Estimacija osnovnog energetskog stanja molekula primjenom algoritama strojnog učenja

Babić, Filip

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

ESTIMACIJA OSNOVNOG ENERGETSKOG STANJA MOLEKULA PRIMJENOM ALGORITAMA STROJNOG UČENJA

Rijeka, ožujak 2024.

Filip Babić

0069072218

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

ESTIMACIJA OSNOVNOG ENERGETSKOG STANJA MOLEKULA PRIMJENOM ALGORITAMA STROJNOG UČENJA

Mentor Prof. dr. sc. Zlatan Car

Rijeka, ožujak 2024.

Filip Babić

0069072218

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 14. ožujka 2023.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Primjena umjetne inteligencije**
Grana: **2.03.06 automatizacija i robotika**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Filip Babić (0069072218)**
Studij: Sveučilišni diplomski studij elektrotehnike
Modul: Automatika

Zadatak: **Estimacija osnovnog energetskog stanja molekula primjenom algoritama strojnog učenja**

Opis zadatka:

Napraviti pregled postojećih istraživanja estimacije osnovnog energetskog stanja molekula primjenom različitih algoritama strojnog učenja. Na javno dostupnom skupu podataka napraviti statističku analizu i pred obradu podataka. Izvršiti treniranje različitih algoritama strojnog učenja te napraviti procjenu koji od algoritama ima najveću točnost estimacije. Ispitati da li je estimaciju moguće poboljšati primjenom metode nasumičnog pretraživanja hiperparametara algoritama strojnog učenja i treniranja korištenjem unakrsne validacije. Ispitati da li se točnost estimacije može poboljšati korištenjem ansambl metoda.

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

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Zlatan Car

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Dubravko Franković

IZJAVA

Ja, Filip Babić (JMBAG: 0069072218), student Tehničkog fakulteta Sveučilišta u Rijeci sukladno pravilima sadržanih u dokumentu „Pravilnik o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija“ izjavljujem i potpisom potvrđujem da je ovaj rad „Estimacija osnovnog energetskog stanja molekula primjenom algoritama strojnog učenja“ izrađen i napisan samostalno koristeći vlastita znanja i navedenu literaturu.

Rijeka, ožujak 2024.

Filip Babić

ZAHVALA

Htio bi se zahvaliti prof. dr. sc. Zlatanu Caru na prilici za odraditi rad pod njegovim mentorstvom i v. asist. dr. sc. Nikoli Anđeliću na savjetima i pomoći prilikom istraživanja i pisanja rada.

Htio bi se isto tako zahvaliti roditeljima koji su me podržavali i gurali svih ovih godina studiranja, koji su mi usadili strast za napredovanje i usvajanje novih znanja. Zahvalio bi se bratu na tome što je neiscrpni izvor diskusija, rasprava i znanja iz biotehnologije i kemo-informatike koja je bila katalizator iza odabira teme rada. Također htio bi se zahvaliti svojoj curi što je uvijek bila uz mene i zbog koje sve ovo na kraju ima smisla.

Posvećujem rad noni koja je bila više ponosna na naše uspjehe nego mi sami, ali na žalost nije doživjela vidjeti ovaj, Počivala u miru.

SADRŽAJ

1. UVOD

Prilikom razvoja kemije kao znanosti redovito se javlja potreba za novim metodama i tehnologijama koje bi ubrzale računanje i simulaciju kemijskih veza između atoma i molekula. Kako bi se mogli modelirati kemijski procesi na toj razini potrebno je imat opis atoma i molekula u vidu poznavanja položaj elektrona koji su sadržani u tom atomu ili molekuli. Naime zbog kvantne prirode subatomskih čestica egzaktna pozicija elektrona u sustavu se ne može odrediti već se promatra kao funkcija gustoće vjerojatnosti da se elektron nalazi u određenoj točci prostora. Vjerojatnost nalaženja elektrona u prostoru sadržana je u valnoj funkciji tog elektrona koja se dobije rješavanjem Schrödingerove jednadžbe gdje se iz tih izračuna mogu izlučiti energetska stanja. Energetsko stanje opisuje elektronsku konfiguraciju atoma te diktira kemijske karakteristike materijala što je potrebno za analizu i razumijevanje kemijskih procesa. Problem se javlja kod čestica sa više od jednog elektrona gdje rješavanje Schrödingerove jednadžbe postaje praktički neizvedivo. Ovaj problem potaknuo je istraživače da razviju druge modele računanja koja se baziraju na aproksimativnim metodama, metode sa generalizirajućim pretpostavkama, iterativnim metodama i metodama temeljene na skupovima podataka (*engl. data-driven methods*) [1-4].

