što je vidljivo, najveći koeficijent determinacije, R^2, postignut je za *max_depth* = 20, *min_samples_leaf* = 2 i *n_estimators* = 300 što znači da se regresijski algoritam nasumičnih šuma sastoji od 300 stabala odluke, svako stablo odluke ima maksimalno 20 listova i potrebno je minimalno 2 podatka u listu prije sljedeće podjele.

Tablica 11. Rezultati optimiranja hiperparametara za algoritam nasumičnih šuma.

max_depth	*min_ssmples_leaf*	*n_estimators*	R^2
20	2	300	0,6990
20	2	200	0,6914
15	2	300	0,6070
15	2	200	0,6806
20	3	300	0,6797
20	3	200	0,6792
15	3	200	0,6708
15	3	300	0,6707

Rezultati su dobiveni unakrsnom validacijom na skupu za učenje podjelom podataka na manje setove podataka. Nakon provedbe 5 iteracija model je testiran na skupu podataka za učenje. Najbolji model imao je RMSE_CV = 1,1639.

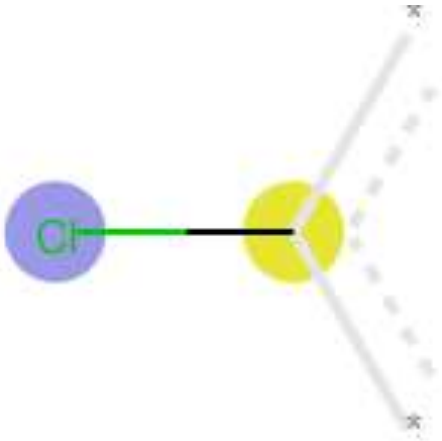
Analizom modela dobiveno je 5 najvažnijih molekulskih otisaka koji utječu na topljivost molekula, a prikazani su u tablici 12.

Tablica 12. 5 najvažnijih molekulskih otisaka dobivenih unakrsnom validacijom algoritma nausumičnih šuma.

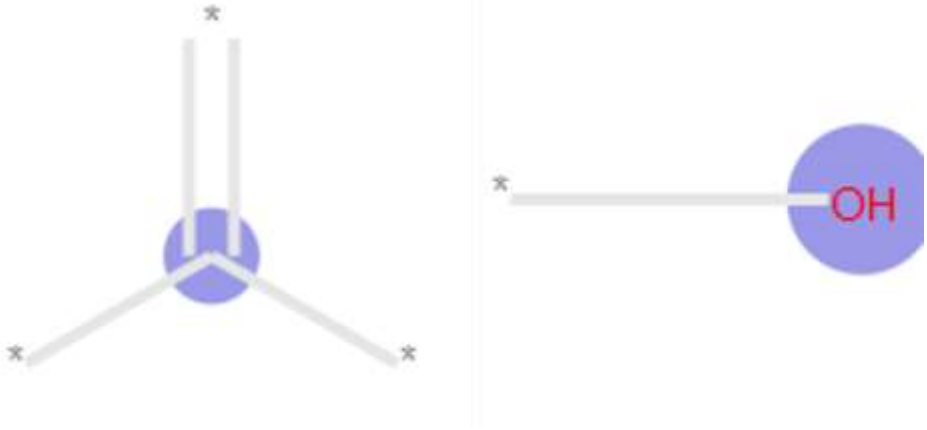
Molekulski otisak	Važnost / %
Fp561	20,07
Fp807	5,40
Fp352	4,59
Fp1873	3,20
Fp1143	3,14

Na slikama 19 - 23 prikazano je 5 najvažnijih molekulskih otisaka dobivenih modelom. Morganovi bitovi istaknuti su različitim bojama. Plavom bojom označen je središnji atom u okolini, žutom bojom označeni su aromatski atomi, a sivom bojom atomi alifatskih prstenova.

