Estimacija osnovnih energijskih razina molekula korištenjem ansambl metoda

Jurinčić, Luka

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski Rad

Estimacija osnovnih energijskih razina molekula korištenjem ansambl metoda

Rijeka, srpanj 2024.

Luka Jurinčić
0069084177

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski Rad

Estimacija osnovnih energijskih razina molekula korištenjem ansambl metoda

Mentor: prof. dr. sc. Zlatan Car

Komentor: doc. dr. sc. Nikola Anđelić

Rijeka, srpanj 2024.

Luka Jurinčić
0069084177

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 14.03.2024.

Zavod: Zavod za automatiku i elektroniku

Predmet: Osnove robotike

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Luka Jurinčić (0069084177)**

Studij: Sveučilišni diplomski studij elektrotehnike (1300)

Modul: Automatika (1331)

Zadatak: **Estimacija osnovnih energijskih razina molekula korištenjem ansambl metoda / Estimation of ground state energies of molecules using ensemble methods**

Opis zadatka:

Napraviti pregled literature u kojoj su korišteni različiti algoritmi umjetne inteligencije za estimaciju osnovnih energijskih razina molekula. Napraviti detaljnu statističku analizu skupa podataka. Primijeniti različite ansamble metode za estimaciju osnovnog energijskog stanja molekula te primjenom evaluacijskih metrika odrediti koje ansambl metode ostvaruju najvišu točnost estimacije. Odabrane ansambl metode spojiti u jedan ansambl s ciljem povećanja točnosti estimacije.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Mentor:
prof. dr. sc. Zlatan Car

Komentor:
dr. sc. Nikola Anđelić

Predsjednik povjerenstva za
diplomski ispit:
prof. dr. sc. Dubravko Franković

IZJAVA

Sukladno članku 7. Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog Fakulteta u Rijeci od 31. ožujka 2023., izjavljujem da sam samostalno izradio diplomski rad prema zadatku preuzetom dana 22. ožujka 2024.

Rijeka, srpanj 2024.

Luka Jurinčić

Ovom bi se prilikom htio zahvaliti svima koji su doprijenili mojim uspjesima. Najveću zahvalu upućujem svojim roditeljima koji se mojim uspjesima raduju više od mene i koji su mi omogućili sve što sam dosad u svome životu zamislio. Zahvaljujem se i ostatku svoje obitelji, kao i prijateljima Goranu i Dinu što su ove duge godine školovanja učinili mnogo ugodnijima. Također bi se još jednom zahvalio svojem kumu koji mi je od malih nogu bio uzor, te me potaknuo na fakultetsko obrazovanje. Nakraju bi se zahvalio svome mentoru prof. dr. sc. Zlatanu Caru na pruženoj prilici, te se dodatno zahvaljujem svome komentoru doc. dr. sc. Nikoli Anđeliću čiji su mi komentari i smjernice bile od velike važnosti pri izradi ovog diplomskog rada.

Sadržaj

1. Uvod

Proračun osnovne energetske razine molekula već dulji niz godina predstavlja veliki izazov. Problem koji se javlja prilikom analize osnovne enrgetske razine molekule je međudjelovanje više tijela. Standarni pristup rješavanju ovog problema podrazumijeva korištenje raznih numeričkih metoda za aproksimaciju osnovne energetske razine molekula. U ovom se radu za rješavanja spomenute problematike predlaže korištenje algoritama umjetne inteligencije.

U literaturi su korištene različite metode strojnog učenja za estimaciju osnovne energetske razine molekula, no u ovom se radu prednost daje ansambl metodama. Ove metode predstavljaju napredne algoritme strojnog učenja koje se sastoje od više osnovnih algoritama. Ideja ansambl metoda je kombinacija jednostavnih modela za rješavanje složenih problema. U ovom radu razmatramo mogu li različite ansambl metode precizno odrediti osnovnu energetsku razinu molekule, također se postavlja pitanje je li moguće odabrane ansambl metode spojiti u jedan ansambl s ciljem povećanja točnosti estimacije.

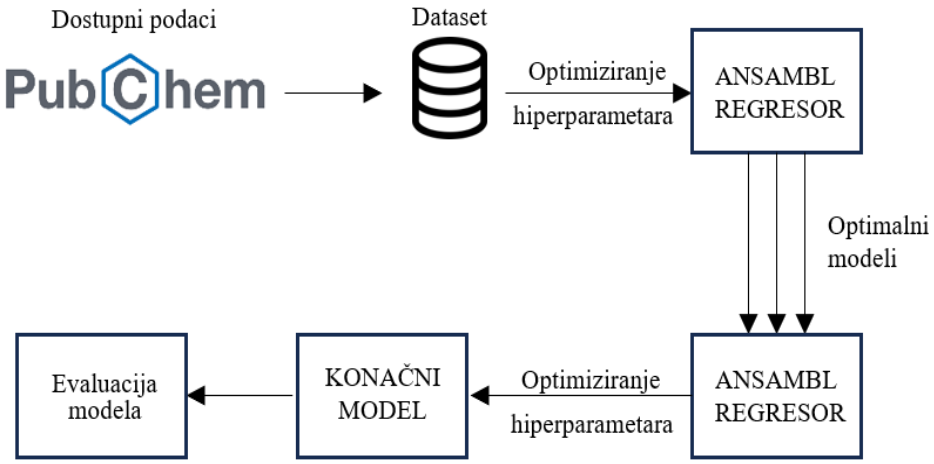
Orginalni dataset korišten u brojnim literaturama kod rješavanja ove problematike uključuje velik broj ulaznih varijabli koje opisuju numeričke metode koje se koriste za izračun osnovne energetske razine molekule. U ovom se radu predlaže alternativni pristup koji umjesto Coulombovih matrica koristi osnovne podatke o molekulama. Saznajemo je li moguće direktno iz osnovnih podataka o molekuli s visokom preciznošću odrediti osnovnu energetsku razinu. Vrši se analiza dataseta i primjenjuju se različite metode skaliranja. Provjerava se utjecaj varijacije osnovnog dataseta na preciznost ansambl metoda.

Za treniranje i validaciju ansambl algoritama koristi se unakrsna validacija i različite metode optimizacije hiperparametara. Postavlja se pitanje je li moguće s nekom od metoda optimizacije pronaći hiperparametre koji ostvaruju veću točnost estimacije. Analizira se utjecaj unakrsne validacije na kvalitetu predikcije ansambl regresora, te se također provjerava na koji način veličina foldova kod unakrsne validacije utječe na preciznost estimacije modela.

Na kraju se rada primjenom evaluacijskih metrika određuju najbolje ansambl metode koje ostvaruju najvišu točnost estimacije. Dobiveni rezultati za različite algoritme uspoređuju se međusobno i sa rezultatima iz analiziranih literatura. Zaključno su pokazane prednosti i nedostatci ansambl metoda kod estimacije osnovne energetske razine molekula, kao i ideja za moguće unaprijeđenje modela.

2. Materijali

U ovom će se poglavlju detaljno opisati korišteni materijali, priprema podataka i pregled literature. Proces započinje analizom znanstvenih radova, prikupljanjem i prilagodbom dostupnih podataka, zatim slijedi analiza prikupljenog dataseta i treniranje različitih algoritama umjetne inteligencije za estimaciju osnovnih energetskih razina molekula. Konačno se koristeći evaluacijske metrike uspoređuju odabrani algoritmi umjetne inteligencije. Pregled procesa modeliranja prikazan je na slici 2.1.



Slika 2.1. Proces modeliranja

2.1. Pregled literature

Osnovna energetska razina molekule predstavlja najnižu moguću razinu energije koju molekula može imati. U kvantoj su mehanici energijske razine kvantizirane, odnosno mogu zazueti jednu diskretnu energetsku razinu. Ovisno o diskretizacijskom broju n razlikujemo dva moguća stanja, osnovna energetska razina molekula $(n = 1)$ i pobuđeno stanje $(n > 1)$. Osnovna energetska razina molekule je stanje koje odgovara najstabilnijoj konfiguraciji elektrona.

Density Functional Theory (DFT) je popularna metoda koja se koristi za određivanje osnovne energetske razine molekula. Energija osnovnog stanja u potpunosti je određena samo za atom vodika (H) koji se sastoji od jednog elektrona i protona. Kod ostalih se atoma i molekula javlja međudjelovanje između više subatomski čestica i stvara se takozvani problem tri tijela koji onemogućuje deterministično rješavanje Schrödinger-ove jednadžbe [1]. Stoga se moraju koristiti numeričke metode koje koriste brojne aproksimacije.

U brojnim se litaraturama promatra ovaj problem, te se kao rješenje predlaže korištenje algoritama strojnog učenja. Dataset koji se analizira nije jednoznačno određen, već se promatraju različite vrste datasetova, primjerice u radovima [2] i [3] analizira se osnovna energetska razina velikih molekula.

Za najbolju usporedbu s rezultatima ovog rada koristit će se literatura [4] i [5] u kojima se koriste različite metode strojnog učenja za estimaciju osnovne razine molekula. U znanstvenom radu [4] koristi se dataset koji je služio kao osnova pri izradi ovog rada. Ulazne varijable tog dataseta dodatno su modificirane jer se postavlja pitanje, može li se direktno iz osnovnih podataka o molekuli precizno estimirati osnovna energetska razina molekule.

U radu [5] korišteni su sljedeći algoritmi: *k-nearest neighbors*, *mean predictor*, *linear regression*, *kernel ridge regression*, *vector regression* i *multilayer neural network*. Za evaluacijsku metriku korišteni su metrike MAE i RMSE. Dobivene vrijednosti MAE kreću se u rasponu od 8 do $178\ kcal/mol$, a vrijednosti RMSE u rasponu od 6 do $223\ kcal/mol$. Najbolji rezultati dobiveni su korištenjem *Laplacian kernel ridge* regresora, dok *mean predictor* algoritam pruža najlošije rezultate. U radu [4] se za estimaciju koriste metode *boosted trees* i *single layer neural network*, dobiveni RMSE rezultati iznose $41.81\ kcal/mol$ za *boosted trees* i $60.06\ kcal/mol$ za jednoslojnu neuronsku mrežu. Dodatno ističu da su njihovi rezultati nešto slabiji zbog značajno manje varijance u datasetu koji se koristio u radu [5], za razliku od dataseta baziranog na podacima iz PubChem kolekcije koji je korišten u radu [4].

2.2. Opis Dataseta

Podaci za istaživanje osnovnih energetskih razina molekula dobiveni su iz najveće svjetske kolekcije kemijskih podataka pod nazivom PubChem. U toj se kolekciji nalazi preko $118,000,000$ kemijskih spojeva. Orginalni dataset korišten u radu [4], kreiran je na način da su uzeti prvih $75,000$ molekula s PubChema, uzimajući u obzir kemijski sastav i primjenjujući sljedeći kriterij.

Svaka molekula mora biti izgrađena od elemenata C, H, N, O, P i S (CHNOPS), svaka molekula mora imati najmanje dva ili najviše 50 atoma, broj elektrona u molekuli mora biti paran, te maksimalna udaljenost između dva atoma ne smije biti veća od $25\ a_0$, gdje je a_0 Bohrov radijus, odnosno najvjerojatnija udaljenost između elektrona i jezgre vodikovog atoma pri osnovnoj energetskoj razini [6].

Ukoliko primjenimo zadani kriterij na početnih $75,000$ molekula, dobiven je dataset koji se sastoji od $16,242$ molekule. Svaka molekula ima svoj odgovarajući PubChem identifikacijski broj pomoću kojeg je moguće vidjeti kemijsku strukturu molekule, iz spomenute kemijske strukture izvučeni su karakteristični podaci o svakoj molekuli čime je stvoren dataset koji je korišten u ovom radu.

U literaturi [4] su se za opis svake molekule koristile detaljne Coulomb-ove matrice, dok je osnovna energetska razina molekule dobivena pomoću DFT-a. Ideja ovoga rada je umjesto ogromnog

broja ulaznih varijabli (Coulomb-ove matrice), direktno iz osnovnih podataka o molekuli precizno odrediti osnovnu energetsku razinu molekule. Ovim je postupkom broj ulaznih varijabli smanjen s 1277 na sedam osnovnih podataka o molekuli.

2.2.1. Opis ulaznih varijabli

Korišteni dataset sastoji se od $16,242$ molekule opisane sa sedam ulaznih varijabli koje će u nastavku biti objašnjene. Izlazna varijabla predstavlja osnovnu energetsku razinu molekule izraženu u Rydbergu, ta je vrijednost zatim transformirana u redovitije korištenu jednicu $kcal/mol$. Vrijedi $1Ry = 313.4953kcal/mol$, te se smatra da je zadovoljavajuća kemijska preciznost za osnovnu energetsku razinu molekule $1\ kcal/mol$.

Slijedi opis ulaznih varijabli:

- *Molekularna težina* poznatija kao relativna molekulska masa koju je moguće izračunati zbrojem svih atomskih masa u molekuli. Ovisno uzimamo li prosječnu vrijednost masa atoma ili koristimo vrijednosti izotopa prisutnih u molekuli razlikujemo *Molecular weight* i *Exact mass*. Razlika u ove dvije vrijednosti je u većini slučajeva neznatna, dok se kod nekih kemijskih elemenata koji imaju značajnije izotope može pojaviti vidljiva razlika. Relativna molekulska masa se koristi u masenoj spektrometriji za identifikaciju i potvrđivanje molekularne strukture. Uzimajući u obzir da *Exact mass* uključuje postojanje izotopa samim time posjeduje veću preciznost.

- *Topological Polar Surface Area* (TPSA) je široko primjenjiv opis molekule koji pokazuje na kakav će se način molekula apsorbirati, što je iznimno važno kod predikcije ponašanja lijekova. Vrijednost TPSA je moguće izračunati zbrojem površina polarnih atoma, kao što su kisik, dušik i s njima vezani atomi vodika [7].

- *Složenost molekule* je mjera za koju ne postoji konkretna definicija, ali ona upućuje na strukturalne zanimljivosti molekule kao što su topologija, grananje, broj prstenova, vrste kemijskih veza, veličina, simetrija i funkcionalnost molekule [8].

- *Hydrogen Bond Donor count* (HBD) predstavlja broj atoma vodika u molekuli koji mogu sudjelovati u stvaranju vodikove veze doniranjem vodikovog atoma.

- *Hydrogen Bond Acceptor count* (HBA) predstavlja broj atoma u molekuli koji mogu prihvatiti vodikovu vezu, najčešće su to atomi kisika i dušika.

- *Rotatable Bond Count* (RBC) pokazuje broj kemijskih veza koje se mogu slobodno rotirati oko svoje osi bez prekidanja ostalih kemijskih veza. Da bi se kemijska veza mogla slobodno rotirati ne smije biti dio prstenaste strukture, mora povezivati atome koji nisu vodik i konačno to mora biti jednostruka kemijska veza, budući da se dvostruka i trostruka veza ne mogu slobodno rotirati.

- *Heavy Atom Count* (HAC) je parametar koji predstavlja broj atoma koji se nalaze u molekuli pritom ne računajući atome vodika. Ovaj podataka omogućuje brzu indikaciju veličine molekule.

2.2.2. Inicijalna statistička analiza

Jednom kada je dataset sastavljen prvi je korak izvršiti inicijalnu statističku analizu. Za učitavanje dataseta koristi se knjižnica *Pandas*, koja se nalazi unutar programskog jezika Python. Ova knjižnica služi za učitavanje, analizu i spremanje .csv formata. *Pandas* knjižnica omogućuje čišćenje neurednih datasetova brisanjem redova koji nisu relevantni i sadrže pogrešne ili *null* vrijednosti.

Inicijalna statistička analiza dobivena je korištenjem *data.describe()* funkcije. Ova funkcija pruža sadržajan pregled dataseta iz kojega je moguće jednostavnije shvatiti strukturu i karakteristike podataka prije daljnje analize. Iz tablice 2.1 moguće je očitati centralne tendencije, raspršivanje podataka, te minimalne i maksimalne vrijednosti pojedinih deskriptora.