Eksplicitno računanje elektronskih struktura čak i aproksimativnim metodama se pokazalo suviše kompleksno te su se napori usmjerili prema direktnom računanju energija za pojedine atomske konfiguracije. Kako bi se zaobišle kompleksne formule razvijaju se metode predviđanja empirijskim metodama za što se pronašlo da su metode strojnog učenja dale dobar kompromis između preciznosti i kompleksnosti. Korištenjem strojnog učenja pokušava se zaobići parametrizacija modela nego se prepušta algoritmu da na temelju velike količine dostupnih podataka izvrši interpolacija vrijednosti iz elektronske strukture. Istraživanja su pokazala da se korištenjem strojnog učenja može približiti preciznosti drugih egzaktnijih metoda uz nedostatak da su tako izučeni modeli ograničeni zbog svoje inherentne specijalizacije. Iako je trud usmjeren da se napravi model koji se može koristiti za što više spojeva sa velikom preciznošću, u praksi se pokazalo da u određenim strukama (npr. tehnički materijali u strojarstvu) točnost ne mora biti apsolutna, a brzina izračuna igra veliku ulogu [5-7].

Većina istraživanja generalno koristi strojno učenje za estimaciju i minimizaciju pojedinih članova unutar postojećih formula, ili određivanja pojedinih koeficijenata kako bi se unaprijedilo računanje ab initio (*lat. ...iz prvih principa*) [1-3]. U ovom radu fokus će biti na učenju regresijskih

algoritama na javno dostupnom setu podataka kako bi se ispitale mogućnosti interpolacije metoda strojnog učenja kao samostalnog alata.

U ovom radu će se obraditi sljedeće hipoteze:

- Može li se izvršiti estimacija osnovnog energetskog stanja molekule korištenjem javno dostupnog seta podataka?
- Koje metode obrade seta podataka i redukcije dimenzija daju najbolje rezultate za odabrane algoritme strojnog učenja?
- Mogu li se pojedinačni modeli poboljšati korištenjem pretrage hiperparametara i korištenjem unakrsne validacije?
- Mogu li se modeli iskoristiti u ansambl metodi u svrhu poboljšanja estimacije?

U sljedećem poglavlju odraditi će se pregled trenutne literature u svrhu upoznavanja sa temom i aktualnim metodama estimacije elektronskih veličina i primjene metoda strojnog učenja u toj grani znanosti. Prvo će se postaviti teorijska baza koja će omogućiti razumijevanje seta podataka i postupci kojima su se ti podatci generirali. Nakon toga odabrani set podataka će se vizualizirati i analizirati kako bi se dobio uvid u karakteristike podataka i odabrao način kojim će se izvršiti redukcija dimenzija i skaliranje. Sa informacijama sakupljenim iz analize podataka odabrane su metode strojnog učenja te su opisani algoritmi estimacije i evaluacijske metrike kojim se ocjenjuju estimatori. To je popraćeno opisom podjela podataka za treniranje i testiranje, odabirom metoda unakrsne validacije, metoda optimizacije hiperparametara, metoda redukcije dimenzija i metoda ansambla estimatora. Na kraju rada prikazani su rezultati nabrojanih metoda. Nakon rezultata cijeli rad je prokomentiran te je izvučen zaključak iz odrađenog pokusa kojim se dao odgovor na postavljene hipoteze.

2. MATERIJALI

Za razumijevanje motivacije iza istraživačkih pitanja te uvod u tematiku rada potrebno je isti smjestit u kontekst znanosti u kojem se nalazi. Rad ovakve prirode zahtjeva interdisciplinarnost u istraživanju pošto teorijska podloga iz koje istraživačko pitanje proizlazi je bazirana na jednoj grani znanosti, a metode kojima se ta pitanja ispituju je bazirana na drugoj. Zbog toga je rad koncipiran tako da su poglavlja podijeljena na materijale i metode. Materijali korišteni u radu se odnose na druge radove trenutnog istraživanja bliske temi i kratkom pregledu istih. Opisuju se osnovna fizikalna načela bitna za kontekstualno razumijevanje tog istraživanja i podataka koji su proizašli iz tog prethodnog istraživanja.