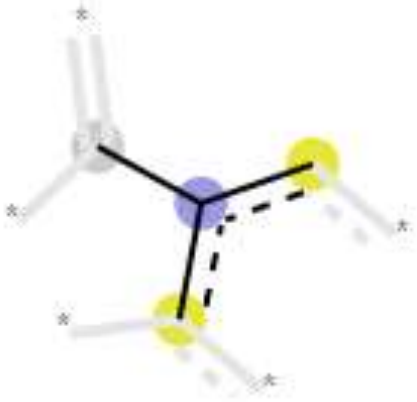
Usporedbom dvaju korištenih modela uočeno je da bolju predikciju pruža model dobiven pomoću molekulskih deskriptora (RMSE = 0,7165) u odnosu na model dobiven pomoću molekulskih otisak (RMSE = 1,1639).



Slika 19. Prikaz molekulskog otiska, Fp561, važnost u modelu 20,07%



Slika 20. Prikaz molekulskog otiska, Fp807, važnost u modelu 4,59%



Slika 21. Prikaz molekulskog otiska, Fp352, važnost u modelu 4,59%

Slika 22. Prikaz molekulskog otiska, Fp1873, važnost u modelu 3,20%

Slika 23. Prikaz molekulskog otiska, Fp1143, važnost u modelu 3,14%

5. ZAKLJUČAK

QSPR modeli koji predviđaju topljivost tvari u vodi, izrađeni su prema literaturno preuzetim vrijednostima topljivosti.

U istraživanju testirane su prediktivne sposobnosti modela pomoću molekulskih deskriptora i molekulskih otisaka s različitim algoritmima, stabla odluke i nasumične šume te.

Usporedbom modela pomoću molekulskih deskriptora, pokazalo se da algoritam nasumičnih šuma ima bolja prediktivna svojstva (RMSE = 0,7165) u odnosu na algoritam stabala odluke (RMSE = 1,001).

Nakon što je utvrđeno da algoritam nasumičnih šuma ima bolja prediktivna svojstva, uspoređivana su prediktivna svojstva između modela sa molekulskim deskriptorima i molekulskim otiscima. Usporedbom je pokazano da model s molekulskim deskriptorima ima bolja prediktivna svojstva (RMSE = 0,7165) u odnosu na model s molekulskim otiscima (RMSE = 1,1639).

6. LITERATURA

1. Nantasenamat, C., Isarankura-Na-Ayudhya, C., Naenna, T., Prachayasittikul, V., *A practical overview of quantitative structure-activity relationship*. EXCLI Journal, **8** (2009) 74–88.
2. Savjani, K. T., *Drug Solubility: Importance and Enhancement Techniques*, ISRN Pharm., 2012 (2012) 195727.
3. Clugston, M., Fleming, R., Advanced Chemistry, Oxford Publishing, 2000.
4. Gozalbes, R., Pineda-Lucena, A., *QSAR-based solubility model for drug-like compounds*, Bioorgan. Med. Chem., **18** (2010) 7078–7084.
5. Llinas, A., Glen, R. C., Goodman, J. M., *Solubility Challenge: Can You Predict Solubilities of 32 Molecules Using a Database of 100 Reliable Measurements?*, J. Chem. Inf. Model, **48** (2008) 1289–1303.
6. Lobenberg, R., Amidon, G. L., *Modern bioavailability, bioequivalence and biopharmaceutics classification system. New scientific approaches to international regulatory standards*. Eur. J. Pharm. Biopharm., **50** (2000) 3–12.
7. Avdeef, A., *Solubility of sparingly-soluble ionizable drugs*, Adv. Drug Deliver. Rev., **59** (2007) 568–590.
8. Roy, K., Kar, S., Das, R. N., Understanding the basics of QSAR for applications in pharmaceutical sciences and risk assessment, Academic Press, 2015.
9. Lipinski, C. A., Lombardo, F., Dominy, B. W., Feeney, P. J., *Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings*, Adv. Drug. Delivery Rev. **46** (2001) 3–26.
10. Russom, C. L., Bradbury, S. P., Carlson, A. R., *Use of Knowledge bases and QSARS to estimatethe relative ecological risk of agrichemicals problem formulation exercise***,** SAR QSAR Environ. Res., **4** (1995) 83–95.
11. Acharya C, Coop A, E. Polli J, D. MacKerell, Jr A. *Recent Advances in Ligand Based Drug Design: Relevance and Utility of the Conformationally Sampled Pharmacophore Approach*, Curr. Comput. Aided Drug. Des. **7** (2011) 10–22.
12. Kubinyi, H., 3D QSAR in Drug Design: Theory, Methods and Applications, ESCOM, Science Publisher, 1993.
13. Leelananda, S. P., Lindert, S., *Computational methods in drug disco*very, Beilstein J. Org. Chem., **12** (2016) 2694–2718.