Tablica 2.1. Inicijalna statistička analiza

	Exact_mass	**TPSA**	**Complexity**	**HBD**	**HBA**	**RBC**	**HAC**	**GSE**
mean	179,772	50,792	196,018	1,049	2,776	2,338	12,727	-11,179
std	58,083	35,787	113,379	1,170	1,795	1,989	4,233	3,659
min	16,031	0	0	0	0	0	1	-23,245
25%	138,043	26	112	0	2	1	10	-13,476
50%	175,047	44	176	1	2	2	12	-10,835
75%	217,076	70	261	2	4	4	15	-8,624
max	522,991	330	927	8	18	15	32	-0,790
count	16242							

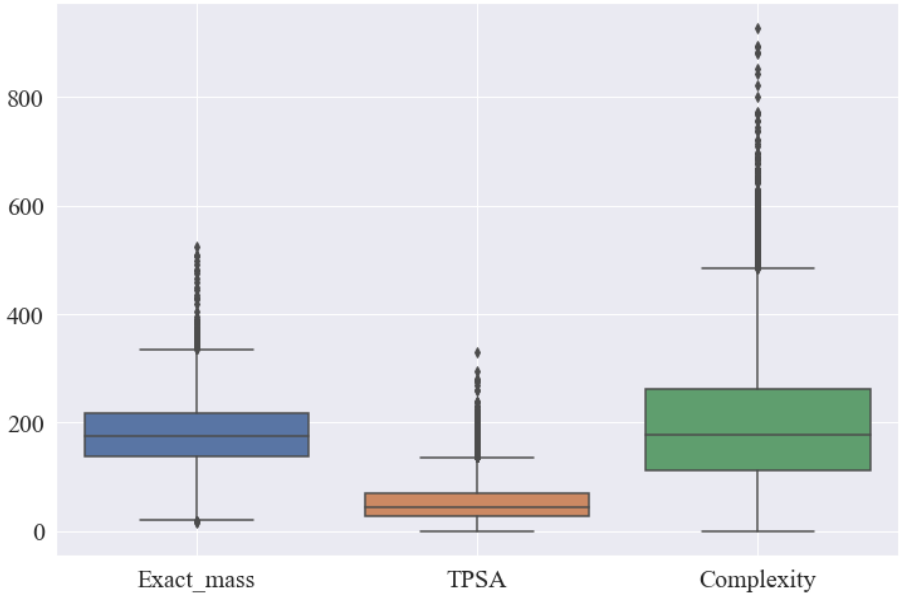
Funkcija *data.describe()* se koristi za provjeru kvalitete dataseta, primjerice iz tablice 2.1 analizom dobivenih vrijednosti moguće je doći do odgovarajućih zaključaka:

- *count* provjerava ukupan broj redova dataseta koji su ispunjeni numeričkim vrijednostima za svaku ulaznu varijablu, što omogućuje detekciju neispravno unesenih podataka.
- *min* i *max* odnosno najmanja i najveća vrijednost svakog stupca, pružaju informaciju o postajanju *outlier-a*, koji će biti detaljnije objašnjeni u nastavku rada.
- *mean* i *std* predstavljaju aritmetičku sredinu i standardnu devijaciju, te ukazuju na centralnu tendenciju podataka.
- *percentile* je podatak koji pokazuje vrijednost koja je jednaka ili veća od određenog postotka dataseta. Omogućuje dobar uvid u distribuciju podataka i nije pod negativnim utjecajem ekstremnih vrijednosti. Drugi kvartil ujedno predstavlja medijan, koji pokazuje da 50%

podataka u datasetu ima vrijednost koja je manja ili jednaka vrijednosti medijana. Medijan i aritmetička sredina statističke su metrike koje pokazuju središnju vrijednost podataka, no prednost medijana leži u tome što je otporan na vrijednosti pojedinih ekstrema, odnosno smanjuje utjecaj *outiler-a* pri računanju središnje vrijednosti.

Outlier je podatak koji značajno odstupa od ostalih uobičajenih vrijednosti u datasetu. Identificiranje postojanja ekstrema i njihov broj od velike je važnosti u statističkim i podatkovnim analizama. Postojanje ekstremnih vrijednosti u datasetu izobličuje centralne tendencije i ostale statističke pokazatelje.

Do pojave *outlier-a* u datasetu može doći zbog različitih utjecaja, od pogrešaka mjerenja do pogrešnog unosa podataka i prirodne nepredvidljivosti promatranog problema. Njihov utjecaj na podatke ovisi o nekoliko čimbenika. U slučaju da se dataset sastoji od manjeg broja ulaznih podataka postojanje nekoliko ekstremnih vrijednosti može značajno utjecati na statistiku i ponašanje dataseta. S druge strane ukoliko se dataset sastoji od velikog broja podataka, onda je po svojoj prirodi robustan i nekoliko ekstremnih vrijednosti neće značajnije utjecati na rezultate.

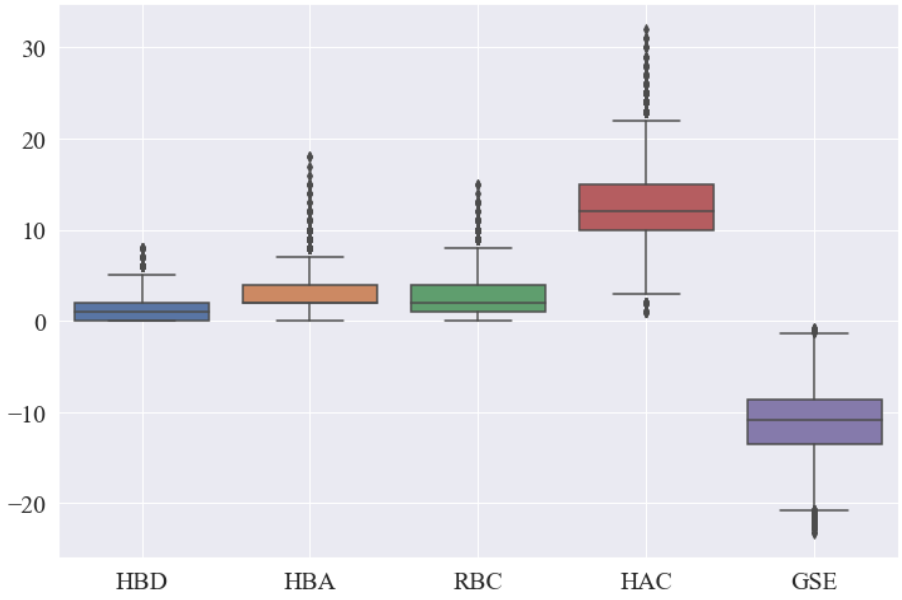


Slika 2.2. Box plot za ulazne varijable: relativnu molekulsku masu, topološku površinu polarnih atoma i složenost molekule

Jedan on načina detekcije postojanja *outlier-a* je vizualizacijom koristeći *Box plot*, koji grafički prikazuje podatke uz pomoć prethodno spomenutih kvantila vidljivih u tablici 2.1. *Box plot* je Python funkcija iz *seaborn* knjižnice. Direktno iz grafičkog prikaza moguće je odrediti medijan, uobičajene vrijednosti koje podaci poprimaju i ključno postojanje ekstremiteta koji odstupaju od ostalih podataka.

Na slikama 2.2 i 2.3 možemo vidjeti *Box plot* za cijeli dataset. Određene ulazne varijable imaju veći broj ekstrema od drugih, primjerice složenost molekule *(Complexity)* sadrži značajan broj ekstremnih vrijednosti, dok broj donora i rotirajućih veza *(HBD,RBC)* sadrže relativno mali

broj ekstremnih vrijednosti. Budući da se dataset sastoji od velikog broja podataka prirodno je robustan, te ekstremne vrijednosti neće značajnije utjecati na dobivene rezultate.



Slika 2.3. Box plot za ulazne varijable: HBD, HBA, RBC, HAC i izlaznu varijablu osnovne energetske razine

2.2.3. Pearsonova korelacijska analiza

Koeficijent korelacije *R* predstavlja ovisnost između dvije kvantitativne varijable. Može poprimiti vrijednosti od -1 do 1, gdje 1 predstavlja potpunu pozitivnu (linearnu) korelaciju kod koje se povećanjem jedne vrijednosti linearno povećava i neka druga vrijednost, dok -1 predstavlja potpunu negativnu ovisnost. Što je vrijednost *R* bliža ± 1 to je linearnost veća, za vrijednosti blizu 0 smatramo da ne postoji ovisnosti između dva podatka. Ukoliko je koeficijent korelacije podataka visok to značajno poboljšava svojstva dataseta. Pearsonov koeficijent korelacije *R* definiran je izrazom:

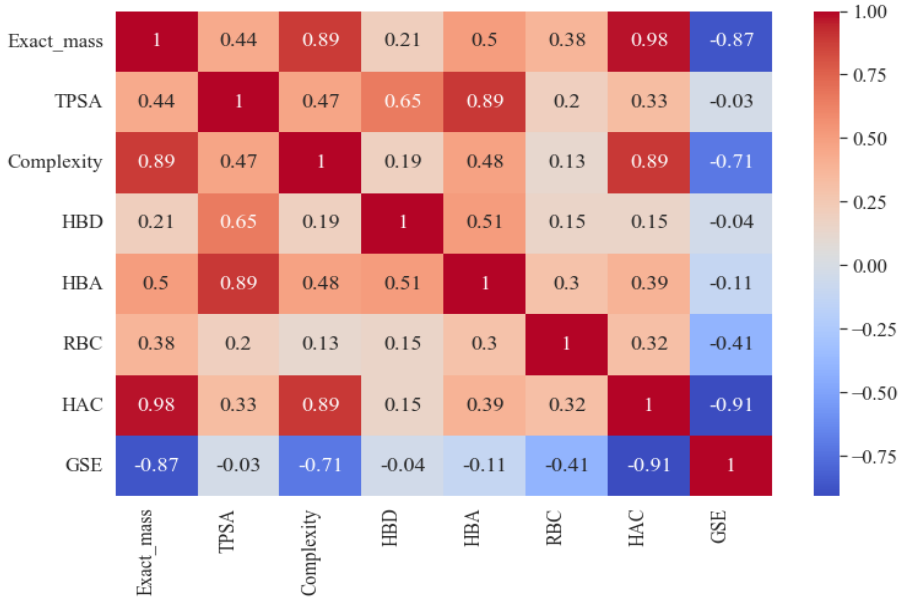
$$R = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{2.1}$$

gdje su x_i i y_i vrijednosti pojedinih varijabla, dok $\overline{x}$ i $\overline{y}$ predstavljaju srednju vrijednost tih varijabli.

Prikaz matrice korelacije na slici 2.4 dobiven je korištenjem funkcije *heatmap()* unutar *seaborn* knjižnice, koja se koristi za vizualizaciju podataka i pruža napredno sučelje za prikazivanje informacijskih statističkih grafičkih prikaza.

Promatrajući matricu korelacije na slici 2.4 moguće je analizirati ovisnost jedne varijable o drugoj. Uzmimo u obzir izlaznu varijablu osnovnog energetskog stanja molekule *(GSE)*. Možemo uočiti da ima najveću korelaciju s ulaznim varijablama: molekulska masa *(Exact_mass)* i brojem

teških atoma *(HAC)*. S druge strane korelacija s ulaznim varijablama TPSA, HBD i HBA je izrazito mala što znači da ne postoji značajna međusobna ovisnost između spomenutih ulaznih varijabli i osnovne energetske razine molekule.



Slika 2.4. Matrica korelacije

3. Metode

U ovom će se poglavlju prezentirati detaljan pregled korištenih metoda, od incijalnog skaliranja podataka i analiza ansambl algoritama umjetne inteligencije do optimizacije hiperparametara i opisa korištenih evaluacijskih metrika. Inicijalno skaliranje napravljeno je korištenjem nekoliko metoda skaliranja iz Python knjižnice *sklearn.preprocessing*. Svaka će transformacija dataseta biti objašnjena i utjecaj svake metode skaliranja na ulazne varijable biti će grafički prikazan.

3.1. Inicijalno skaliranje podataka

Skaliranje podataka važan je korak za većinu algoritama strojnog učenja, omogućuje skaliranje vrijednosti ulaznih varijabli što osigurava da utjecaj svake varijable u procesu treniranja algoritama bez obzira na raspon vrijednosti bude jednako značajan. Skaliranjem se smanjuje udaljenost između podataka, što dovodi do jednostavnijeg treniranja modela, te se ujedno postiže veća brzina i učinkovitost algoritama umjetne inteligencije. Postoje različite metode skaliranja podataka, svaka metoda ima svoje prednosti i nedostatke što primarno ovisi o raznim aspektima dataseta.

U nastavku će biti objašnjene različite metode skaliranja koje su korištene pri izradi ovog rada, te će se promatrati utjecaj pojedinih transformacija na svojstva dataseta kao što su srednja vrijednost, standardna devijacija i broj ekstremnih vrijednosti.

3.1.1. Standard Scaler

Standard Scaler je često korištena metoda skaliranja ulaznih varijabli, koja transformira dataset na način da svaka ulazna varijabla ima srednju vrijednost 0 i standardnu devijaciju 1. Dataset se skalira koristeći izraz:

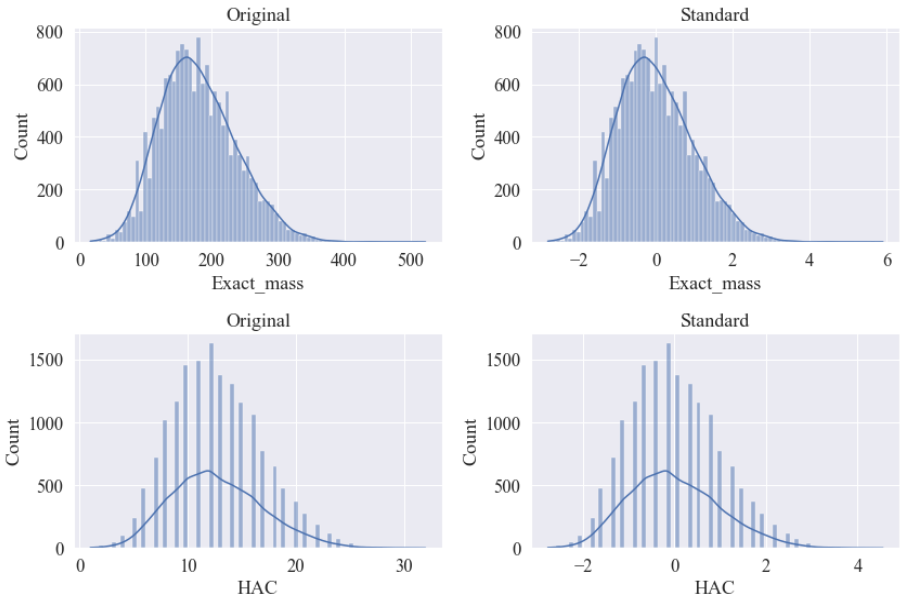
$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

gdje je x orginalna vrijednost, μ srednja vrijednost i σ standardna devijacija.

Standard scaler se koristi za poboljšanje ponašanja određenih algoritama koji ovise o udaljenosti između podataka i za regresijske algoritme koji su zasnovani na gradijentu, kod kojih omogućuje bržu konvergenciju. Standardizacijom podataka je također osigurano da svaka od ulaznih varijabli bez obzira na raspon vrijednosti poprima istu važnost.

Budući da izraz za standardno skaliranje 3.1 koristi aritmetičku srednju vrijednost i standardnu devijaciju iznimno je osjetljiv na postojanje *outlier-a*. Ekstremne vrijednosti značajno utječu na

ovu metodu skaliranja što može dovesti do toga da standartizirane vrijednosti neprecizno opisuju većinu podataka u datasetu.



Slika 3.1. Histogram za dvije ulazne varijable prije i poslije Standard skaliranja

Na slici 3.1 prikazan je histogram za dvije ulazne varijable, na lijevoj strani nalazi se orginalni dataset, dok je na desnoj strani skalirani dataset dobiven korištenjem *Standard Scaler-a*. Sa slike 3.1 jasno vidimo da su vrijednosti skaliranih ulaznih varijabli u značajno manjem rasponu $[-3, 6]$, dok je kod orginalnog dataseta raspon vrijednosti za molekularnu težinu *(Exact_mass)* $[0, 500]$.

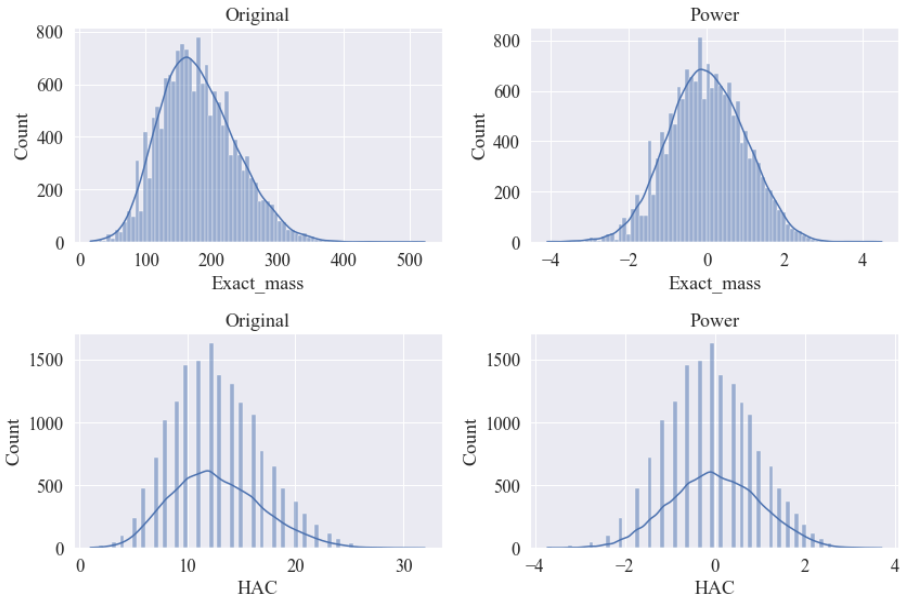
3.1.2. Power Transformer

Power Transformer skalira ulazne varijable na način da stabilizira varijancu, oblikuje izgled dataseta u normalnu raspodjelu (Gaussova distribucija) i značajno smanjuje utjecaj ekstremnih vrijednosti na dataset. Ova metoda skaliranja vrlo je korisna kod asimetričnih podataka koji ne prate Gaussovu distribuciju.

Ova metoda skaliranja u Pythonu ima dva načina pretvorbe: *Box-Cox* transformaciju i *Yeo-Johnson* transformaciju. Spomenute transformacije funkcioniraju na sličan način no prednost *Yeo-Johnson* pretvorbe je u tome što može transformirati i pozitivne i negativne vrijednosti za razliku od *Box-Cox* koja pretvara striktno pozitivne vrijednosti. *Yeo-Johnson* transformacija definirana je kao:

$$y(\lambda) = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda} & \text{za } \lambda \neq 0\,,\, x \geq 0 \\ \ln(x+1) & \text{za } \lambda = 0\,,\, x \geq 0 \\ \frac{(1-x)^{2-\lambda}-1}{2-\lambda} & \text{za } \lambda \neq 2\,,\, x < 0 \\ -\ln(1-x) & \text{za } \lambda = 2\,,\, x < 0 \end{cases} \tag{3.2}$$

cilj ove transformacije je stabilizirati varijancu i normalizirati distribuciju koristeći *power* funkciju određenu parametrom λ.