2.1 Pregled postojećih istraživanja

Kvantna kemija kao polje znanstvenog istraživanja nalazi se na križanju kvantne fizike i klasične kemije te je utjecala na razvoj drugih polja i unapređenja gotovo svih modernih tehnologija. Od začeća grana kvantne fizike najveći problem predstavljalo je stupanj tehnološke razvijenosti koja je bila potrebna za računanje, testiranje i simulaciju postavljenih teorija koja nije postojala u to vrijeme. Par desetljeća prije početka 21. stoljeća obilježeno je eksponencijalnim rastom računalnih znanosti i računalnih tehnologija koji je stvorio uvjete za preporod i transformaciju znanosti baziranih na kvantnim procesima [2,3]. Pojava računala koja imaju dovoljno memorije i procesne snage za računanje određenih čestičnih interakcija omogućilo je do tad neviđene simulacije procesa. U vrijeme kada su određena otkrića u znanosti materijala često pronalažena slučajno ili stručnim nagađanjem mogućnost računalnih testiranja ubrzala je i pojednostavila taj proces jer se evaluacija hipoteza mogla izvršavati *in silico*. U zadnjih desetak godina kao katalizator za novu transformaciju na poljima kvantnih znanosti bio je zaslužan razvoj metoda strojnog učenja zbog svojeg svojstva predviđanja i izlučivanja karakteristika iz naizgled složenih sustava [5,4].

Jedna od najkorištenijih metoda koja se koristi za istraživanje elektronskih struktura u fizici, kemiji i znanosti o materijalima je metoda teorije funkcionala gustoće (*engl. density functional theory - DFT*). Broj objavljenih radova koji koriste *DFT* se kreće od 10 do 30 tisuća radova godišnje u odnosu na par tisuća objavljenih radova u istom vremenskom periodu za druge metode kao što su istraga vezanih klastera (engl. *coupled-cluster-CC*) [9-11]. *DFT* kao metoda ima manju točnost izračuna od drugih, ali se izvršava značajno brže i može se skalirati na sustave sa velikim brojem čestica pa je veliki trud usmjeren u razvoj optimizacije te metode. U ovom radu odabran je fokus na DFT metodu zbog činjenice da je u širokoj primjeni pa postoji dostupnost velike baze podataka koja će biti temelj za treniranje algoritama estimatora [10-12].

2.2 Teorijska podloga – Schrödingerova jednadžba

Za uvod u *DFT* potrebno je postaviti skraćenu teorijsku podlogu kvantne fizike iz koje ta teorija proizlazi. Postavljanje formula također će služili za daljnje razumijevanje seta podataka i pridodati će kontekst postavljenim hipotezama, a ujedno i pokazat zbog čega je bitno poznavat osnovno energetsko stanje.

Kemijske i fizikalne karakteristike elemenata sadržane su u energiji subatomskih čestica tog elementa u vidu među-interakcije istih. Ako se uzme analogija klasične mehanike te energije mogu se postaviti kao kinetička energija izazvana gibanjem čestica (elektrona i jezgre) i potencijalne energije izazvana pozicijom elektrona u odnosu na jezgru uslijed elektrostatskog privlačenja i odbijanja nabijenih čestica. Ako se za takvu pretpostavku postavi Hamilton funkcija ona izgleda kao suma tih energija za svaku česticu u prostoru za jedan atom[13, 1].

$$H = \sum(j)\frac{p_j^2}{2m_j} + V(q_{1,\dots}q_n), \tag{2.1.1}$$

Gdje je: *j* - broj čestice, *p* - moment čestice, *m* – masa čestice, *V* – potencijalna energija čestice u prostornoj koordinati *q* za *j*-tu česticu. Situacija se mijenja kada se uzme u obzir relativističko ponašanje elektrona i njegove suspregnute prostorno-vremenske valne funkcije Ψ koja predstavlja vjerojatnost nalažena elektrona u prostoru. Također uzima se u obzir da pozicije elektrona oko atoma se mogu nalaziti u određenim diskretnim razinama (ljuske) koje ovise o energetskom stanju elektrona. Spajanjem ova dva postulata zaključuje se da elektroni imaju različite valne funkcije ovisno o ljusci u kojoj se nalaze zbog svoje pozicije, te da je energija elektrona *E* vezana za poziciju elektrona. Kada se te pretpostavke uvrste u formulu (2.1.1) dobije se Schrödingerova jednadžba koja glasi [1]

$$H\Psi = E\Psi. \tag{2.1.2}$$

Ako se Schrödingerova jednadžba raspiše i proširi za slučaj više atoma (molekulu) onda imamo