14. Weininger, D., Weininger, A., Weininger, J. L., SMILES. 2. Algorithm for generation of unique SMILES notation, *J. Chem. Inf. Model*, **29** (1989) 97–101
15. Varnek, A., Tutorials in Chemoinformatics, Wiley, 2017.
16. Consonni, V., Tedeschini, R., Handbook of molecular descriptors, Wiley, 2000.
17. Roy K., Kar S., Das R. N., A primer on QSAR/QSPR modeling – fundamental concepts, Springer, 2015.
18. Danishuddin, K.A.U., *Descriptors and their selection methods in QSAR analysis: paradigm for drug design*, Drug Dev. Res., **21** (2016) 1291–1302.
19. Katritzky, A. R., Gordeeva, E. V., *Traditional topological indexes vs electronic, geometrical, and combined molecular descriptors in QSAR/QSPR research*. J. Chem. Inf. Comput. Sci., 1993, **33**, 835–857.
20. Karelson M, S. Lobanov V. *Quantum-Chemical Descriptors in QSAR/QSPR Studies*, Chem. Rev., **96** (1996) 1027–1044.
21. Karelson, M, Molecular Descriptors in QSAR/QSPR, John Wiley & Sons, 2000.
22. Rogers, D., Hahn, M., *Extended-Conectivity Fingerprints*, J. Chem. Inf. Model, **50** (2010) 742–754.
23. Landrum, G., Fingerprints in the RDKit, ppt prezentacija, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwj2qtuk8LvkAhVQzBoKHRQSB08QFjAAegQIAhAC&url=https%3A%2F%2Fwww.rdkit.org%2FUGM%2F2012%2FLandrum_RDKit_UGM.Fingerprints.Final.pptx.pdf&usg=AOvVaw1on_e9_FUhFQ2d_SsG1Su-, (pristup 6. rujna 2019.)
24. Bolanča, T., Ukić, Š., Kemometrika u analitici okoliša, u Kaštelan-Macan, M., Petrović, M. (ur.), Analitika okoliša, HINUS i Fakultet kemijskog inženjerstva i tehnologije, Zagreb, 2013., pp. 359–374.
25. Ukić, Š., *Nazivlje u kemometriji/kemometrici?*, Kem. Ind., **65** (2016) 181–182.
26. Tanimoto, S. L., The Elements of Artificial Intelligence, Computer Science Press, 1987.
27. Graupe, D., Principles of Artificial Neural Networks (2. izd), World Scientific Publishing, 2006.
28. Petrović, I., Perić N., Inteligentno upravljanje sustavima, interna skripta, Fakultet elektrotehnike i računarstva, Zagreb, 2008.
29. Kröse, B., van der Smagt, P., An Introduction to Neural Networks, University of Amsterdam, 1996.