Slika 3.2. Histogram za dvije ulazne varijable prije i poslije Power transformer skaliranja

Na slici 3.2 prikazan je histogram za dvije ulazne varijable, te se i grafičkog prikaza jasno vidi utjecaj *Power Transformer-a*, orginalni dataset je preoblikovan tako da slijedi Gaussovu distribuciju.

3.1.3. Normalizer

Za razliku od ostalih metoda skaliranja *Normalizer* ne preoblikuje dataset s ciljem da podaci imaju određen raspon i varijancu, već transformira podatke na način da ih opisuje jedinični vektor. Ova je metoda vrlo korisna kada je usmjerenost vektora podataka važnija od njihove magnitude. Najčešće se primjenjuje u klasifikacijskim problemima i definirana je kao:

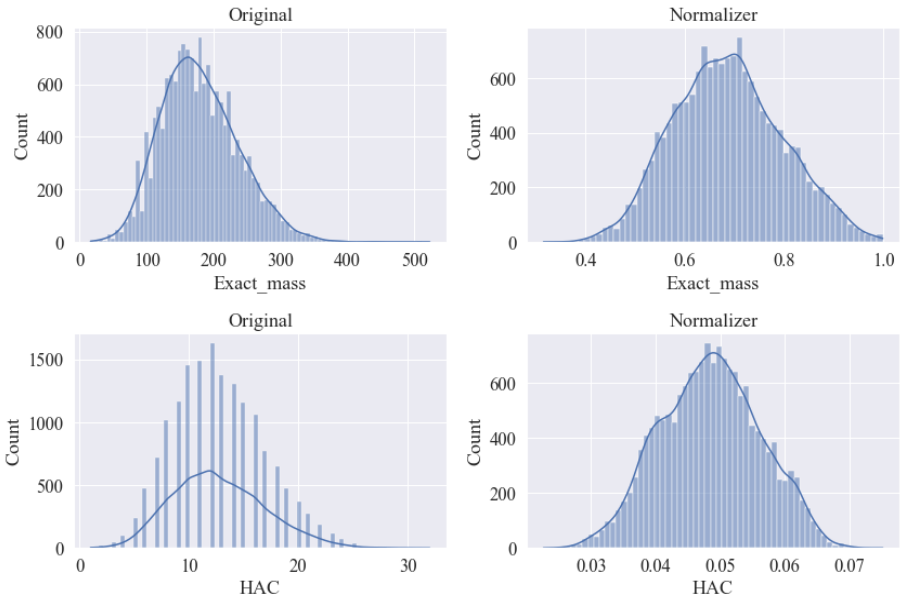
$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{3.3}$$

$$\mathbf{u} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \tag{3.4}$$

gdje za normalizaciju vektora ulazne varijable $\mathbf{x} = [x_1, x_2, ..., x_n]$ koristimo izraz 3.3, dok je jedinični vektor $\mathbf{u}$ definiran izrazom 3.4.

Na slici 3.3 moguće je vidjeti kako izgledaju dvije ulazne varijable prije i poslije korištenja *Normalizer()* skaliranja. Ova metoda daje prednost usmjerenosti vektora nad magnitudom, što je korisno za neke od algoritama umjetne inteligencije kao što je KNN *(K-nearest neighbors)* i

različite probleme klasifikacije. Budući da se u ovom radu promatra regresijski problem, očekivano je da ova metoda skaliranja neće pružiti najbolje rezultate.



Slika 3.3. Histogram za dvije ulazne varijable prije i poslije Normalizer skaliranja

3.1.4. Robust Scaler

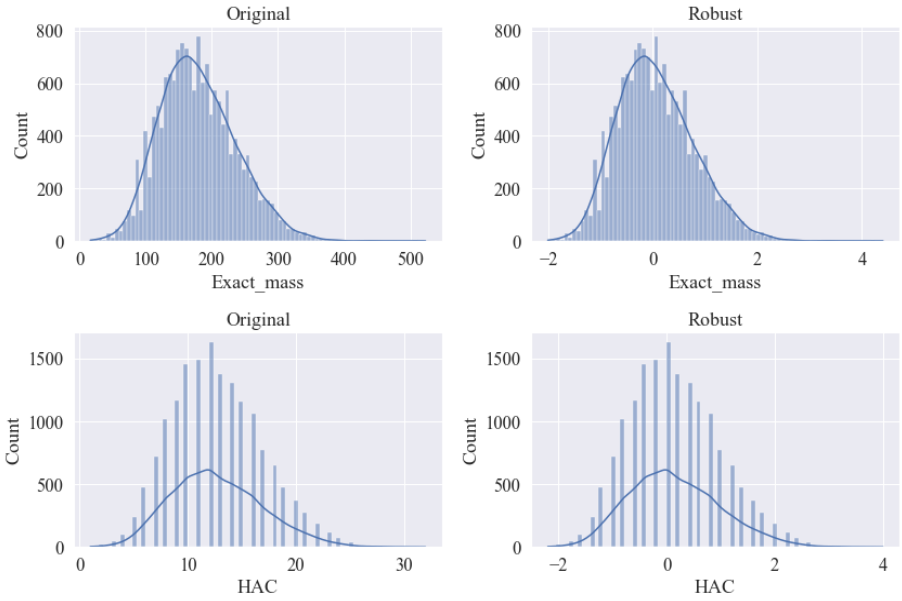
Robust Scaler je metoda skaliranja koja je izrazito otporna na postojanje ekstremnih vrijednosti u datasetu, pri svojoj transformaciji koristi statističke podatke na koje *outlier-i* ne utječu, kao što su medijan i percentili. Za skaliranje koristi izraz:

$$x_{scaled} = \frac{x - medijan}{IQR} \tag{3.5}$$

gdje je IQR dobiven razlikom između trećeg i prvog kvartila.

Na slici 3.4 možemo vidjeti da je 50% vrijednosti ulaznih varijabli u rasponu od -1 do 1. Prednosti *Robust Scaler-a* su u tome što utjecaj ekstremnih vrijednosti ne dolazi do izražaja, ova metoda skaliranja pokazuje najbolje rezultate za datasetove koji sadrže ekstremne vrijednosti i ne prate Gaussovu distribuciju.

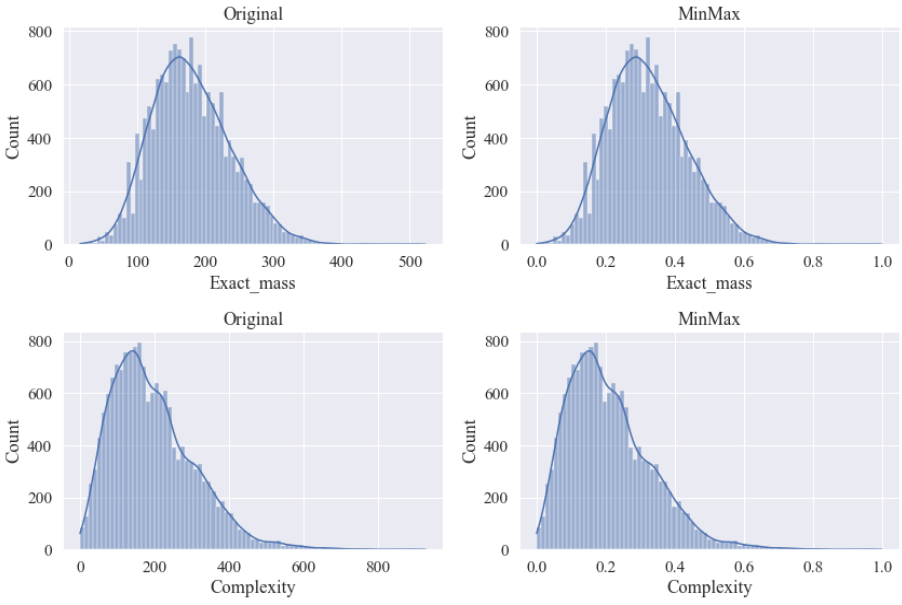
Nedostatak ove metode je to što skaliranje ekstremnih vrijednosti također podrazumijeva i gubitak informacija o varijabilnosti podataka. Nadalje ako vrijednosti u datasetu prate Gaussovu distribuciju i ne sadrže velik broj ekstremnih vrijednosti, tada ostale metode skaliranja pokazuju bolje rezultate.



Slika 3.4. Histogram za dvije ulazne varijable prije i poslije Robust skaliranja

3.1.5. MinMax Scaler

MinMax Scaler metoda skaliranja funkcionira na način da transformira sve vrijednosti dataseta u zadani raspon, najčešće $[0, 1]$. Za skaliranje koristi najmanju i najveću vrijednost ulazne varijable, pritom skalira proporcionalno i odnos između vrijednosti ostaje sačuvan.



Slika 3.5. Histogram za dvije ulazne varijable prije i poslije MinMax skaliranja

Za skaliranje u željeni raspon koristi se izraz:

$$x_{scaled} = a + \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right) (b - a) \tag{3.6}$$

gdje su a i b, granice željenog raspona, a x_{min} i x_{max} najmanja i najveća vrijednost ulazne varijable.

Na slici 3.5 možemo vidjeti da su sve vrijednosti ulazne varijable nakon *MinMax* transformacije u rasponu $[0, 1]$ i usporedbom s orginalnom vrijednosti ulazne varijable možemo uočiti da je transformacija proporcionalna, te su sačuvani omjeri vrijednosti.

Nedostatak *MinMax* metode skaliranja je u tome što za transformaciju koristi minimalnu i maksimalnu vrijednost, te dvije veličine su pod znatnim utjecajem ekstremnih vrijednosti, što u nekim slučajevima može dovesti do iskrivljenog skaliranja vrijednosti. U slučaju da dataset sadrži značajan broj *outlier-a*, prednost se daje robusnijim metodama skaliranja.

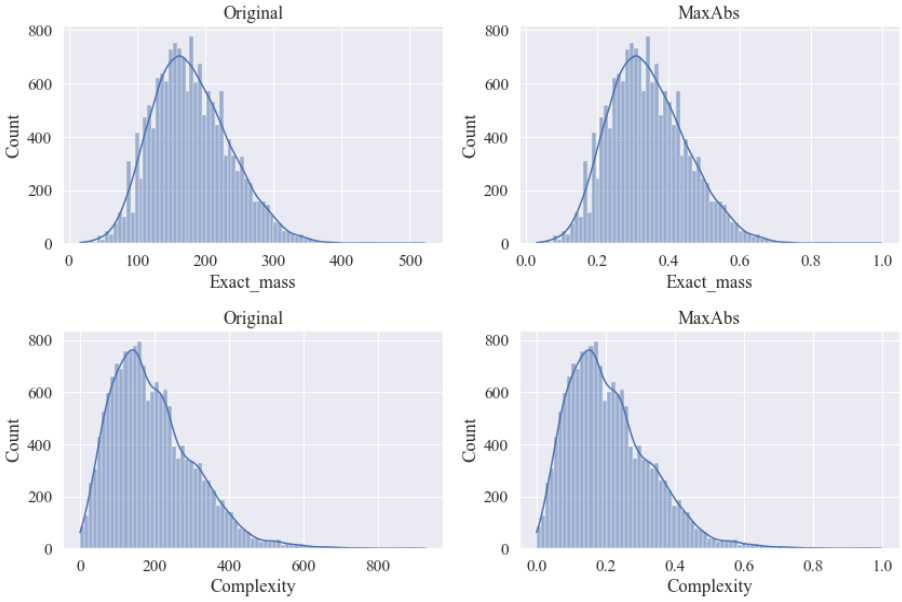
3.1.6. MaxAbs Scaler

MaxAbs Scaler za skaliranje koristi maksimalnu apsolutnu vrijednost ulazne varijable, to osigurava da raspon vrijednosti bude u intervalu $[-1, 1]$. Ova se metoda često primjenjuje za podatke koji su centrirani oko nule, te sadrže pozitivne i negativne vrijednosti.

U slučaju da se dataset sastoji od striktno pozitivnih vrijednosti i minimalna vrijednost svake ulazne varijable iznosi 0, *MaxAbs* i *MinMax* skaliranje rezultira jednakom transformacijom. Promatrajući sljedeći izraz za *MaxAbs* transformaciju:

$$x_{scaled} = \frac{x}{max_{abs}(x)} \tag{3.7}$$

možemo doći do zaključka da ekstremne vrijednosti imaju negativan utjecaj na ovu metodu skaliranja budući da za skaliranje koristi apsolutnu maksimalnu vrijednost ulazne varijable.



Slika 3.6. Histogram za dvije ulazne varijable prije i poslije MaxAbs skaliranja

Na slici 3.6 možemo vidjeti kako izgledaju dvije ulazne varijable prije i poslije primjene *MaxAbs* skaliranja. Ukoliko usporedimo dobivene rezultate s rezultatima *MinMax* skaliranja na slici 3.5, možemo uočiti da je u oba slučaja raspon vrijednosti u intervalu $[0, 1]$, budući da se dataset sastoji od striktno pozitivnih vrijednosti. Dobiveni rezultati ove dvije metode skaliranja vrlo su slični, jedina razlika se pojavljuje za ulazne varijable čija minimalna vrijednost nije nula, kao što je primjerice molekularna težina *(Exact_mass)*.

3.2. Algoritmi umjetne inteligencije

U strojnom učenju postoje razni pristupi problemu, koju od metoda umjetne inteligencije koristiti primarno ovisi o razmatranom problemu. U ovom radu rješavamo problem regresije kod kojeg je cilj precizno odrediti numeričku vrijednost izlazne varijable, odnosno osnovnu energetsku razinu molekule, na osnovu nekoliko ulaznih varijabli. Za rješavanje ovog regresijskog problema odabrane su ansambl metode.

Ansambl metode predstavljaju napredne algoritme strojnog učenja, sastoje se od jednog ili više osnovnih modela i završnog meta regresora. Ideja korištenja ansambl metoda je kombinacija više jednostavnih modela za rješavanje složenog problema. Kombinacijom većeg broja osnovnih regresora ostvaruje se veća preciznost nego kod individualnih modela.

Ansambl metode koriste tehnike kao što su *bagging*, *boosting*, *stacking* i *voting* za drugačiji pristup strojnom učenju. Svaki od osnovnih i meta regresora, kao i tehnike koje se koriste imaju svoje prednosti i nedostatke, te će u nastavku biti detaljno opisani.

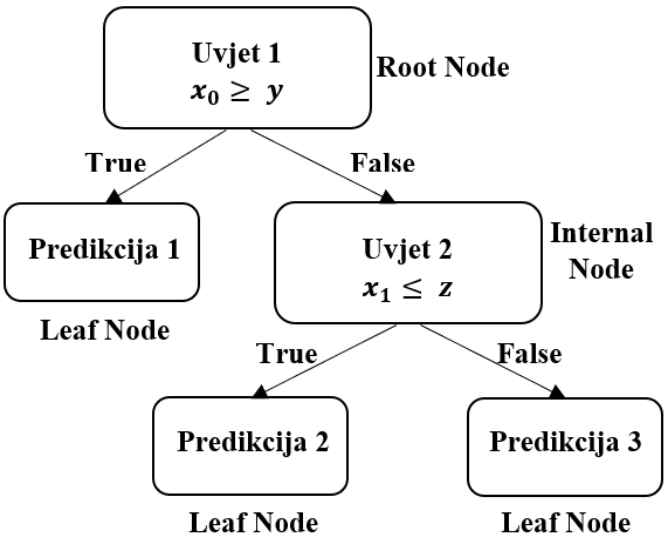
3.2.1. Decision Tree Regressor

Decision Tree Regressor je osnovni regresor koji ne spada pod ansambl metode, no predstavlja osnovni temeljni element na kojemu su građeni ostali ansambl regresori. Kao što i naziv govori strukturu *Decision Tree Regressor* možemo prikazati kao stablo odluka.

Struktura prikazana na slici 3.7 jednostavno prikazuje funckionalnost *Decision Tree* regresora. Stablo se sastoji od korijena *(eng. Root Node)* koji postavlja prvi uvjet. Uspoređuje ulazne varijable s određenom brojčanom vrijednosti. Ukoliko uzorci zadovoljavaju uvjet granaju se u lijevi unutarnji čvor *(eng. Internal Node)*, u suprotnome granaju se u desni unutarnji čvor. Postoje dvije vrste čvorova, oni koji postavljaju novi uvjet za neku drugu ulaznu varijablu i nastavljaju se dalje granati, te oni koji završavaju s grananjem i služe za određivanje vrijednosti izlazne varijable s određenom čistoćom *(eng. purity)* odnosno preciznošću. Takvi se čvorovi nazivaju završnim čvorovima *(eng. Leaf Node)*.

Prvi korak *Decision Tree* regresora je pronalazak najboljeg početnog uvjeta koji djeli dataset na način da pruža najviše korisnih informacija. Proces grananja i djeljenja dataseta se nastavlja do

određene dubine koja je zadana ili dok se ne ostvare potpuno čisti završni čvorovi koji precizno određuju iznos izlazne varijable.



Slika 3.7. Struktura Decision Tree regresora

Za određivanje najbolje podijele dataseta koristi se izraz 3.9, koji izračunava smanjenje varijance. Za odrediti promjenu u varijanci potrebno je izračunati varijancu prema izrazu 3.8 za orginalni dataset i za podatke nastale grananjem. Algoritam izračunava smanjenje varijance za svaku moguću podjelu i kao najbolja podjela uzima se ona koja ima najveće smanjenje varijance.