$$H = \left[-\sum_i \frac{\hbar}{2m_e}\nabla_i^2 - \sum_k \frac{\hbar}{2m_k}\nabla_k^2 - \sum_i\sum_k \frac{e^2 Z_k}{r_{ik}} + \sum_{i<j}\frac{e^2}{r_{ij}} + \sum_{k<l}\frac{e^2 Z_k Z_l}{r_{kl}}\right]\Psi, \tag{2.1.3}$$

gdje *i* i *j* iteriraju elektrone, *k* i *l* iteriraju jezgre, *ħ* predstavlja Planckovu konstantu, m_e i m_k su masa elektrona i masa jezgre *k*, ∇^2 je Laplacijski operator, *e* je konstanta naboja, *Z* je atomski broj, r_{ab} su udaljenosti između čestica. Redom s lijeve strane članovi predstavljaju kinetičku energiju elektrona, kinetičku energiju jezgre, Coulombovo elektrostatičko privlačenje jezgre i elektrona, Coulombovo obijanje elektrona i Coulombovo odbijanje jezgri. Gledajući formulu (2.1.3) postoji više jedinstvenih valnih funkcija (moguće i beskonačno puno) koje opisuju molekulu za *H* gdje je svaka od njih karakterizirana drugačijom vrijednošću energije *E* [13][1].

Valne funkcije kao takve su generalno opisane u tri koordinate te ako se ortogonalno normira jednadžba tako da za $i \neq j$ rješenje je jednako 0 a za $i = j$ ima rješenje onda se jednadžba (2.1.2) raspisuje kao

$$\int \Psi_i \Psi_j H dr = E. \tag{2.1.4}$$

Iz ovoga se zaključuje da ako imamo skup ortogonalnih valnih funkcija mogu se dobiti sve mjerljive vrijednosti sustava uključujući i energiju. Ako se pretpostavi da odabirom proizvoljne funkcije Φ koja se može uvrstiti u H tako da postoji neko rješenje (poznate su pozicije jezgri i elektrona, broj elektrona i atomski brojevi) onda ta funkcija mora biti linearna kombinacija normirane valne funkcije kao

$$\Phi = \sum_i c_i \Psi_i \, , \tag{2.1.5}$$

gdje su c_i koeficijenti linearne kombinacije. Ubacivanjem (2.1.5) u (2.1.4) uz član H i E koeficijenti c_i (koji su nam nepoznati) se pokrate, te uzimanjem u obzir definicije koja kaže da najniža razina energije E_0 mora biti veća ili jednaka 0 izvede se formula koja izgleda kao

$$\frac{\int \Phi H \Phi dr}{\int \Phi^2} \geq E_0 \, , \tag{2.1.6}$$

koja implicira da se nasumičnim odabirom funkcije Φ koja minimizira vrijednost E_0 može eventualno dobiti točna valna funkcija te su ove funkcije dokazi varijacijskog teorema [13,1].

2.2 Teorijska podloga – DFT

Prethodni izvodi nagovještaju mogućnost dobivanja elektronskih karakteristika, doduše to se pokazalo samo kao teoretska mogućnost. U praktičnom smislu rješavanje Schrödingerove jednadžbe u svrhu dobivanja valne funkcije odrađeno je uspješno za atom vodika i helija kao najmanje složeni kemijski elementi. Problem se javlja što se valna funkcija mora izračunati u tri prostorne koordinate za svaki elektron gdje su se prethodno još i zanemarivale među-interakcije elektrona za atome s više čestica. *DFT* metoda je razvijena tako da se pokuša dobiti rješenje sustava upotrebom neke druge funkcije koja bi poželjno bila u praksi mjerljiva veličina [13-15].

Činjenica da prethodno definirana funkcija (2.1.4) sadržava sve potrebne informacije za rješavanje sustava postavlja se pitanje može li se supstituirati neka druga funkcija koja dozvoljava konstrukciju Hamiltonijana? Ako se promotre članovi vidi se da je potrebno da funkcija ovisi samo o pozicijama i atomskim brojevima jezgri i ukupnog broja elektrona. Korisna mjerljiva veličina bila bi u tom slučaju prostorna elektronska gustoća. Integriranjem gustoće ρ u prostoru dobije se skalar koji predstavlja ukupan broj elektrona N te je prikazana u sljedećem raspisu

$$N = \int \rho(r) dr. \tag{2.2.1}$$

Uzevši u obzir da se jezgre atoma mogu smatrati točkastim nabojem proizlazi da će pozicije tih jezgri biti prikazane kao lokalni maksimum elektronske gustoće u prostoru, a također iz toga može se odrediti i atomski broj koji je proporcionalan naboju (amplitudi maksimuma) [13-15].