30. Novak, V., Perfilieva, I., Mockor, J., Mathematical Principles of Fuzzy Logic, Springer, 1999.
31. https://www.mathworks.com (pristup 05. lipnja 2019)
32. Geron, A., Hands-On Machine Learning with Sckit-Learn and Tensor Flow, O'Reilley, 2017.
33. http://www.ef.uns.ac.rs/Download/multivarijaciona-statisticka-analiza/2013-02-08-Principal-Component-Analysis.pdf (pristup 05. srpnja 2019.)
34. Jolliffe, I. F., Principal Components Analysis (2. izd), Springer-Verlag, 2002.
35. Consonni, V., Tedeschini, R., Molecular Descriptors, Wiley-VCH, 2009.
36. https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 (pristup 15. srpnja 2019.)
37. https://www.geeksforgeeks.org/different-types-clustering-algorithm/ (pristup 16. srpnja 2019.)
38. https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/ (pristup 01. lipnja 2019.)
39. https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd (pristip 01. lipnja 2019.)
40. https://www.python.org (pristup 23. listopada 2018.)
41. Lovrić, M., *Molekulsko modeliranje odnosa strukturnih svojstava i aktivnosti molekula s pomoću programskog jezika Python (prvi dio)*, Kem. Ind., **67** (2018) 409–419.
42. McKinney, W., Data Structures for Statistical Computing in Python, u Stéfan van der Walt, S., Millman, J. (ur.): Proceedings of the 9th Python in Science Conference, ScyPi, 2010, pp. 51–56.
43. https://numpy.org (pristup 23. listopada 2018.)
44. Pedregosa, F.,Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., *Scikit-learn: Machine Learning in Python*, J. Mach. Learn. Res. **12** (2012) 2825–2830.
45. Hunter, J. D., *Matplotlib: A 2D Graphics Environment*, Comput. Sci. Eng. **9** (2007) 90–95.
46. https://zenodo.org/record/883859#.XXIpbX9S-M8 (pristup 6. rujna 2019.)
47. http://www.rdkit.org/ (pristup 15. prosinca 2018.))
48. McKinney, W., Python for Data Analysis, O'Reilley, Sebastopol, 2017.

49. http://pandas.pydata.org (pristup 23. listopada 2018.)
50. http://sckit-learn.org (pristup 23. listopada 2018.)
51. http://matplotlib.org (pristup 24. listopada 2018.)
52. http://seaborn.pydata.org (pristup 24. listopada 2018.)
53. Tetko, I. V., Gasteiger, J., Todeschini, R., Mauri, A., Livingstone, D., Ertl, P., Palyulin, V. A., Radchenko, E. V., Zefirov, N. S., Makarenko, A. S., Tanchuk, V. Y., Prokopenko, V. V., *Virtual Computational Chemistry Laboratory – Design and Description*, J. Comput. Aided. Mol. Des., **19** (2005) 453–463.
54. Huuskonen, J., *Estimation of Aqueous Solubility for a Diverse Set of Organic Compounds Based on Molecular Topology*, J. Chem. Inf. Comput. Sci., **40** (2000) 773–777.
55. https://www.anaconda.com (pristup 31. kolovoza 2019.)
56. https://chemaxon.com/ (pristup 1. ožujka 2019.)
57. http://www.yapcwsoft.com/dd/padeldescriptor/ (prisup 18 srpnja 2018.)
58. http://biotriangle.scbdd.com/static/media/BioChem_feature.pdf (pristup 31. Kolovoza 2019.)
59. Rogers, D., Hahn, M., *Extended-Connectivity Fingerprints*, J. Chem. Inf. Model., **50** (2010) 742–754.
60. Zang, Q., Mansouri, K., Williams, A. J., Judson, R. S., Allen, D. G., Casey, W. M., Kleinstreuer, N. C., *In Silico Prediction of Physicochemical Properties of Environmental Chemicals Using Molecular Fingerprints and Machine Learning*, J. Chem. Inf. Model., **57** (2017) 36–49.