Vrijedi:

$$\text{Var} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{3.8}$$

$$\Delta\text{Var} = \text{Var}(S) - \left(\frac{|S_L|}{|S|}\text{Var}(S_L) + \frac{|S_R|}{|S|}\text{Var}(S_R)\right) \tag{3.9}$$

gdje je:

- $\text{Var}(S)$ - varijanca izlazne varijable u korijenu stabla,
- $\text{Var}(S_L)$ - varijanca izlazne varijable u lijevom čvoru,
- $\text{Var}(S_D)$ - varijanca izlazne varijable u desnom čvoru,
- $|S|$ - broj uzoraka u korijenu stabla,
- $|S_L|$ - broj uzoraka u lijevom čvoru i
- $|S_D|$ - broj uzoraka u desnom čvoru.

Nakon treniranja *Decision Tree* regresora stvorena je struktura stabla kroz koju zatim prolaze testni uzorci. Uspoređuju se ulazne varijable i postavljeni uvjeti, prolazeći tako kroz cijelu strukturu sve dok ne završe u *Leaf Node-u* unutar kojeg se zatim vrši predikcija na način da se uzima srednja vrijednost svih uzoraka koji se tamo nalaze.

Prednost *Decision Tree* regresora je u tome što može na jednostavan i intuitivan način opisati nelinearnu regresiju. Nedostatak ovog regresora je njegova fleksibilnost, naime na podacima za treniranje najčešće ostvaruje iznimne rezultate, dok kod podataka za testiranje dolazi do znatno lošijih rezultata. Ova se pojava naziva *overfitting* i do nje dolazi zbog nekoliko razloga. Kako bi stablo precizno određivalo izlaznu varijablu potrebna je primjerice velika dubina stabla *(eng. max depth)*, veliki broj unutarnjih čvorova i visoka čistoća listova.

Postoji nekoliko načina kako smanjiti *overfitting* i unaprijediti ponašanja modela kao što su uklanjanje grana stabla *(eng. Tree Pruning)*, korištenje ansambl metoda koje izbjegavaju nedostatke *Decision Tree* regresora i oslanjaju se na prednosti istog, te optimizacija hiperparametara. Sama problematika procesa pronalaženja optimalnih hiperparametara biti će razmotrena u nekom od nadolazećih poglavlja.

3.2.2. Random Forest Regressor

Ansambl metode se sastoje od većeg broja osnovnih modela koji zajedno pridonose konačnoj predikciji. Za pristup ansambl metodama koristi se Python knjižnica *sklearn.ensemble*. Prva razmatrana ansambl metoda je *Random Forest* regresor, kod kojeg je cilj izgraditi veliki broj osnovnih *Decision Tree* regresora. Algoritam prvo izvodi *bootstrapping* orginalnog dataseta.

Bootstrapping je metoda generiranja modificiranih datasetova za treniranje pojedinih *Decision Tree* regresora, na način da se nasumično *(eng. random)* odabiru redovi orginalnog dataseta i zatim izabrani uzorci popunjavaju novonastali dataset sve dok ne poprimi veličinu orginalnog dataseta. Pojedini redovi mogu biti nasumično odabrani više puta, što tim uzorcima pridaje veći značaj.

Uključuje se i dodatna nasumičnost na način da se pri kreiranju modificiranog dataseta nasumično odabire određen broj ulaznih varijabli koje će se koristiti pri generiranju dataseta. Postavlja se pitanje koliko ulaznih varijabli treba uključiti pri kreiranju novog dataseta. Statistika pokazuje da se najbolji rezultati postižu kada se koristi broj koji je jednak drugom korijenu ili logaritmu baze 2 ukupnog broja ulaznih varijabli.

Svrha generiranja novih datasetova iz postojećeg i korištenje samo određenog broja ulaznih varijabli je izgraditi velik broj različitih i međusobno neovisnih *Decision Tree* regresora. U slučaju da ne koristimo ovakvu nasumičnost izgradili bi stabla koja su jako slična što bi rezultiralo lošim ansamblom. Ansambl se mora sastojati od raznovrsnih i neovisnih modela kako bi poboljšao nedostatke osnovnih regresora.

Kod *Random Forest* regresora uvjeti grananja ostaju isti kao i kod *Decision Tree* algoritma.

Algoritam provjerava svaku moguću podjelu dataseta i traži optimalne uvjete grananja koji donose najviše informacija. Jednom kada su sva stabla izgrađena pri dolasku testnih uzoraka, svi kreirani regresori vrše predikciju i kao konačna estimacija uzima se srednja vrijednost *(eng. average)* predikcija svih regresora.

Zbog nasumičnosti neki regresori biti će trenirani s manje bitnijim ulaznim varijablama te će lošije estimirati izlaznu varijablu, na primjer s vrijednošću koja je manja od tražene. Kada pogledamo takav regresor mogli bi zaključiti da će narušiti preciznosti modela, no budući da se u slučaju *Random Forest* metode koristi velik broj regresora postojati će i oni koji će estimirati vrijednost koja je veća od tražene, te će se takva odstupanja međusobno usrednjiti. Metode koje koriste *bootstrapping* i za konačnu estimaciju uzimaju srednju vrijednost svih regresora nazivaju se *bagging* metode.

Možemo zaključiti da nasumičnost *Random Forest* ansambla poboljšava nedostatke *Decision Tree* regresora na način da čini algoritam robusnijim, fleksibilnijim i manje sklon *overfitting-u*. Nedostatak modela koji se zasnivaju na velikom broju *Decision Tree* regresora je vrijeme treniranja modela i njihova računska zahtjevnost.

3.2.3. Extra Tree Regressor

Extra Tree regresor je ekstremno nasumičan ansambl algoritam koji se sastoji od više osnovnih *Decision Tree* regresora. Princip rada je sličan kao kod *Random Forest* ansambla, no uvodi se dodatna nasumičnost. U ovom se slučaju treniranje algoritma vrši na orginalnom datasetu bez korištenja *bootstrapping* metode.

Za razliku od *Random Forest* algoritma *Extra tree* regresor ne pretražuje sve moguće uvjete grananja, već uzima određen broj nasumičnih podjela dataseta i među njima uzima najbolji uvjet grananja. Ova dodatna nasumičnost uvodi veću varijancu, no značajno smanjuje vrijeme treniranja i računske zahtjeve ove metode.

Konačna estimacija algoritma isto kao i kod *Random Forest* regresora uzima srednju vrijednost estimacija svih osnovnih modela u ansamblu. *Extra Tree* regresor dodatno smanjuje mogućnost *overfitting-a*, vrijeme treniranje algoritma je kraće i manje zahtjevnije, povećava se fleksibilnost, ali se istovremeno smanjuje preciznost modela.

3.2.4. AdaBoost Regressor

AdaBoost regresor je ansambl algoritam koji koristi *boosting* tehniku za rješavanje regresijskih problema. Strukturu ovog ansambla čine veći broj posebnih vrsta *Decision Tree* regresora. Ova posebna stabla nazivaju se *stumps* i građena su od jednog početnog čvora *(Root node)* i dva završna čvora *(Leaf nodes)* koji sadrže predikcije.

Kod *boosting* ansambla koristi se kombinacija više slabijih algoritama u cilju poboljšanja preciznosti modela strojnog učenja. Pod pojmom slabijih algoritama smatraju se jednostavni oblici stabla odluka kao što je *stump*. Budući da se *stump* sastoji od samo jedne odluke ne donosi preciznu estimaciju na složenom datasetu. Ovo svojstvo nije problem kod *AdaBoost* regresora koji zahtjeva kombinaciju slabijih algoritama.

Kreiranje *Stump-ova* započinje postavljanjem početnih težina *(eng. weights)* svakom uzorku u datasetu. Vrijednost koja određuje važnost ili utjecaj određenog podatka pri treniranju ili konačnoj predikciji naziva se težina. Inicijalna vrijednost težine iznosi $1/N$ gdje N predstavlja ukupan broj redova u datasetu, ovime je osigurano da u prvoj iteraciji svi uzorci imaju jednaku važnost.

Za razliku od *Random forest* ansambla, gdje se stabla generiraju neovisno jedno o drugome, *AdaBoost* regresor sekvencijalno generira stabla odluke. Novonastali *stump* direktna je posljedica pogrešaka koje je napravilo prijašnje stablo u ansamblu. Veća težina pridodaje se uzorcima koji su pogrešno estimirani, na ovaj se način model primarno fokusira na složenije slučajeve predikcije izlazne varijable.

Prvi *stump* se kreira na sličan način kao i stablo odluka dosad objašnjenih regresora, traži se najbolja moguća podjela dataseta koja rezultira najboljom predikcijom. Kada algoritam odredi najbolju podjelu slijedi izračun stope pogreške prema izrazu 3.10. Zatim se koristeći izraz 3.11 izračunava težina koju će ovaj *stump* imati pri konačnoj predikcija ansambla:

$$\epsilon_m = \frac{\sum_{i=1}^{n} w_i \cdot |y_i - \hat{y}_i|}{\sum_{i=1}^{N} w_i} \tag{3.10}$$

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right) \tag{3.11}$$

gdje je:

- ϵ_m - stopa pogreške,
- w_i - težina uzorka,
- y_i - stvarna vrijednost izlazne varijable,
- $\hat{y}_i$ - predikcija vrijednosti izlazne varijable i
- α_m - težina pri konačnoj predikciji ansambla.

Sljedeći je korak ponovno odrediti težine uzoraka na način da se veća važnost daje slabije estimiranim uzorcima, kako bi sljedeći *stump* mogao naučiti na pogreškama svog prethodnika. Nove težine se zatim normaliziraju na način da njihov zbroj bude jednak 1. Za kreiranje sljedećeg stabla prvo je potrebno napraviti novi dataset, jedna od mogućih metoda je generiranje nasumičnih brojeva koji će ovisno o težini uzimati uzorke iz orginalnog dataseta. Isti je uzorak moguće uzeti

više puta, što je i očekivano budući da lošije estimirani uzorci imaju veće težine. Na ovaj se način prilikom treniranja novog stabla daje veći značaj složenijim slučajevima predikcije izlazne varijable. Za konačnu predikciju *AdaBoost* ansambla koristi se izraz:

$$F(x) = \sum_{i=1}^{n} \alpha_m \hat{y}_i \tag{3.12}$$

koji uzima u obzir težinske predikcije svih *stump-ova*.

Prednosti *AdaBoost* algoritma dolaze više do izražaja kod klasifikacijskih problema, no i u slučaju regresije pružaju alternativni pristup problemu. Glavna karakteristika *AdaBoost* regresora je savladavanje određenih uzoraka u datasetu koji su izrazito teški za predvidjeti jer na neki način odstupaju od ostatka dataseta. Kombinacijom većeg broja slabijih modela ansambl dobiva na fleksibilnost, robusnosti i smanjuje se rizik od *overfitting-a*. Jedna od prednosti ovog algoritma je i mali broj hiperparametara što omogućuje jednostavniju primjenu i optimizaciju modela. Mogući nedostatak ovog pristupa je to što se velika važnost daje podacima koje je teško predvidjeti i tade se algoritam trenira većinom na *outlier-ima*, što može negativno utjecati na učinkovitost *AdaBoost* ansambla.

3.2.5. HistGradientBoosting Regressor

GradientBoosting ansambl kao i *AdaBoost* koristi *boosting* metodu za rješavanje regresijskih problema. U slučaju *GradientBoost* regresora umjesto *stump-ova* koriste se regresijska stabla. Početno stablo sastoji se samo od jednog čvora koji sadrži početnu aproksimaciju koja je jednaka srednjoj vrijednosti izlazne varijable.

Nakon početne aproksimacije algoritam kreira stabla većih dimenzija čija se struktura obično sastoji od 8 do 32 završna čvora. Kao i kod *AdaBoost* regresora nova stabla bazirana su na pogreškama prijašnjih predikcija. Sljedeće stablo odluke kreira se na osnovu razlike između stvarne i estimirane vrijednosti izlazne varijable. Ova se razlika naziva *pseudo residual* i pri izgradnji novog stabla korsitimo ulazne varijable kako bi ostvarili predikciju te razlike.

Budući da je broj završnih čvorova ograničen, prilikom izgradnje stabla u krajnjim čvorovima nalazit će se veći broj predikcija razlike, vrijednost predikcije takvih čvorova zamijenjuje se srednjom vrijednosti svih predikcija koje se nalaze u tom čvoru. Kako bi se smanjila mogućnost *overfitting-a* sva kreirana stabla skalirana su određenim brojem poznatijim kao *learning rate*.

Za kvalitetno ponašanje ovog ansambla bitna je dobra optimizacija hiperparametara broja osnovnih estimatora i koraka učenja *(learning rate)*. Ukoliko povećamo broj estimatora korak učenja se mora smanjiti, vrijedi i obratno. Ukoliko bi se vrijednosti oba hiperparametra povećala postoji velika mogućnost *overfitting-a*. Optimizacija hiperparametara biti će detaljno objašnjena u nastavku rada.

Za kreiranje sljedećih estimatora u nizu, ponovno je potrebno odrediti razliku između stvarne i estimirane vrijednosti. Ovaj put se za estimiranu vrijednost koristi zbroj osnovne predikcije i skalirana vrijednost predikcije prvog stabla. Ovaj se postupak ponavlja sve dok se ne dostigne maksimalni broj estimatora u ansamblu ili vrijednost razlike nije moguće dodatno smanjiti.

Svakim dodatnim stablom predikcija se malim korakom približava željenoj vrijednosti. Empirijska istraživanja pokazuju da uzimanje mnogo malih koraka u pravom smjeru rezultira značajno boljom predikcijom na testnim podacima, odnosno smanjuje varijancu [9]. Konačna estimacija *GradientBoosting* ansambla dobivena je zbrojem inicijalne predikcije i skaliranih predikcija svih estimatora u ansamblu.

HistGradientBoosting ansambl predstavlja posebnu vrstu *GradientBoosting* algoritma koji je namijenjen za datasetove s iznimno velikim brojem podataka. Kod izgradnje regresijskih stabla umjesto da se koriste sve vrijednosti ulaznih varijabli, *HistGradientBoosting* koristi poseban pristup u kojemu se vrijednosti ulaznih varijabli pohranjuju u određen broj intervala. Na ovaj se način kontinuirane veličine dijele u nekoliko diskretnih intervala. Podijelom ulaznih podataka u intervale značajno se povećava brzina i smanjuje računska zahtjevnost modela, te se dodatno povećava učinkovitost *GradientBoosting* metode kod velikih datasetova.

3.2.6. Extreme GradientBoosting Regressor

Extreme GradientBoosting regresor poznatiji kao *XGBoost* je ansambl metoda dizajnirana za izrazito velike i složene datasetove. *XGBoost* je vrlo napredan ansambl algoritam koji koristi velik broj hiperparametara za bolje treniranje i regulizaciju modela. Kao i *AdaBoost* koristi velik broj slabijih algoritama i svako novo stablo kreira se na osnovu pogrešaka koje je napravilo prethodno. Unutar ansambla koriste se *XGBoost* stabla koja predstavljaju posebnu vrstu *Decision Tree* algoritma.

XGBoost koristi i metode kao što su uklanjanje grana *(Tree Pruning)*, *cross-validation* i rano zaustavljanje u cilju stvaranja kompletnog algoritma koji se izrazito brzo trenira i sadrži kompleksnu strukturu za ostvarivanje preciznih predikcija kod velikih datasetova. *XGBoost* ansambl je izrazito složen algoritam koji sadrži mnogo različitih načina treniranja, veći broj mogućih funkcija gubitaka i regulizacija, te načina kreiranja *XGBoost* stabla. U nastavku će biti objašnjen jedan od često korištenih pristupa.

Kao i kod *AdaBoost* algoritma, prvo se postavlja jedan završni čvor koji predstavlja inicijalnu predikciju. Vrijednost inicijalne predikcije u ovom slučaju je proizvoljna, u većini slučajeva uzima se srednja vrijednost izlazne varijable. Za izgradnju prvog *XGBoost* stabla prvo se uspoređuju stvarna vrijednost i inicijalna predikcija izlazne varijable. Dobivena razlika predstavlja osnovu za izgradnju novog stabla.

Vrijednosti razlika spremaju se u početni čvor za koji se zatim računa ocjena kvalitete prema

izrazu 3.13. Sljedeće što algoritam razmatra je može li se podijelom dobivenih razlika u dva nova čvora postići bolji rezultati. Provjeravaju se svi mogući uvjeti grananja podataka i za svaku podjelu izračunavaju se ocjene kvalitete za novonastale čvorove. Slijedi izraz za ocjenu kvalitete:

$$Q = \frac{(\sum_{i=1}^{n} R_i)^2}{n + \lambda} \tag{3.13}$$

gdje je:

- R_i - vrijednost razlike,
- n - ukupan broj uzoraka,
- Q - ocjena kvalitete i
- λ - parametar regulacije.

Za provjeru dobitka podijele koristi se izraz 3.14, uvjet koji pruža najveći *Gain* smatra se najboljom podjelom i postaje početni čvor stabla. Algoritam zatim provjerava broj vrijednosti razlika u novonastalim čvorovima, ako se u njima nalazi više od jedne vrijednosti, ponovno se razmatra podijela takvih čvorova u dva nova čvora. Onaj uvjet koji pruža najveći *Gain* postaje novi unutarnji čvor odluke. Ovaj se proces nastavlja dok svi krajnji čvorovi ne sadrže jednu vrijednost razlike ili dok se ne dostigne maksimalna veličina stabla *(max depth)*. Izraz za provjeru dobitka definiran je kao:

$$\text{Gain} = Q(L) + Q(D) - Q(P) \tag{3.14}$$

gdje je:

- $Q(L)$ - ocjena kvalitete za lijevi čvor,
- $Q(D)$ - ocjena kvalitete za desni čvor i
- $Q(P)$ - ocjena kvalitete za početni čvor.

Sljedeće što algoritam provjerava je korisnost grana unutar *XGBoost* stabla, na način da se vrijednost *Gain-a* najniže grane uspoređuje s hiperparametrom γ. Ovaj hiperparametar predstavlja prag koji korisnost grane mora prijeći. Ukoliko je razlika između vrijednosti *Gain-a* grane i praga γ pozitivna, stablo ostaje kompletno. U slučaju da je razlika negativna dolazi do orezivanja te grane i zatim se provjerava korisnost sljedeće grane u *XGBoost* stablu. Ovakvim je orezivanjem moguće u potpunosti ukloniti cijelo stablo. Veće vrijednosti γ dovode do ekstremnijeg orezivanja stabla.

Još jedan iznimno važan hiperparametar *XGBoost* ansambla je parametar regulacije λ, koji smanjuje osjetljivost na pojedine uzorke. Ocjena kvalitete je obrnuto proporcionalna vrijednosti λ. Povećanjem parametra regulacije λ smanjuje se ocjena kvalitete proporcionalno s brojem uzoraka u čvoru, također se smanjuje i vrijednost *Gain-a* što možemo vidjeti iz izraza 3.13 i 3.14, što dovodi do ekstremnijeg orezivanja stabla.

Konačna predikcija *XGBoost* ansambla jednaka je zbroju osnovne predikcije i skaliranoj vrijednosti estimacije svakog stabla u ansamblu. Vrijednost s kojom se predikcije skaliraju η predstavlja *learning rate*. Ukoliko se predikcija pojedinih *XGBoost* stabla sastoji od više uzoraka tada se uzima njihov zbroj i dijeli s zbrojem hiperparametra λ i ukupnog broja uzoraka koji se u tom završnom čvoru nalaze.

Optimizacija hiperparametara λ i γ ključna je za postizanje preciznih rezultata, povećanjem ovih parametara smanjuje se *overfitting* na uzorcima za treniranje što dovodi do veće fleksibilnosti modela pri predikciji novih uzoraka, ali ujedno povećava varijancu. Ovaj kompromis između pristranosti modela uzorcima za treniranje i varijance na podacima za testiranje biti će razmotren u sljedećem poglavlju.

3.2.7. Bagging, Voting i Stacking Regressor

Posljednje korištene metode u radu su posebne vrste algoritama koji koriste jednu od ansambl tehnika *bagging*, *voting* ili *stacking* na druge već optimizirane ansambl algoritme koje smo do sad analizirali. Cilj ovih metoda je poboljšati rezultate pojedinih ansambl algoritama njihovom kombinacijom. Prednosti jednog ansambla mogu nadoknaditi nedostatke drugih. U nastavku slijedi opis svake od ansambl tehnika koje ovi algoritmi koriste.

Bagging je ansambl tehnika koja je već objašnjena u sklopu *Random Forest* regresora, ukratko unapređuje algoritam korištenjem *bootstrapping* metode i za konačnu estimaciju koristi srednju vrijednost estimacije svih algoritama unutar ansambla. Prednosti *Bagging* regresora su poboljšanje stabilnosti i robusnosti modela, te smanjivanje varijance i rizika od *overfitting-a*, no te prednosti dolaze s nedostatcima kao što su povećanje vremena treniranja modela i dodatna računska zahtjevnost.

Voting regresor koristi tehniku glasanja pri konačnoj predikciji, pri treniranju ovog algoritma nasumično se određuju težine koje se pridodaju pojedinim optimiziranim algoritmima unutar *Voting* ansambla. One težine koje pri treniranju pružaju najbolje rezultate uzimaju se kao parametri *Voting* ansambla. Konačnu predikciju predstavlja srednja vrijednost težinskih predikcija pojedinih algoritama u ansamblu.

Stacking anasmbl se sastoji od više optimiziranih algoritama i jednog finalnog regresora. Treniranje se izvodi pomoću *cross* validacije tako da se dataset podijeli na K dijelova. Svaki se osnovni algoritam trenira na $K-1$ dijelovima dataseta, dok se poslijednji dio koristi za validaciju predik-

cije pojedinih modela. Dobiveni rezultati služe kao ulazna varijabla u finalni regresor, koji može biti bilo koji od dosad spomenutih algoritama ili bilo koja vrsta linearnih modela. Cilj finalnog regresora je na osnovu predikcija osnovnih algoritama unutar ansambla precizno odrediti izlaznu varijablu.

3.3. Evaluacijska metrika

Evaluacijske metrike pružaju korisne informacije o ponašanju modela, ukazuju na to koliko precizno model opisuje izlaznu varijablu. U strojnom učenju koriste se brojne metrike za određivanje i uspoređivanje kvaliteta različitih algoritama. U nastavku je dan sadržajan pregled metrika koje su korištene za evaluaciju ansambl regresora.

R2 *(R squared)* je statistička metrika koja pokazuje koliki postotak varijance izlazne varijable model precizno opisuje. Poprima vrijednosti u rasponu $[0, 1]$, u rijetkim slučajevima može biti negativan. Vrijednosti blizu 0 označava da model loše opisuje varijancu izlazne varijable, dok bi vrijednost jednaka 1 označavala da model savršeno opisuje varijancu izlaznu varijablu. Ova se metrika još naziva determinacijskim koeficijentom i izrazito je jednostavna za interpretaciju. Iznos R2 moguće je izračunati sljedećim izrazom:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.15}$$

gdje je:

- y_i - stvarna vrijednost izlazne varijable,
- $\hat{y}_i$ - estimirana vrijednost izlazne varijable,
- $\bar{y}$ - srednja vrijednost stvarnih vrijednosti izlazne varijable i
- n - ukupan broj uzoraka.

Mean Absolute Error (MAE) predstavlja jednu od najjednostavnijih evaluacijskih metrika, izraz 3.16 pokazuje da je to mjera za srednju apsolutnu vrijednost razlike između stvarne i estimirane vrijednosti:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.16}$$

ova metrika ne kažnjava dodatno velike pogreške, ima jednaku mjernu jedinicu kao i izlazna varijabla što je čini jednostavnom za interpretaciju i uspoređivanje performansi različitih algoritama. Što je vrijednost MAE metrike manja to je kvaliteta predikcije regresora veća.

Mean Absolute Percentage Error (MAPE) je vrlo često korištena metrika u strojnom učenju, pokazuje postotak pogreške predikcije modela. Omogućuje jednostavnu usporedbu različitih algoritama. Mogući nedostatak ove metrike je u slučaju kada postoji nekoliko ekstremnih vrijednosti pogrešaka, tada srednja vrijednost ne reprezentira precizno sve uzorke. Dodatno promatrajući izraz 3.17 za izračun MAPE:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.17}$$

možemo doći do zaključka da ukoliko su stvarne vrijednosti izlazne varijable jednake ili blizu 0 tada se javlja problem dijeljenja s 0 ili jako malom vrijednosti što će rezultirati izrazito velikim postotokom pogreške. Manja vrijednost MAPE ukazuje na veću preciznost modela.

Mean Squared Error (MSE) je evaluacijska metrika koja pokazuje srednju vrijednost odstupanja estimirane i stvarne vrijednosti:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.18}$$

ova je metrika osjetljiva na velike pogreške, te ih dodatno kažnjava za razliku od spomenute MAE metrike koja ima manju osjetljivost na veće pogreške. Za izračun MSE koristi se izraz 3.18.

Root Mean Squared Error (RMSE) predstavlja drugi korijen MSE, što možemo vidjeti iz sljedećeg izraza:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.19}$$

budući da se RMSE i MSE računaju gotovo sličan način prednosti i nedostatci ovih metrika su jednake. U većini slučajeva prednost se daje RMSE zbog lakše interpretacije i usporedbe s stvarnim vrijednostima izlazne varijable.

Kling-Gupta Efficiency (KGE) je evaluacijska metrika dizajnirana za davanje uravnotežene procjene kvalitete modela. Izračunava se na temelju izraza 3.20, u obzir uzima tri komponente: koeficijent korelacije, pristranost *(eng. bias)* i varijancu:

$$\text{KGE} = 1 - \sqrt{(R-1)^2 + (\alpha - 1)^2 + (\beta - 1)^2} \tag{3.20}$$

gdje je:

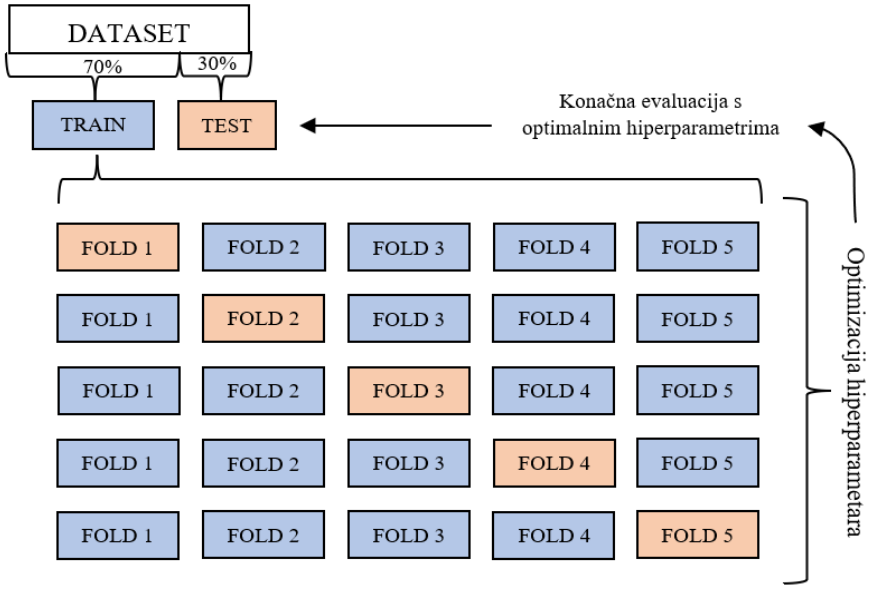
- R - Pearsonov koeficijent korelacije,
- α - omjer standardne devijacije (σ) estimirane i stvarne vrijednosti $\alpha = \frac{\sigma_e}{\sigma_r}$ i

- β - omjer srednjih vrijednosti (μ) estimirane i stvarne vrijednosti $\mu = \frac{\mu_e}{\mu_r}$.

KGE nije često korištena metrika, no pruža dobar uvid u stabilnost modela. Koeficijent korelacije, pristranost i varijanca ključni su pokazatelji kvalitete modela, KGE metrika uključuje ove pokazatelje i rješava ograničenja s kojima se susreću tradicionalne metrike bazirane na jednom pokazatelju. Što je vrijednost KGE bliža 1 to je kvaliteta modela veća, odnosno smatra se da model ima visoku preciznost pri estimacije izlazne varijable, dobru korelaciju, nisku pristranost i odgovarajuću varijancu. Vrijednosti KGE manje ili jednake 0.75 ukazuju na lošu stabilnost, *overfitting* i nisku preciznost modela.

3.4. Treniranje algoritama umjetne inteligencije

Korištenjem inicijalnog skaliranja dobiveno je 6 modificiranih dataseta, ako tome pribrojimo orginalni dolazimo do ukupno 7 skupina podataka. Na svakoj će se od ovih varijacija dataseta provesti treniranje algoritama umjetne inteligencije. Postupak treniranja koristeći 5-fold cross-validaciju za pretraživanje optimalnih hiperparametara prikazan je na slici 3.8.



Slika 3.8. Struktura 5-fold cross validacije

Prvo se za inicijalno treniranje svaki dataset podijeli na *training* i *testing* set. Od toga se 70% koristiti za treniranje modela, dok se ostalih 30% koristi za testiranje. U Pythonu se za ovu podjelu koristi naredba *train_test_split* iz *sklearn* knjižnice. Inicijalno se treniraju sve dosad objašnjene ansambl metode. Treniranje podrazumijeva pronalazak optimalnih hiperparametara, procjenu kvalitete modela cross-validacijom i evaluacijskim metrikama.

Optimizacija hiperparametara je proces pronalaženja najefikasnije skupine parametara za određeni algoritam. Hiperparametri su vrijednosti koje opisuju strukturu algoritma i njihova je optimizacija ključna za postizanje visoke kvalitete modela. Ispravne vrijednosti hiperparametara mogu

značajno poboljšati pristranost, robusnost, varijancu i smanjiti mogućnosti *overfitting-a*, što poboljšava rezultate modela kod predikcije novih podataka. Postoji više različitih metoda pretraživanja optimalnih hiperparametara. U ovom su radu korištene metode *Grid Search* i *Randomized Search*.

Randomized Search je metoda u kojoj se iz skupine hiperparametara nasumično pretražuju različite kombinacije. Prednost ove metode je to što može brzo pronaći dobre hiperparametre neovisno o broju mogućih kombinacija. Za razliku od ostalih metoda, učinkovitost *Randomized Search* metode ne opada s rasponom i brojem mogućih hiperparametara. Nedostatak ove metode je to što se provjerava samo određen broj nasumičnih kombinacija i postoji mogućnost da se pri nasumičnom odabiru optimalna kombinacija hiperparametara preskoči.

Grid Search je iscrpna metoda pretraživanja optimalnih hiperparametara, model se trenira sa svakom mogućom kombinacijom zadanih vrijednosti hiperparametara. Ova je metoda jednostavna za implementaciju i garantira pronalazak najbolje kombinacije hiperparametara unutar zadane skupine. Nedostatci *Grid Search* metode je to što efikasnost opada s ukupnim brojem hiperparametara koji se razmatraju. Kod velikih količina mogućih kombinacija pretraga optimalnih hiperparametara je izrazito računski i vremenski zahtjevna.

Tablica 3.1. Pretraživanje optimalnih parametara za Extra Tree regresor

ExtraTree Regressor		
Hiperparametar	Moguće vrijednosti	Ukupan broj
n_estimators	300, 400, 500, 600, 700, 800, 900	7
max features	'sqrt', 'log2', 'None'	3
max depth	10, 20, 30, 40, 50, 60, 70, 80, 90, 100	10
min_sample_split	2,3,4,5	4
min_sample_leaf	1,2,3,4	4
bootstrap	false, true	2
Ukupan broj kombinacija:		6720

Cross-validacija je standardno korištena tehnika u strojnom učenju. Podrazumijeva podijelu dataseta na K jednakih dijelova *(eng. Folds)*. Model se trenira i validira K puta, svaki se put koristi drugi fold za validaciju, dok ostatak $K-1$ foldova služe za treniranje modela. Na slici 3.8 možemo vidjeti kako izgleda podjela foldova kod 5-fold cross-validacije.

Na kraju validacije uzima se srednja vrijednost promatranih evaluacijskih metrika (R2, MSE, RMSE, MAE, MAPE). Unakrsna validacija pruža dobru estimaciju robusnosti modela. U sklopu ovoga rada uključena je analiza cross-validacije u kojoj se ispituje utjecaj veličine foldova na točnost estimacije ansambl modela.

Nakon korištenja metoda pretraživanja optimalnih hiperparametara nekoliko puta za svaki dataset odabiru se oni hiperparametri s kojima se dobiva najviša vrijednost R2 i najniže vrijednosti MSE, RMSE , MAE, MAPE. U nastavku je dan primjer pretraživanja optimalnih hiperparametara za neke od ansambl algoritama. Radi preglednosti prikazana su samo tri modela.

Tablice 3.1, 3.2 i 3.3 prikazuju *grid* u kojem se nalaze hiperparametri i raspon vrijednosti koje mogu poprimiti. Koristeći *Grid Search* metodu provjerava se svaka moguća kombinacija hiperparametara, te se zapisuje ona koja postiže najbolje rezultate. Ovo pretraživanje optimalnih hiperparametara napravljeno je za sve dosad analizirane algoritme na svim modificiranim datasetovima.

Tablica 3.2. Pretraživanje optimalnih parametara za AdaBoost regresor

AdaBoost Regressor		
Hiperparametar	Moguće vrijednosti	Ukupan broj
n_estimators	100, 200, 300, 400, 500, 600, 700, 800, 900	9
learning rate	0.01, 0.1, 0.2, 0.5, 0.8, 0.9, 1, 1.2, 1.5	9
Ukupan broj kombinacija:		81

Tablica 3.3. Pretraživanje optimalnih parametara za XGBoost regresor

XGBoost regressor		
Hiperparametar	Moguće vrijednosti	Ukupan broj
n_estimators	300, 400, 500, 600, 700, 800, 900,1000	8
learning rate	0.01, 0.05, 0.1	3
max depth	4, 5, 6, 7	4
subsample	0.8, 0.9, 1	3
colsample_bytree	0.8, 0.9, 1	3
alpha	0, 0.1, 0.5, 1	4
lambda	0, 0.1, 0.5, 1	4
gamma	0, 0.1, 0.5, 1	4
Ukupan broj kombinacija:		55296

Možemo uočiti da se *AdaBoost* ansambl, kao što je u analizi tog regresora istaknuto, sastoji od malog broja hiperparametara, dok je kod *XGBoost* ansambla taj broj znatno veći zbog složenosti algoritma. Nakon korištenja *Grid Search* metode dobivene su optimalne kombinacije hiperparametara u definiranom *grid-u*. Za dodatno poboljšanje kvalitete modela, stvoren je novi *grid* s vrijednostima koje se nalaze u malom rasponu oko dobivenih optimalnih vrijendosti. Za pretraživanje novog grida korištena je *Randomized Search* metoda. U tablici 3.4 možemo vidjeti koliko iznose konačni optimalni hiperparametri za *Extra Tree* regresor.