7. ŽIVOTOPIS

Valnea Sindičić [redacted] Pohađala je srednju Medicinsku školu u Rijeci od 2007. do 2011. godine. Nakon završene srednje škole odrađuje godinu dana strukovnog usavršavanja. 2012. upisuje preddiplomski studij Primijenjena kemija na Fakultetu kemijskog inženjerstva i tehnologije Sveučilišta u Zagrebu. Tijekom studija odradila je stručnu praksu u PLIVA Hrvatska d.o.o. na odjelu Analitičke kemije. Radila je kao student suradnik od 2016. do 2019. godine na odjelu Analitičke kemije u PLIVA Hrvatska d.o.o. Zvanje sveučilišne prvostupnice primijenjene kemije stekla je u rujnu 2017. obranivši rad pod naslovom „Primjena metode ubrzanog stabilitetnog testiranja na aktivnim farmaceutskim supstancama“ pod mentorstvom docenta Šime Ukića. Iste godine upisuje diplomski studij Primijenjena kemija na matičnom fakultetu.

8. PRILOZI

8.1. Prilog 1. Programski kod za modeliranje pomoću molekulskih deskriptora

```
import numpy as np
import pandas as pd
from rdkit import Chem
from rdkit.Chem import AllChem, Draw, Descriptors, rdmolops
from rdkit import DataStructs
from rdkit.Chem import rdMolDescriptors
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.feature_selection import VarianceThreshold
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor as forest
from sklearn.tree import DecisionTreeRegressor as tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV
import graphviz

load_inded=pd.read_csv("../01_solubility.txt", delim_whitespace=True, names=['cas', '
smiles', 'logs'], index_col='cas')
load_inded.head()

desc_1D_2D=pd.read_csv("../solub_stand_rdkit_1d2d_desc.csv",
index_col='Name').dropna(how='any', axis=1)
desc_1D_2D.index=load_inded.index

def kick_corr(df):
    # Create correlation matrix
    corr_matrix = df.corr().abs()

    # Select upper triangle of correlation matrix
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(np.bool))

    # Find features with correlation greater than 0.95
    to_drop = [column for column in upper.columns if any(upper[column] > 0.80)]

    # Drop features
    df=df.drop(to_drop, axis=1)
    return df
```

```
def sep_n_scale(X):
   X=X.apply(lambda s: pd.to_numeric(s, downcast='integer',
errors='ignore')).dropna(how='any', axis=1)

   #sep integers
   Xint=X.select_dtypes(include='integer')

Xint_nan=Xint.select_dtypes(include='integer').replace(0,np.nan).isnull().sum()/len(Xi
nt)>.9
   Xint2=Xint[Xint_nan[Xint_nan==False].index.tolist()]

   #sep float
   Xfloat=X.select_dtypes(include='float')

   Xscaled=pd.DataFrame(MinMaxScaler().fit_transform(Xfloat), index=Xfloat.index,
columns=Xfloat.columns)

   selector = VarianceThreshold(threshold=0.01)
   selected_floats=pd.DataFrame(selector.fit_transform(Xscaled),
columns=Xscaled.columns[selector.get_support()],\
                  index=Xscaled.index)#.columns
   decorr_floats = kick_corr(selected_floats).columns
   #X_back = Xfloat[selected_floats]
   X_back = Xscaled[decorr_floats]

   Xfinal=pd.concat([X_back,Xint2],axis=1)
   return Xfinal

X=sep_n_scale(desc_1D_2D)
X.head()

X.shape

y=load_inded.logs

Random Forest - Full dataset

train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(X, y, test_size=0.25,
random_state=42)

param_grid = {
            'n_estimators': [100,200],
            'max_depth': [10,15],
            'max_features':['auto'],
            'min_samples_leaf':[2,3],
            'random_state':[42],
            'n_jobs':[-1],

         }
grid_clf = GridSearchCV(forest(), param_grid, cv=5)
```

```
grid_clf.fit(train_set_x, train_set_y)
model_forest = grid_clf.best_estimator_
grid_clf.best_params_

cols_rf=['param_max_depth', 'param_min_samples_leaf',
      'param_n_estimators', 'mean_test_score']
pd.DataFrame(grid_clf.cv_results_).sort_values('mean_test_score',
ascending=False)[cols_rf].round(4)

y_pred_forest=model_forest.predict(test_set_x)
RMSE_forest=np.sqrt(MSE(test_set_y, y_pred_forest)).round(4)
RMSE_forest

feat_imp=pd.Series(model_forest.feature_importances_,
index=X.columns).sort_values(ascending=False)*100
feat_imp[feat_imp>.01][:10]

feat_imp[feat_imp>.01][:10].index
```

Random Forest - Small dataset

```
Xt = X[['LipoaffinityIndex', 'AATS1i', 'MIC5', 'AATS5v', 'AATS0v', 'SM1_DzZ',
      'WPATH', 'BCUTc-1l', 'BCUTw-1h', 'apol']]

train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(Xt, y, test_size=0.25,
random_state=42)

#parametrizacija
param_grid = {
              'n_estimators': [100,200],
              'max_depth': [10,15],
              'max_features':['auto'],
              'min_samples_leaf':[2,3],
              'random_state':[42],
              'n_jobs':[-1],

            }
grid_clf = GridSearchCV(forest(), param_grid, cv=5)
grid_clf.fit(train_set_x, train_set_y)
model_forest = grid_clf.best_estimator_
grid_clf.best_params_

cols_rf=['param_max_depth', 'param_min_samples_leaf',
      'param_n_estimators', 'mean_test_score']
pd.DataFrame(grid_clf.cv_results_).sort_values('mean_test_score',
ascending=False)[cols_rf].round(4)

y_pred_forest=model_forest.predict(test_set_x)
RMSE_forest=np.sqrt(MSE(test_set_y, y_pred_forest)).round(4)
RMSE_forest
```

```
feat_imp=pd.Series(model_forest.feature_importances_,
index=Xt.columns).sort_values(ascending=False)*100
feat_imp[feat_imp>.01]

LipoaffinityIndex = X.LipoaffinityIndex
AATS1i = X.AATS1i
MIC5 = X.MIC5
AATS5v = X.AATS5v
WPATH = X.WPATH

plt.scatter(LipoaffinityIndex, y)
plt.xlabel('LipoaffinityIndex')
plt.ylabel('logS')

plt.scatter(AATS1i, y)
plt.xlabel('AATS1i')
plt.ylabel('logS')

plt.scatter(MIC5, y)
plt.xlabel('MIC5')
plt.ylabel('logS')

plt.scatter(AATS5v, y)
plt.xlabel('AATS5v')
plt.ylabel('logS')

plt.scatter(WPATH, y)
plt.xlabel('WPATH')
plt.ylabel('logS')

data_joined = pd.concat([y, X], axis=1)
data_joined.head()

data_joined[['logs', 'LipoaffinityIndex', 'AATS1i', 'MIC5', 'AATS5v',
'WPATH']].corr().round(2)

%matplotlib inline
from pandas.plotting import scatter_matrix
scatter_matrix(data_joined[['logs', 'LipoaffinityIndex', 'AATS1i', 'MIC5', 'AATS5v',
'WPATH']], figsize=(10,12))
```