Tablica 3.4. Optimalni hiperparametri za Extra Tree Regressor za sve datasetove

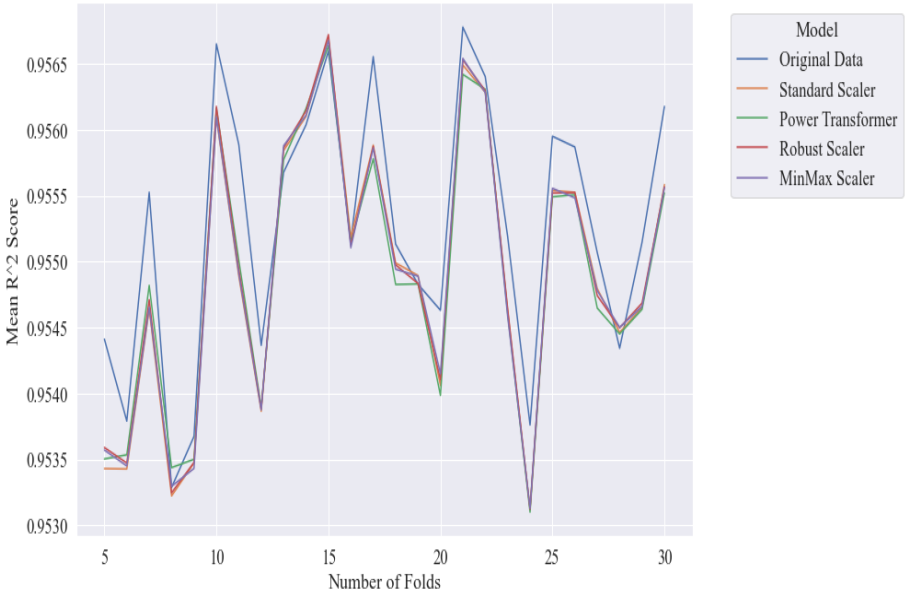
Extra Tree Regressor							
Hiperparametar	Original	Standard	Power	Normalizer	Robust	MinMax	MaxAbs
n_estimators	900	790	790	700	651	966	966
max features	1.0	1.0	1.0	None	None	1.0	1.0
max depth	50	54	54	50	65	44	44
min_sample_split	2	2	2	5	3	2	2
min_sample_leaf	1	1	1	2	1	1	1
bootstrap	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Iz tablice 3.4 možemo vidjeti da se hiperparametri za pojedine skalirane datasetove mijenjaju. Broj stabla *Extra Tree* regresora određen je hiperparametrom *n_estimators*, *max_features* predstavlja broj ulaznih varijabli koje se promatraju pri podjeli dataseta. Vrijednost *None* označava da se u obzir uzimaju sve ulazne varijable. Hiperparametar *max_depth* predstavlja veličinu stabla, odnosno maksimalni broj grana i čvorova. *Min_sample_split* određuje najmanji broj uzoraka u unutarnjem čvoru pri kojem dolazi do grananja tog čvora, dok *min_sample_leaf* određuje najmanji broj uzoraka potrebnih da čvor postane *leaf node*. Posljednji parametar određuje hoće li se koristiti *bootstrap* metoda ili će se treniranje odvijati na orginalnom datasetu.

U dodatku A prikazan je programski kod koji je korišten za incijalno pretraživanje optimalnih hiperparametara *XGBoost* regresora. Optimalni hiperparametri su na isti način određeni za sve spomenute algoritme na svim varijacijama dataseta. U sljedećem djelu rada promotrit ćemo dobivena rješenja i usporediti sve modele koristeći evaluacijske metrike. Isto tako u nastavku se nalazi analiza utjecaja veličine foldova na točnost ansambl algoritama. U dodatku B prikazan je programski kod koji je korišten za analizu utjecaja veličine foldova unakrsne validacije na točnost estimacije.

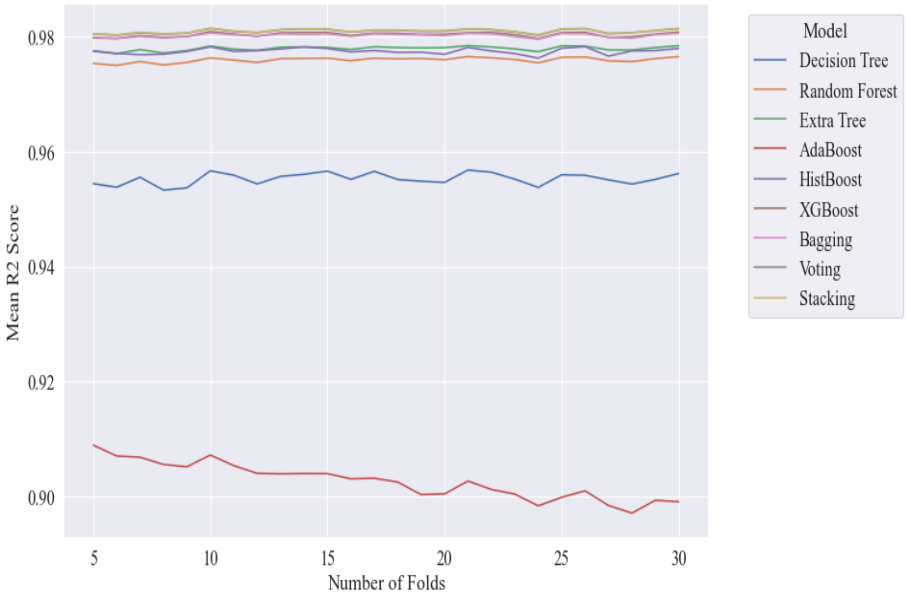
4. Rezultati

U ovom će poglavlju biti prezentirani dobiveni rezultati. Analiza rezultata i usporedba kvaliteta ansambl algoritama međusobno i sa rezultatima prezentiranim u literaturi nalazit će se u poglavlju diskusija. Prvo su prikazani rezultati analize utjecaja veličine foldova cross-validacije na preciznost modela, a zatim rezultati ansambl algoritama koristeći MAE, MAPE, MSE, RMSE, R2 i KGE metriku.



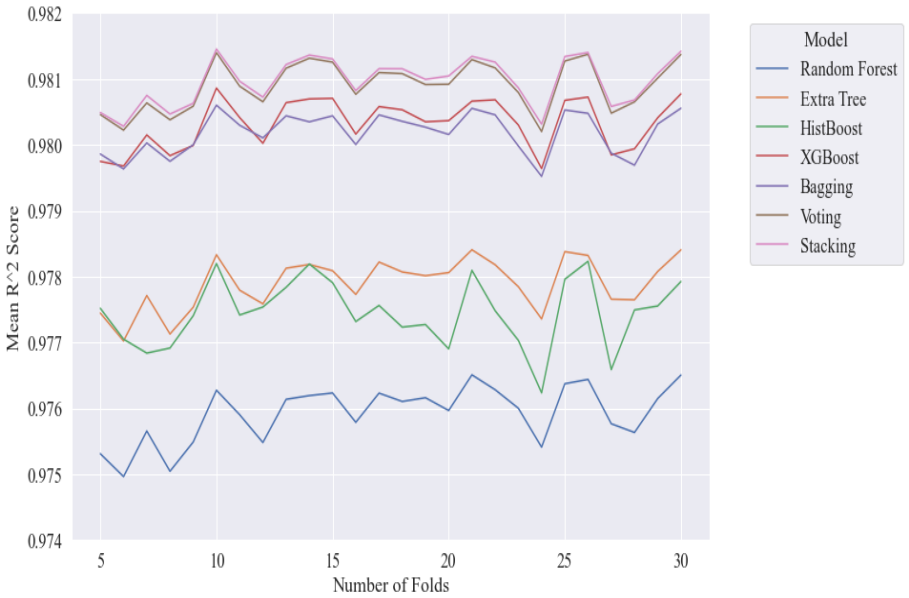
Slika 4.1. Utjecaj veličine foldova na Decision Tree Regresor za različite varijacije dataseta

Slika 4.1 prikazuje utjecaj veličine foldova na R2 *Decision Tree* regresora za transformirane datasetove. Prikazani su redom *Original*, *Standard scaler*, *Power Transformer*, *Robust scaler* i *MinMax scaler*. Izostavljen je prikaz *MaxAbs* skaliranja jer se ponaša jednako kao i *MinMax* skaliranje, također je izostavljena *Normalizer* transformacija jer pruža značajno slabije rezultate.



Slika 4.2. Utjecaj veličine foldova na R2 metriku za sve ansambl metode

Slika 4.2 prikazuje utjecaj veličine foldova na R2 metriku za sve ansambl metode. Sa slike je vidljivo da *AdaBoost* regresor ima najnižu R2 vrijednost, dok *Stacking*, *Voting*, *Bagging* i *XGBoost* regresori ostvaruju najviše vrijednosti R2.



Slika 4.3. Utjecaj veličine foldova na R2 metriku najboljih ansambl metoda

Na slici 4.3 prikazan je utjecaj foldova na R2 metriku za najbolje ansambl metode. Sa slike možemo uočiti da *Voting* i *Stacking* ansambli postižu najviše vrijednosti R2. *Xgboost* i *Bagging* ansambl metode ostvaruju nešto niže vrijednosti R2. U nastavku će biti prikazana ovisnost vrijednosti R2 metrike o broju foldova unakrsne validacije zasebno za svaku korištenu ansambl metodu.

Slika 4.4. Utjecaj veličine foldova na Decision Tree Regresor

Slika 4.4 prikazuje utjecaj veličine foldova na vrijednost R2 za *Decision Tree* regresor treniran na orginalnom datasetu. Na slici možemo uočiti da vrijednost R2 metrike oscilira oko središnje vrijednosti, najviša se vrijednost R2 postiže kada je broj foldova jednak 21, dok se najniža vrijednost postiže za 8 foldova.

Slika 4.5. Utjecaj veličine foldova na Random Forest Regresor

Na slici 4.5 prikazan je utjecaj veličine foldova na vrijednost R2 za *Random Forest* regresor. Sa slike možemo uočiti tendenciju porasta vrijednosti R2 metrike povećanjem broja foldova. Najviša se R2 vrijednost postiže kada je broj foldova unakrsne validacije jednak 21. Zanimljivo je uočiti i najveći pad vrijednosti kada je broj foldova jednak 24.

Slika 4.6. Utjecaj veličine foldova na Extra Tree Regresor

Slika 4.6 prikazuje utjecaj veličine foldova na R2 vrijednost za *Extra Tree* regresor. Sliku možemo usporediti s *Random Forest* regresorom na slici 4.5, te uvidjeti slično ponašanje ove dvije ansambl metode. Najviša vrijednost se postiže kada je broj foldova jednak 21.

Slika 4.7. Utjecaj veličine foldova na AdaBoost Regresor

Na slici 4.7 prikazan je utjecaj veličine foldova na R2 vrijednost za *Adaboost* regresor. Sa slike je vidljivo da *AdaBoost* regresor poprima najniže vrijednosti R2 (0.909) i povećanjem broja foldova unakrsne validacije dolazi do pada vrijednosti R2 metrike. Ovakvo ponašanje uočeno je samo za *AdaBoost* regresor.

Slika 4.8. Utjecaj veličine foldova na HistGradientBoost Regresor

Slika 4.8 prikazuje utjecaj veličine foldova na vrijednost R2 za *HistGradientBoost* regresor. Na slici je vidljivo da vrijednost R2 metrike značajno oscilira. Oblik funkcije ovisnosti R2 vrijednosti o broju foldova unakrsne validacije kod *Boosting* metoda poprima drugaciji oblik nego što su imale *Random Forest* i *Extra Tree* metode. Do najniže vrijednosti R2 dolazi kada je broj foldova jednak 24, dok se najviša vrijednost ostvaruje kada je broj foldova jednak 26.

Slika 4.9. Utjecaj veličine foldova na XGBoost Regresor

Slika 4.9 prikazuje utjecaj veličine foldova na vrijednost R2 za *XGBoost* regresor. Ova metoda postiže najbolje rezultate od osnovnih ansambl regresora, na slici je vidljivo da se najviša vrijednost R2 postiže kada je broj foldova jednak 10, a najniža kada je broj jednak 24.

Slika 4.10. Utjecaj veličine foldova na Bagging Regresor

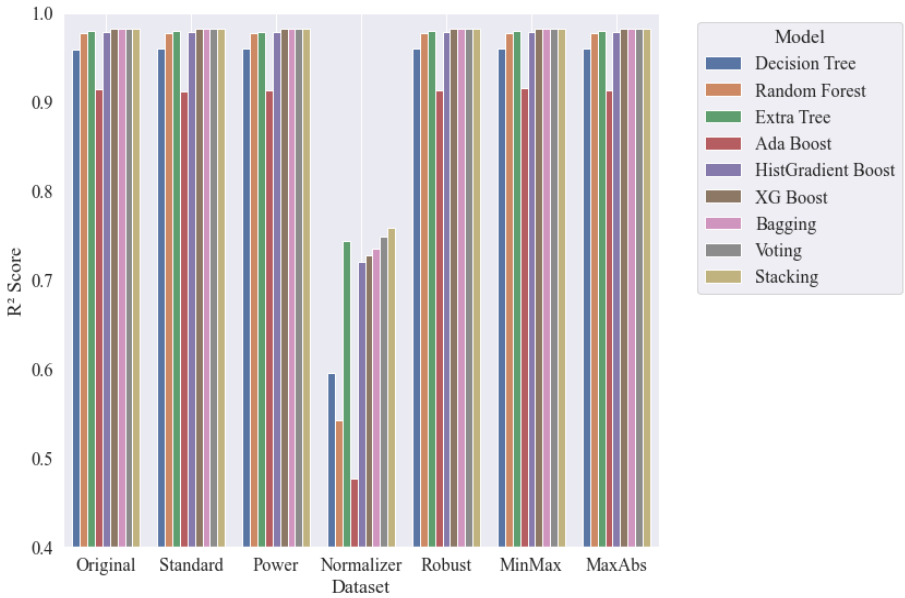
Slika 4.10 pokazuje utjecaj veličine foldova na vrijednost R2 za *Bagging* regresor. Ovo je prva od ansambl metoda koja u svom ansamblu koristi optimizirani *XGBoost* ansambl. Usporedbom sa slikom 4.9 možemo uočiti da *Bagging* metoda nije uspijela povećati vrijednosti R2 dobivene za *XGBoost* regresor.

Slika 4.11. Utjecaj veličine foldova na Voting Regresor

Slika 4.11 pokazuje utjecaj veličine foldova na vrijednost R2 za *Voting* regresor, ovaj se regresor sastoji od *ExtraTree*, *HistGradientBoost* i *XGboost* ansambl metoda. Sa slike možemo uočiti da se ostvaruju više vrijednosti R2 od individualnih metoda koje su se koristile za izradu ovog ansambla. Najviša se vrijednost ostvaruje kada je broj foldova jednak 10, a najniža kada je jednak 24.

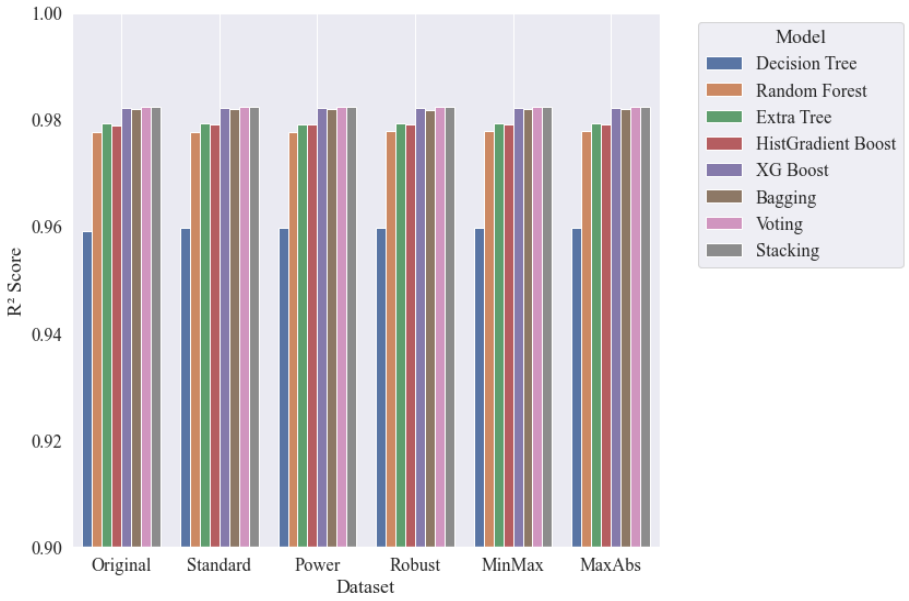
Slika 4.12. Utjecaj veličine foldova na Stacking Regresor

Na slici 4.12 prikazan je utjecaj veličine foldova na vrijednost R2 za *Stacking* regresor. Ova ansambl metoda postiže najbolje rezultate, što dokazuje da se spajanjem odabranih ansambl metoda u jedan ansmbl može postići veća točnost estimacije. Kao i *Voting* metoda sastoji se od tri optimizirane ansambl metode: *ExtraTree*, *HistGradientBoost* i *XGboost*.