Decision Tree

```
train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(X, y, test_size=0.25,
random_state=42)

model_tree = tree(criterion='mse', splitter='best', max_depth=None,
min_samples_split=2, min_samples_leaf=2, random_state=0).fit(train_set_x,
train_set_y)
```

```
y_pred_tree=model_tree.predict(test_set_x)
RMSE_tree=np.sqrt(MSE(test_set_y, y_pred_tree)).round(3)
RMSE_tree

feat_imp=pd.Series(model_tree.feature_importances_,
index=X.columns).sort_values(ascending=False)*100
feat_imp[feat_imp>1]

train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(X, y, test_size=0.25,
random_state=42)

model_tree = tree(criterion='mse', splitter='best',
max_depth=3,random_state=0).fit(train_set_x, train_set_y)
y_pred_tree=model_tree.predict(test_set_x)
RMSE_tree=np.sqrt(MSE(test_set_y, y_pred_tree)).round(3)
RMSE_tree

import graphviz
from sklearn.tree import export_graphviz
Rdot_data = export_graphviz(model_tree, out_file=None,
                    feature_names=X.columns.tolist(),
                    filled=True, rounded=True)
graph = graphviz.Source(Rdot_data)

graph.render('pic_tree.pdf', view=False)
graph
```

8.2. Prilog 2. Programski kod za modeliranje pomoću molekulskih otisaka

```
import numpy as np
from rdkit import Chem
from rdkit.Chem import AllChem, Draw, Descriptors, rdmolops
from rdkit import DataStructs
from rdkit.Chem.Fingerprints import FingerprintMols
from rdkit.Chem import rdMolDescriptors
import pandas as pd
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor as forest
from sklearn.tree import DecisionTreeRegressor as tree

from rdkit.Chem import rdMolDescriptors
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
```

```
from rdkit import DataStructs

def smi2fp(smiles, rad=4, bits=4096):
    '''
    :param smiles: smiles to calc fingerprints from
    :param rad: radius
    :param bits: length of vector, 1024,2048,4096,5120
    :return: pandas series of the bit vector
    '''
    #prep dict and array
    bi = {}
    bv = np.zeros(bits)
    #convert to mol and calc FP
    mol_for_fp=Chem.MolFromSmiles(smiles)
    fp = rdMolDescriptors.GetMorganFingerprintAsBitVect(mol_for_fp, radius=rad,
bitInfo=bi, nBits=bits)
    #convert to array and series
    DataStructs.ConvertToNumpyArray(fp,bv)
    bs=pd.Series(bv)
    return bs

def draw_one_fp(smiles, rad, length, code):
    assert type(smiles)==str
    m1=Chem.MolFromSmiles(smiles)
    bi = {}
    fp = rdMolDescriptors.GetMorganFingerprintAsBitVect(m1, radius=rad, bitInfo=bi,
nBits=length)
    bv = np.zeros((len(fp),),np.int16)
    DataStructs.ConvertToNumpyArray(fp,bv)
    bs=pd.Series(bv)
    return Draw.DrawMorganBit(m1,code,bi)

def draw_all_fp(smiles, rad, length):
    assert type(smiles)==str
    m1=Chem.MolFromSmiles(smiles)
    bi = {}
    fp = rdMolDescriptors.GetMorganFingerprintAsBitVect(m1, radius=rad,
nBits=length, bitInfo=bi)
    # show 10 of the set bits:
    tpls = [(m1,x,bi) for x in fp.GetOnBits()]
    display(Draw.DrawMorganBits(tpls,molsPerRow=4,legends=[str(x) for x in
fp.GetOnBits()]))

def parceli(x, postotak):
    '''funkcija za selekciju fingeprintsa'''
    parc_fp=x.sum(axis=0)/1311*100
    indic_=parc_fp[parc_fp>postotak].index.tolist()
    return x[indic_]
```

```
data_y=pd.read_csv("../01_solubility.txt", delim_whitespace=True, names=['cas',
'smiles', 'logs'], index_col='cas')
data_y.head()

y = data_y['logs']
y.head()

data_series=data_y.smiles

radius = []
length = []
shape = []
RMSE = []

for rad_ in [2,3,4,5]:
   for length_ in [2048, 3072, 4096, 5120]:

      #printanje
      print('radius FP ', rad_, 'duljina FP ', length_)

      #dodavanje u matricu
      radius.append(int(rad_))
      length.append(int(length_))

      #postavit x - rijeseno
      x = data_series.apply(lambda z: smi2fp(z, rad_,length_))
      x.columns=['fp'+str(z) for z  in x.columns]

      #selekcija fingeprintsa
      x=parceli(x,2)

      print(x.shape)

       #dodavanje oblika u matricu
      shape.append(x.shape)

      # y mora bit poznat
      # train test set
      train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(x, y,
test_size=0.25, random_state=42)

      model_forest = forest(n_estimators=200, max_depth=20, n_jobs=-1,
min_samples_leaf=2, random_state=42).fit(train_set_x, train_set_y) #trenirat na train
setu
      #testirat model na test setu
      #racunat RSME
      y_pred_forest=model_forest.predict(test_set_x)
      RMSE_forest=np.sqrt(MSE(test_set_y, y_pred_forest)).round(4)
      print(RMSE_forest)
```

```
        #dodavanje u matricu
        RMSE.append(RMSE_forest)

#dataframe prima samo np.array pa je potrebno konvertirati
radius = np.asarray(radius)
length = np.asarray(length)
shape = np.asarray(shape)
RMSE = np.asarray(RMSE)

#definiranje dataframea gdje će se sakupljati podatci
RMSE_rf = pd.DataFrame(columns=('Radius FP', 'Duljina FP', 'Shape',
'RMSE_forest'))

RMSE_rf["Radius FP"] = radius
RMSE_rf["Duljina FP"] = length
RMSE_rf["RMSE_forest"] = RMSE
RMSE_rf["Shape"] = shape
RMSE_rf.sort_values('RMSE_forest')

import seaborn as sns
import matplotlib.pyplot as plt
sns.set(font_scale=2)
fig, ax = plt.subplots(figsize=(8,5))
cmap = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
ax = sns.scatterplot(x="Duljina FP", y="RMSE_forest", size='Radius FP',
data=RMSE_rf,palette="Set2",
                     sizes=(20, 200),legend="full")
ax.legend(bbox_to_anchor=(1.1, 1.1))

#set X and y
x_final = data_series.apply(lambda z: smi2fp(z, 5,2048))
x_final.columns=['fp'+str(z) for z  in x_final.columns]
#selekcija fingeprintsa
x_final=parceli(x_final,2)


y = data_y['logs']

train_set_x, test_set_x, train_set_y, test_set_y = train_test_split(x_final, y,
test_size=0.25, random_state=42)


from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import GridSearchCV

param_grid = {
              'n_estimators': [200,300],
              'max_depth': [15,20],
              'max_features':['auto'],
              'min_samples_leaf':[2,3],
```

```
            'random_state':[42],
            'n_jobs':[-1],

          }
grid_clf = GridSearchCV(forest(), param_grid, cv=5)
grid_clf.fit(train_set_x, train_set_y)
model_forest = grid_clf.best_estimator_
grid_clf.best_params_

cols_rf=['param_max_depth', 'param_min_samples_leaf',
      'param_n_estimators', 'mean_test_score']
pd.DataFrame(grid_clf.cv_results_).sort_values('mean_test_score',
ascending=False)[cols_rf].round(4)

y_pred_best=model_forest.predict(test_set_x)
RMSE_best=np.sqrt(MSE(test_set_y, y_pred_best)).round(4)
RMSE_best

#vaznost FPa u modelu
feat_imp=pd.Series(model_forest.feature_importances_,
index=test_set_x.columns).sort_values(ascending=False)*100
feat_imp[feat_imp>2]

fp_code=1143
for m1 in x_final[x_final['fp%s'%fp_code]==1].index.tolist():
   try:
      smi=data_y.smiles[m1]
      print(smi)

      display(draw_one_fp(smi,code=fp_code,rad=5, length=2048))
      #display(draw_all_fp(smi,rad=5, length=2048))

   except:
      print('er')
   #break
```