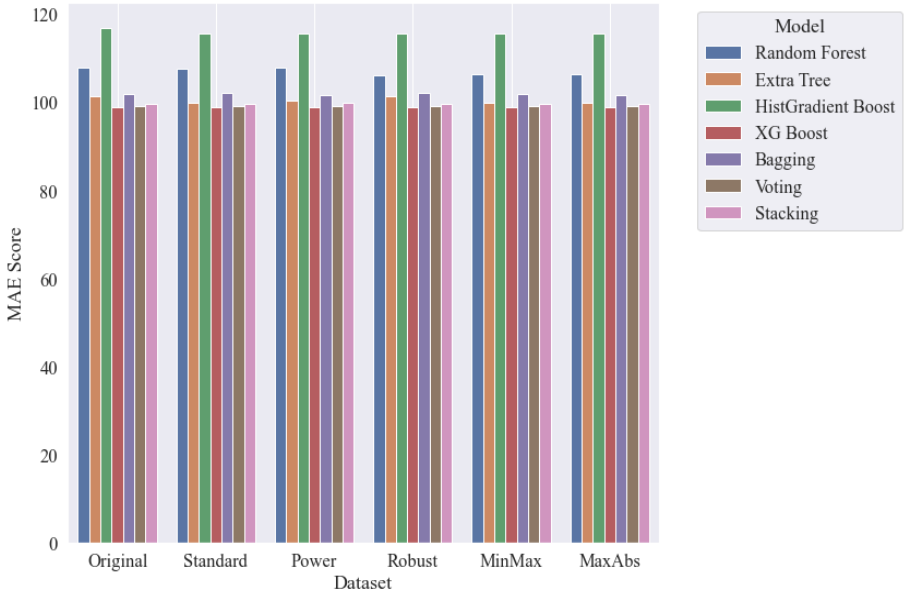
Slika 4.13. R2 metrika za sve datasetove i ansamble algoritme

Slika 4.13 prikazuje vrijednost R2 metrike za sve ansambl algoritme trenirane na svim varijacijama osnovnog dataseta. Sa slike je vidljivo da dataset skaliran *Normalizer* metodom ostvaruje znatno lošije rezultate. Također je moguće uočiti da *AdaBoost* i *Decision Tree* regresori rezultiraju niskom vrijednosti R2. U nastavku će se razmatrati samo algoritmi i varijacije dataseta koje ostvaruju zadovoljavajuće rezultate.



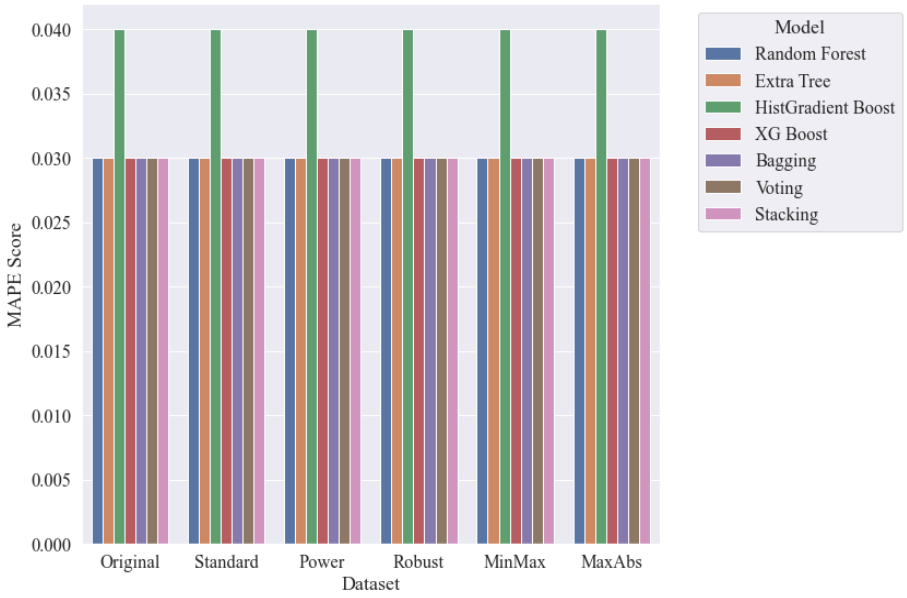
Slika 4.14. Usporedba ansambl algoritama R2 metrikom

Slika 4.14 prikazuje vrijednosti R2 metrike za odabrane ansambl algoritme trenirane na određenim varijacijama dataseta. Sa slike je vidljivo da ansambl algoritmi postižu praktički jednake vrijednosti neovisno o varijaciji dataseta na kojoj su trenirani. Najviše vrijednosti R2 postižu *Stacking* i *Voting* ansambl metode, dok *Decision Tree* metoda ostvaruje najniže vrijednosti R2.



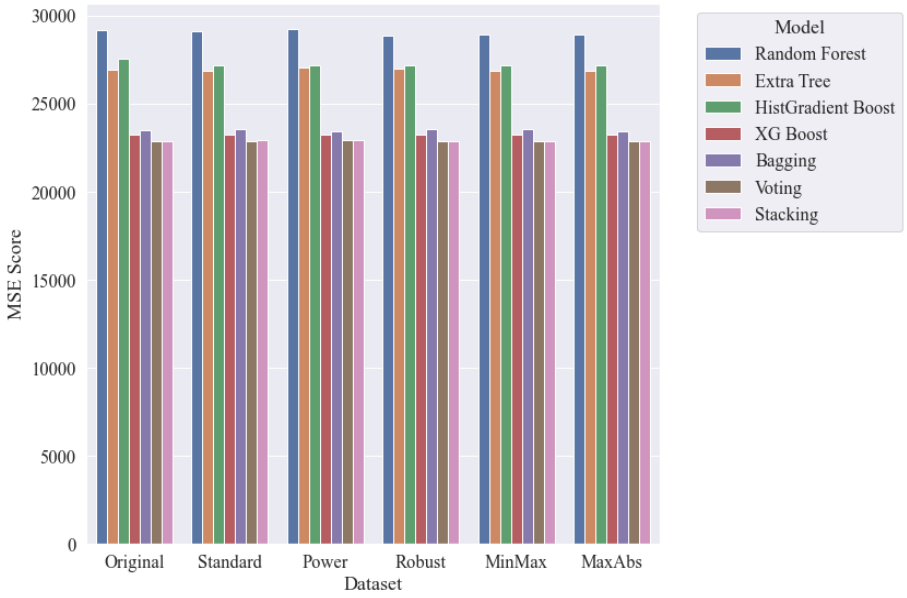
Slika 4.15. Usporedba ansambl algoritama MAE metrikom

Slika 4.15 prikazuje vrijednosti MAE metrike za odabrane ansambl algoritme trenirane na varijacijama osnovnog dataseta. Što je vrijednost MAE manja to je veća točnost estimacije algoritma. Sa slike je vidljivo da *XGBoost* poprima najniže vrijdnosti MAE, dok *HistGradientBoost* ostvaruje najviše vrijednosti.



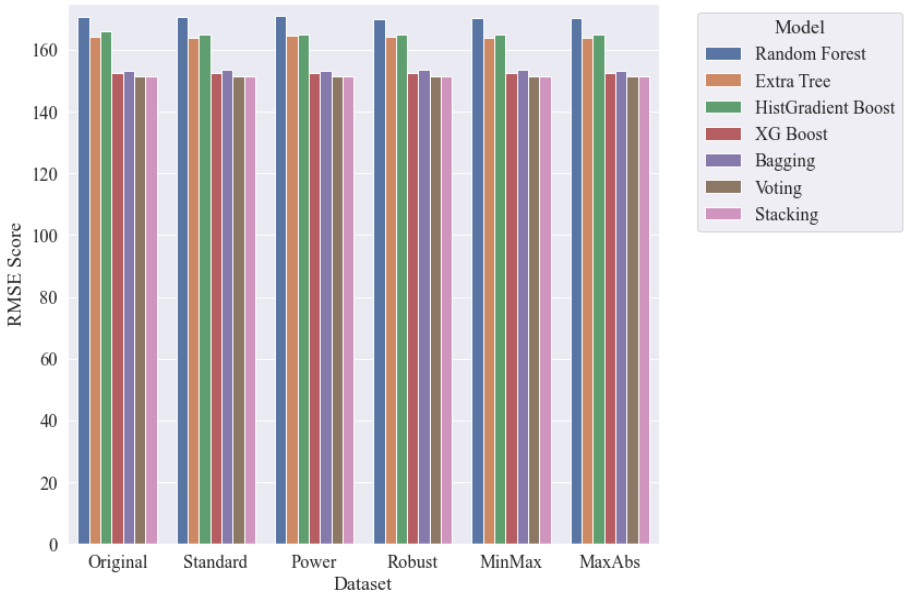
Slika 4.16. Usporedba ansambl algoritama MAPE metrikom

Slika 4.16 prikazuje vrijednosti MAPE metrike za odabrane ansambl algoritme trenirane na varijacijama osnovnog dataseta. Na slici možemo uočiti da sve ansambl metode osim *HistGradientBoosta* ostvaruju postotnu pogrešku od 3%.



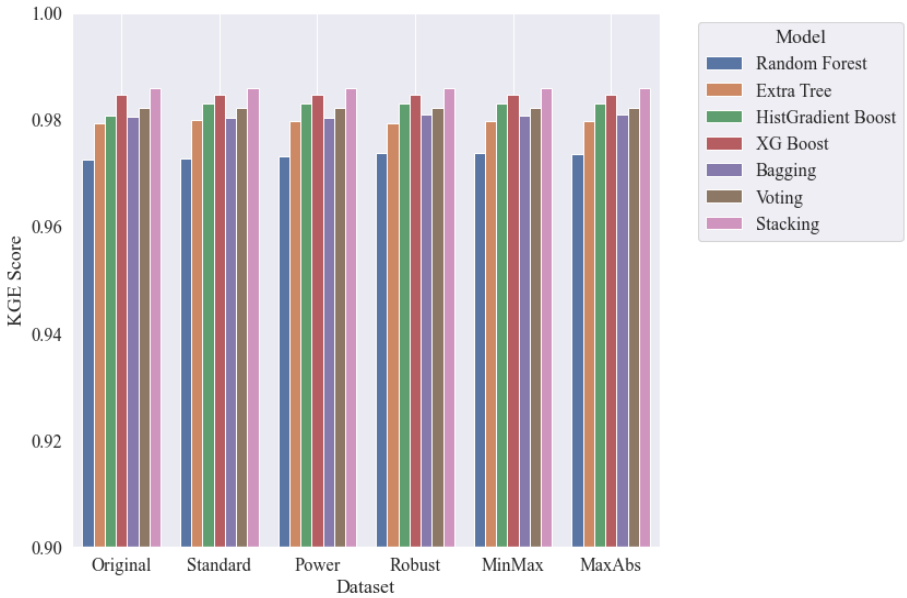
Slika 4.17. Usporedba ansambl algoritama MSE metrikom

Slika 4.17 prikazuje vrijednosti MSE metrike za odabrane ansambl algoritme trenirane na varijacijama osnovnog dataseta. Metode s manjom vrijednosti MSE ostvaruju veću preciznost. Sa slike je vidljivo da *Voting* i *Stacking* ansambl metode ostvaruju najniže vrijednosti MSE, dok *Random Forest* metoda ostvaruje najviše vrijednosti MSE.



Slika 4.18. Usporedba ansambl algoritama RMSE metrikom

Slika 4.18 prikazuje vrijednosti RMSE metrike za odabrane ansambl algoritme trenirane na varijacijama osnovnog dataseta. Vrijednost RMSE jednaka drugom korijenu vrijednosti MSE metrike. Sa slike je vidljivo da kao i na slici 4.17 najniže vrijednosti postižu *Voting* i *Stacking* ansambl metode, dok najvišu vrijednost RMSE postiže *Random Forest*.



Slika 4.19. Usporedba ansambl algoritama KGE metrikom

Slika 4.19 prikazuje vrijednosti KGE metrike za odabrane ansambl algoritme trenirane na varijacijama osnovnog dataseta. Vrijednost KGE pokazuje koliko je model stabilan, uobičajeno je da se vrijednost dobrih modela nalazi u rasponu od 0.9 do 1, dok za modele koji imaju vrijednost KGE nižu od 0.75 kažemo da su to neupotrebljivi modeli. Sa slike je vidljivo da sve odabrane ansambl metode postižu vrijednost blizu 1, što dovodi do zaključka da su ovi modeli stabilni.

5. Diskusija

U ovom će se poglavlju analizirati dobiveni rezultati i napraviti usporedba između pojedinih ansambl algoritama koristeći evaluacijske metrike. Promatrat će se utjecaj veličine foldova i varijacija dataseta na točnost ansambl metoda. Ovi su rezultati postignuti koristeći dosad objašnjenu metodologiju.

Za početak razmatramo utjecaj veličine foldova na točnost estimacije ansambl regresora. Na prikazanim grafovima možemo uočiti da R2 metrika ovisi o broju foldova unakrsne validacije. Na x-osi prikazan je raspon od minimalno 5 do maksimalno 30 foldova. Možemo uočiti da ne postoji konstantan trend rasta ili pada vrijednosti R2 metrike, već veličine osciliraju oko neke središnje vrijednosti. Očekivano ponašanje prilikom povećanja broja foldova unakrsne validacije je asimptotski porast do određene maksimalne vrijednosti. U ovom radu analiza veličine foldova pokazuje da veći broj foldova ne rezultira automatskim povećanjem točnosti modela.

Prilikom usporedbe utjecaja veličine foldova *Decision Tree* regresora za različite varijacije dataseta (slika 4.1) možemo vidjeti da najvišu vrijednost R2 metrike postiže orginalni dataset. Temeljem ovog prikaza mogli bi doći do zaključka da nema potrebe za skaliranjem orginalnog dataseta, no transformacija dataseta omogućuje brže i učinkovitije treniranje pojedinih ansambl metoda što dolazi uz cijenu blagog smanjenja točnost estimacije modela. Niža vrijednost R2 metrike kod skaliranih dataseta rezultat je skaliranja ulaznih vrijednosti pri čemu dolazi do određenog gubitka informacije iz dataseta.

Ostatak grafičkih prikaza utjecaja veličine foldova na R2 evaluacijsku metriku ansambl metoda napravljeni su za orginalni dataset. Na slici 4.2 istovremeno je prikazana ovisnost R2 metrike svih razmatranih ansambl algoritama o broju foldova unakrsne validacije. Iz ovog se prikaza može jednostavno uočiti da *AdaBoost* i *Decision Tree* regresori ostvaraju znatno niže vrijednosti R2 metrike. Na slici 4.3 prikazani su oni ansambl regresori koji ostvaruju najbolje rezultate.

Možemo uočiti da *Stacking* i *Voting* ansambl regresori ostvaruju najbolje rezultate. Ova činjenica daje odgovor na početnu tezu u kojoj se postavlja pitanje: je li moguće odabrane ansambl metode spojiti u jedan ansambl s ciljem povećanja točnosti estimacije. Ova dva algoritma pri konačnoj estimaciji koriste predikcije osnovnih regresora. Za osnovne regresore odabrani su oni koji ostvaruju najbolje rezultate. *Voting* metoda u svojem ansamblu sadrži *ExtraTree*, *HistGradientBoost* i *XGBoost* regresore. Kod *Voting* ansambla ovim se osnovnim regresorima pridružuje određena težina. Konačna estimacija *Voting* ansambla odgovara srednjoj vrijednosti težinskih predikcija regresora u ansamblu.

Za *Stacking* ansambl odabrani su isti osnovni regresori, no u ovom slučaju estimacije pojedinih regresora unutar ansambla služe kao ulazne vrijednosti u jedan meta regresor koji zatim vrši treniranje i daje konačnu estimaciju. Za odabir meta regresora korištena je *GridSearch* metoda.

Kandidati za meta regresor bili su *RidgeCV* i svi osnovni ansambl regresori. Najbolji rezultat dobiven je za *RidgeCV* regresor koji ima ugrađenu unakrsnu validaciju i traži optimalni parametar regulacije (α) za smanjenje varijance.

Na ostalim grafičkim prikazima utjecaja veličine foldova na preciznost modela zasebno je prikazan svaki ansambl algoritam. Većina algoritama pokazuje sličnu ovisnost, zanimljivo je uočiti da algoritmi postižu najvišu vrijednost R2 kada je broj foldova jednak 10, 21 ili 26, dok se najveći pad vrijednosti R2 za sve algoritme događa kada je broj foldova jednak 24. Isto tako je moguće kod većine algoritama uočiti blagi rast maksimalne vrijednosti R2 s povećanjem broja foldova, dok kod *AdaBoost* regresora vrijedi da povećanje broja foldova uzorkuje pad vrijednosti R2 metrike.

Ovi su grafički prikazi pokazali važnost dobro odabranog broja foldova koji se koriste u unakrsnoj validaciji. Možemo zaključiti da optimizacija hiperparametara i pravilan odabir broja foldova u unakrsnoj validaciji mogu značajno doprinijeti povećanju točnosti estimacije.

U nastavku se analiziraju dobiveni rezultati svih evaluacijskih metrika. Kao i kod analize utjecaja veličine foldova na slici 4.13 možemo vidjeti da su rezultati dobiveni na datasetu transformiranom pomoću *Normalizer* metode skaliranja znatno lošiji, stoga će se zbog preglednosti ti rezultati eliminirati. Iz istog razloga također eliminiramo rezultate *Decision Tree* i *AdaBoost* regresora.

Možemo vidjeti da rezultati ansambl modela kroz varijacije dataseta poprimaju gotovo jednake iznose. Do toga dolazi jer je napravljena specifična optimizacija hiperparametara za svaki regresor na svakoj varijaciji dataseta, što dovodi to toga da se neovisno o datasetu postiže određena maksimalna vrijednost metrike za svaki ansambl regresor koju nije moguće prijeći. Iako se varijacijama dataseta ne postiže veća točnost estimacije, metode skaliranja omogućuje brže i učinkovitije treniranje ansambl modela.

Najbolji modeli *Stacking* i *Voting* su ostvarili vrijednost R2 jednaku 0.9825, dok su *Bagging* i *XGBoost* regresori ostvarili vrijednost od 0.9821 i 0.9822. Možemo uočiti da od osnovnih ansambl algoritama *XGBoost* poprima najviše vrijednosti R2, dok *Bagging*, *Stacking* i *Voting* koji unutar svog ansambla koriste optimizirane *XGBoost*, *ExtraTree* i *HistGradientBoost* regresore poprimaju višu vrijednost R2 metrike. Ova činjenica pokazuje da se spajanjem više osnovnih ansambl metoda u jedan ansambl može ostvariti veća točnost estimacije.

Promatrajući rezultate za MAE i MAPE metriku dolazimo do sličnih zaključaka. *XGBoost*, *Voting* i *Stacking* ansambl modeli poprimaju najniže vrijednosti. Za ove dvije metrike *XGBoost* regresor ostvaruje najbolje rezultate: MAE $= 99.08\ kcal/mol$ i MAPE $= 3\%$. Dobivenu vrijednost MAE možemo usporediti s radom [5] u kojem *k-nearest neighbors* algoritam ima vrijednost MAE jednaku $71.54\ kcal/mol$, dok najučinkovitiji algoritam *Kernel Ridge* regresor ostvaruje vrijednost MAE jednaku $3.07\ kcal/mol$.

Važno je još uočiti da je *HistGradientBoost* regresor imao veću vrijednost R2 od *Random Forest* regresora, dok u slučaju MAE i MAPE metrika *Random Forest* regresor ostvaruje bolje

rezultate. Ovo je dobar podsjetnik važnosti korištenja više različitih evaluacijskih metrika prilikom procjene kvalitete modela.

Najbolju vrijednost MSE i RMSE metrike ostvaruje *Voting* ansambl, gdje MSE iznosi 22889.12 $kcal/mol$, a RMSE 151.29 $kcal/mol$. Dobivenu vrijednost RMSE možemo usporediti s rezultatima prijavljenim u istraživačkim radovima [4, 5]. U radu [4] se za estimaciju koriste metode *boosted trees* i *single layer neural network*, dobiveni RMSE rezultati iznose 41.81 $kcal/mol$ za *boosted trees* i 60.06 $kcal/mol$ za jednoslojnu neuronsku mrežu. U radu [5] algoritam *k-nearest neighbors* ostvaruje RMSE vrijednost od 95.97 $kcal/mol$, dok najbolji algoritam *Kernel Ridge* regresor ostvaruje vrijednost RMSE od 4.84 $kcal/mol$. Važno je napomenuti da su u radovima [4, 5] korišteni različiti datasetovi. U radu [5] korišten je dataset čija izlazna varijabla ima značajno manju varijancu od dataseta korištenog u radu [4].

U ovom se radu napravila dodatna modifikacija dataseta iz literature [4] na način da se umjesto Coulombovih matrica koriste osnovni podaci o molekuli. Uspoređujući rezultate i uzimajući u obzir da za stvarnu primjenu prihvatljiva pogreška estimacije osnovne energetske razine mora biti manja od 1 $kcal/mol$, dolazimo do zaključka da osnovnu energetsku razinu molekule na temelju osnovnih podataka o molekuli nije moguće precizno estimirati koristeći ansambl metode.

Na kraju možemo pogledati KGE metriku koja pokazuje stabilnost korištenih modela uzimajuću u obzir koeficijent korelacije, pristranost i varijancu. Vrijednosti KGE u rasponu 0.9 do 1 označavaju odličnu stabilnost modela, dok vrijendosti KGE metrike manje od 0.75 označavaju lošu stabilnost modela. Na grafičkom prikazu KGE metrike ansambl algoritama možemo ustanoviti da svi prikazani algoritmi imaju KGE vrijednost blizu 1 što znači da su svi korišteni ansambl modeli stabilni.

6. Zaključak

Kroz ovaj smo se rad upoznali s problematikom međudjelovanja više tijela pri izračunu osnovne energetske razine molekule. Kao pristup rješavanja ove problematike predloženi su algoritmi umjetne inteligencije, odnosno ansambl metode. Inicijalno su iz PubChem kolekcije izvučeni osnovni podaci o molekuli i stvoren je dataset koji će služiti za treniranje ansambl metoda. Ulazne varijable su detaljno opisane i zatim je izvršena inicijalna statistička analiza dataseta. Opisuje se proces detekcije prisutnosti *outlier* podataka i prikazuje se Pearsonova korelacijska analiza. Za kreiranje varijacija dataseta korištene su različite metode inicijalnog skaliranja. Za treniranje ansambl metoda koristila se unakrsna validacija i različite metode optimizacije hiperparametara. Svaka korištena ansambl metoda detaljno je analizirana i svaka je od njih zasebno trenirana na svim varijacijama dataseta. Dan je pregled evaluacijskih metrika koje se koriste za usporedbu ansambl algoritama i određivanje točnosti modela.

Dobiveni rezultati pružaju odgovore na pitanja koja su postavljena na početku rada. Zaključujemo da korištenjem osnovnih podataka o molekuli nije moguće precizno odrediti osnovnu energetsku razinu molekula. Korištenjem ansambl metoda ostvareni su dobri rezultati, no za realnu primjenu pogreška estimacije u slučaju predikcije osnovne energetske razine molekula mora biti manja od dobivene. U diskusiji smo došli do zaključka da je moguće odabrane ansambl metode spojiti u jedan ansambl s ciljem povećanja točnosti. Iz dobivenih rezultata vidjeli smo da utjecaj varijacije dataseta na točnost korištenih algoritama nije velik. Skaliranjem ulaznih podataka ostvaruju se nešto slabiji rezultati, ali je vrijeme treniranja i testiranja određenih algoritama kraće. Optimizacijom hiperparametara ostvarena je veća točnost estimacije modela. Provjerom utjecaja veličine foldova kod unakrsne validacije došli zmo do zaključka da se dobrim odabirom broja foldova može povećati točnost estimacije.

Pokazalo se da su *XGBoost*, *Voting* i *Stacking* najbolji ansambl modeli za estimaciju osnovne energetske razine molekula. Prednost korištenja *XGBoost* regresora naspram ostalih ansambl metoda je to što postiže dobre rezultate i što je vrijeme treniranja ovog algoritma znatno kraće od ostalih korištenih ansambl metoda. *XGBoost* algoritam se sastoji od velikog broja hiperparametara što ovoj metodi omogućuje postizanje dobrih rezultata za kompleksne datasetove. Nedostatak ovog regresora je proces pronalaska optimalnih hiperparametara koji je u slučaju *XGBoost* regresora vrlo složen, te vremenski i računski zahtjevan. *Voting* i *Stacking* ansambl metode postižu veću točnost estimacije, ali je vrijeme treniranja ovih algoritama znatno dulje od *XGBoost* regresora. Jedna od ideja za nastavak ovog rada i poboljšanje točnosti estimacije modela je korištenje umjetne neuronske mreže, gdje bi se skriveni sloj izgradio koristeći ansambl regresore. Umjetne neuronske mreže imaju specifično svojstvo da mogu s visokom točnosti estimirati izlaznu vrijednost podataka niske korelacije, što bi u slučaju estimacije osnovne energetske razine molekule iz osnovnih podataka o molekuli moglo rezultirati visokom točnosti estimacije.

Popis Literature

[1] Koch, W.; Holthausen, M.C.: "A chemist's guide to density functional theory". John Wiley & Sons, 2015.

[2] Dandu, N. i dr.: "Quantum-chemically informed machine learning: prediction of energies of organic molecules with 10 to 14 non-hydrogen atoms." The Journal of Physical Chemistry A 124.28 (2020): 5804-5811.

[3] Li, S.; Wei L.; Tao F.: "An efficient fragment-based approach for predicting the ground-state energies and structures of large molecules." Journal of the American Chemical Society 127.19 (2005): 7215-7226.

[4] Himmetoglu, B.: "Tree based machine learning framework for predicting ground state energies of molecules." The Journal of chemical physics 145.13 (2016).

[5] Hansen, K. i dr.: "Assessment and validation of machine learning methods for predicting molecular atomization energies." Journal of chemical theory and computation 9.8 (2013): 3404-3419

[6] "Bohr Radius", s Interneta, https://www.techtarget.com/, 05.05.2024.

[7] Prasanna, S.; Robert J. D.: "Topological polar surface area: a useful descriptor in 2D-QSAR." Current medicinal chemistry 16.1 (2009): 21-41.

[8] Bonchev, D.: "The problems of computing molecular complexity." Computational Chemical Graph Theory. 1990. 33-63.

[9] Friedman, J. H.: "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

[10] Breiman, L.: "Classification and regression trees", Routledge, 2017.

[11] Friedman, J H.; Bogdan E. P.: "Predictive learning via rule ensembles." (2008): 916-954.

[12] Montavon, G. i dr.: "Machine learning of molecular electronic properties in chemical compound space." New Journal of Physics 15.9 (2013): 095003.

Sažetak i ključne riječi

U ovom se radu razmatra problem estimacije osnovne energetske razine molekula koristeći ansambl metode. Na početku je rada dan kratak uvod u problematiku osnovne energetske razine molekula. Slijedi analiza literature i opis korištenog dataseta. U ovom se radu izvodi dodatna modifikacija dosad razmatranih dataseta na način da se za ulazne varijable uzimaju osnovni podaci o molekuli. U drugom dijelu rada analiziraju se ansambl algoritmi i različite metode skaliranja dataseta. Objašnjen je način rada svakog ansambl algoritma i opisani su principi svake metode skaliranja. U ovom radu razmatrani su sljedeći ansambl algoritmi: *Decision Tree*, *Random Forest*, *Extra Tree*, *AdaBoost*, *HistGradientBoost*, *XGBoost*, *Bagging*, *Voting* i *Stacking* regresori. Grafički su prikazani utjecaji pojedinih metoda skaliranja na ulazne podatke. Korištene su sljedeće metode skaliranja: *Standard Scaler*, *Power Transformer*, *Normalizer*, *Robust Scaler*, *MinMax Scaler* i *MaxAbs Scaler*. Napravljena je detaljna analiza korištenih evaluacijskih metrika koje služe kao pokazatelji točnosti estimacije, te se koriste za usporedbu učinkovitosti različitih modela. Nakon ove analize uslijedio je opis procedure treniranja i optimizacije hiperparametara pojedinih ansambl algoritama. Za treniranje modela koristila se unakrsna validacija, dok su se za optimizaciju hiperparametara koristile metode *GridSearchCV* i *RandomizedSearchCV*. U drugom dijelu rada prezentirani su dobiveni rezultati. Promatrao se utjecaj veličine foldova unakrsne validacije na točnost estimacije ansambl algoritama i međusobno su uspoređeni rezultati pojedinih ansambl metoda. Za usporedbu ansambl metoda korištene su sljedeće metrike: R2, MAE, MAPE, MSE, RMS i KGE. Najbolje rezultate estimacije osnovne energetske razine molekula ostvarili su *XGBoost*, *Voting* i *Stacking* regresori. Na kraju rada objašnjene su neke prednosti i nedostatci korištenih algoritama, te su se doneseni sljedeći zaključci: kvalitetnim odabirom broja foldova koji se koriste pri unakrsnoj validaciji moguće je ostvariti veću točnost estimacije i spajanje odabranih ansambl metoda u jedan ansambl ostvaruje povećanje točnosti estimacije. U zaključku se predlaže jedan od mogućih načina unaprijeđenja modela, a to je korištenje umjetne neuronske mreže u kojoj je skriveni sloj izgrađen od optimiziranih ansambl metoda.

Ključne riječi: osnovna energetska razina molekula, ansambl metode, metode skaliranja, unakrsna validacija, optimizacija hiperparametara, evaluacijske metrike, algoritmi umjetne inteligencije

Summary and key words

The main focus of this paper is using ensemble methods to estimate the ground state energies of molecules. In the first part we analyze different literatures that focus on issues that are encountered when predicting ground state energies of molecules. Dataset which is used in this paper is created by using basic information about molecules from PubChem database. First we create a few modified versions of dataset using various scaling methods such as: *Standard Scaler*, *Power Transformer*, *Normalizer*, *Robust Scaler*, *MinMax Scaler* and *MaxAbs Scaler*. These modified datasets will be used to train ensemble algorithms and we want to observe the impact that different scaling methods have on original dataset and how that reflects onto the precision of ground state energy estimations. The ensemble methods used in this paper are: *Decision Tree*, *Random Forest*, *Extra Tree*, *AdaBoost*, *HistGradientBoost*, *XGBoost*, *Bagging*, *Voting* and *Stacking* regressors. The training for each of these methods is performed on every modified dataset. The training is done by using cross-validation and finding optimal hyperparameters. To find optimal hyperparameters two functions are used: *GridSearchCV* and *RandomizedSearchCV*. The main objectives are to determine the impact that number of folds in cross-validation have on the model accuracy and to determine if we can improve the model accuracy by combining multiple basic ensembles into one ensemble. To compare the ensemble methods we use the following evaluation metrics: R2, MAE, MAPE, MSE, RMS and KGE. The second part of the paper focuses on analyzing the acquired results. By comparing evaluation metrics for each of the ensemble methods used we come to a conclusion that *XGBoost*, *Voting* and *Stacking* regressors perform the best and that by combining multiple ensemble methods into one ensemble we can increase the model performance. When analyzing the impact that number of folds have on the accuracy of the model we came to a conclusion that the higher number of folds doesn't automatically result in higher model accuracy, which means that when using cross-validation it is necessary to find the optimal number of folds to maximize model performance. Lastly we compare the best ensemble methods and explain their strengths and weaknesses and offer an idea to improve the results of ensemble methods by using them in a hidden layer of an artificial neural network (ANN).

Keywords: ground state energy, ensemble methods, scaling methods, cross-validation, hyperparameter optimization, evaluation metrics, artificial intelligence algorithms

A Programski kod za pretraživanje optimalnih hiperparametara

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV

data = pd.read_csv('NewRobo_GSE.csv')
z=data.pop('molecular_weight')
y=data.pop('GSE')*313.495392
X_train, X_test, y_train, y_test = train_test_split(data, y, test_size = 0.3,
    random_state=42)

param_grid = {
    'n_estimators': [300,400,500,600,700,800,900,1000],
    'learning_rate': [0.1,0.01,0.05],
    'max_depth': [4,5,6,7],
    'subsample': [0.8,0.9,1.0],
    'colsample_bytree': [0.8,0.9,1],
    'alpha': [0,0.1,0.5,1.0],
    'lambda': [0,0.1,0.5,1.0],
    'gamma': [0,0.1,0.5,1.0]
}

xgb_reg = xgb.XGBRegressor()

grid_search = GridSearchCV(estimator=xgb_reg, param_grid=param_grid, scoring='
    r2', cv=10, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_
best_model = grid_search.best_estimator_

y_pred_train = best_model.predict(X_train)
y_pred_test = best_model.predict(X_test)

r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)
```

B Programski kod za analizu utjecaja veličine foldova unakrsne validacije

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import VotingRegressor, StackingRegressor
from sklearn.linear_model import RidgeCV

data = pd.read_csv('NewRobo_GSE.csv')

z=data.pop('molecular_weight')
y=data.pop('GSE')*313.495392

base_model= xgb.XGBRegressor(learning_rate=0.05,n_estimators=1100,max_depth=8,
    subsample=0.7,colsample_bytree=1,gamma=1,min_child_weight=4,random_state
    =42)
best_weights=[0.21824004, 0.54531324, 0.23644672]
reg1 = ExtraTreesRegressor(n_estimators=900,max_depth=(50),max_features=1.0,
    min_samples_leaf=1,min_samples_split=2,bootstrap=False,random_state=42)
reg2 = xgb.XGBRegressor(learning_rate=0.05,n_estimators=1100,max_depth=8,
    subsample=0.7,colsample_bytree=1,gamma=1,min_child_weight=4,random_state
    =42)
reg3 = HistGradientBoostingRegressor(max_depth=(17),l2_regularization=0.05,
    learning_rate=0.1,max_iter=848,min_samples_leaf=2,random_state=(42))
best_meta_regressor=RidgeCV()
base_estimators = [('ET', reg1), ('XGB', reg2), ('HGB', reg3)]

models = {
    'Decision Tree': (data,y,DecisionTreeRegressor(max_depth=(None),
    max_features=1.0,min_samples_leaf=3,min_samples_split=13,splitter='best',
    random_state=42)),
    'Random Forest': (data,y,RandomForestRegressor(n_estimators=651,max_depth
    =(90),max_features='log2',min_samples_leaf=1,min_samples_split=3,bootstrap
    =False,random_state=42)),
    'Extra Tree': (data,y,ExtraTreesRegressor(n_estimators=900,max_depth=(50),
    max_features=1.0,min_samples_leaf=1,min_samples_split=2,bootstrap=False,
    random_state=42)),
```

```
    'AdaBoost': (data,y,AdaBoostRegressor(n_estimators=805, learning_rate=1.6,
     random_state=42)),
    'HistBoost': (data,y,HistGradientBoostingRegressor(max_depth=(17),
    l2_regularization=0.05,learning_rate=0.1,max_iter=848,min_samples_leaf=2,
    random_state=(42))),
    'XGBoost': (data,y,xgb.XGBRegressor(learning_rate=0.05,n_estimators=1100,
    max_depth=8,subsample=0.7,colsample_bytree=1,gamma=1,min_child_weight=4,
    random_state=42)),
    'Bagging': (data,y,BaggingRegressor(base_model,max_features=1.0,
    max_samples=1.0,n_estimators=20,random_state=42)),
    'Voting' : (data,y,VotingRegressor(estimators=[ ('ET', reg1), ('XGB', reg2
    ), ('HGB', reg3), ], weights=best_weights)),
    'Stacking': (data,y,StackingRegressor(estimators=base_estimators,
    final_estimator=best_meta_regressor))
}


folds = range(5, 31)

r2_scores = {name: [] for name in models}

for name, (X_data, y_data, regressor) in models.items():
    print(f"Processing model: {name}")
    for k in folds:
        print(f"Running cross-validation with {k} folds")
        r2 = cross_val_score(regressor, X_data, y_data, cv=k, scoring='r2')
        r2_scores[name].append(np.mean(r2))
        print(f"Finished cross-validation with {k} folds")
```