Ostalo je još odrediti kako riješiti sustav i dobiti osnovnu energetsku razinu sustava, a to znači primjenjivanje varijacijskog teorema, ali za slučaj elektronske gustoće. Za to uvodi se prvi Hohenberg-Kohn teorem odnosno teorem o egzistenciji koji dokazuje da mora postojati jedinstvena osnovna elektronska gustoća koja opisuje osnovnu energetsku razinu. Ako za isti atom se postave dva različita Hamiltonijana moraju se razlikovati u članu potencijalne energije. Ako imamo dva različita Hamiltoniana H_a i H_b razlika između njih biti će razlika potencijala v_a i v_b prikazana kao

$$\widehat{H}_a - \widehat{H}_b = v_a - v_b \ , \tag{2.2.2}$$

i ako znamo da elektronska gustoća sadrži svu potrebne informacije sustava onda možemo napisati

$$\Psi^*_{0,a}(r)\Psi_{0,a}(r)dr = \Psi^*_{0,b}(r)\Psi_{0,b}(r)dr = \rho_0(r) \ , \tag{2.2.3}$$

gdje su prikazane prostorne vjerojatnosti valne funkcije osnovne razine jednake elektronskoj gustoći osnovne razine. Postavljanjem varijacijskog teorema uvrštavanjem Hamiltonijana za slučaj a) dobije se raspis oblika

$$E_{o,a} < \int \Psi^*_{0,a}\left(\widehat{H}_a - \widehat{H}_b\right)\Psi_{0,a}dr + \int \Psi^*_{0,b}\widehat{H}_b\Psi_{0,b}dr \ , \tag{2.2.4}$$

u kojem onda se u lijevi član integrala uvrštava formula (2.2.2) i (2.2.3), a desni integral je isti kao (2.1.4) koji onda postaje $E_{0,b}$ te se može raspisati kao

$$E_{0,a} < \int (v_a - v_b)\rho_0(r)dr + E_{0,b} \ . \tag{2.2.5}$$

Napravi li se isti raspis za Hamiltonijan u slučaju b), odnosno za $E_{0,b}$ onda te dvije jednadžbe oblika (2.2.5) mogu se zbrojiti i dobije se nejednakost

$$E_{0,a} + E_{0,b} < E_{0,b} + E_{0,a} \ . \tag{2.2.6}$$

Ova nejednakost očito ne važi što dokazuje da dvije različite valne funkcije sa različitim energijama ne mogu postojati za istu funkciju elektronske gustoće. Odnosno osnovna energetska razina je određena samo za jedinstvenu osnovnu razinu funkcije elektronske gustoće. Činjenica da je pokazana direktna povezanost energije sa elektronskom gustoćom znači da se varijacijski teorem (2.1.6) može primijeniti na isti način [13-15].

2.3 Teorijska podloga – Kohn-Sham jednadžba

Trenutni raspisi dokazali su da elektronska gustoća određuje vanjski potencijal, koja određuje Hamiltonijan koji određuje valnu funkciju. Sa valnom funkcijom i Hamiltonijanom može se varijacijom doći do energetske razine. To znači da se izračun energije u zadnjem koraku svejedno svede na rješavanje Schrodingerove jednadžbe kod varijacije pa taj dio predstavlja problem u rješavanju zbog već spomenutih razloga. Član koji opisuje interakciju između elektrona je problematičan za strogo odrediti. Kohn i Sham postulirali su da ako se postavi sustav gdje su elektroni neinteraktivni jedni od drugih takav bi Hamiltonijan bio predstavljen sumom jednoelektronskih Hamiltonijana. Takav Hamiltonijan naziva se Slaterovom determinantom koji predstavlja eigenfunkciju koji je suma eigenvrijednosti elektrona. Potreba za taj postulat je pretpostaviti da se krene od zamišljenog sustava neinteraktivnih elektrona da ima istu osnovnu elektronsku gustoću kao isti takav sustav sa interaktivnim elektronima pošto se pozicije i atomski brojevi jezgri jednaki, a oni su funkcional gustoće. Naime razlika će biti u članu kinetičke energije i članu neklasične elektron-elektron interakcije. Ta razlika može se pridodat formuli funkcionala gustoće kao korekcijski članovi te se postavlja Kohn-Sham jednadžba energije kao funkcionala gustoće formulom sljedećeg raspisa [13-15]

$$E[\rho_0(r)] = T_{ni}[\rho_0(r)] + V_{ne}[\rho_0(r)] + V_{ee}[\rho_0(r)] + \Delta T[\rho_0(r)] + \Delta T_{ee}[\rho_0(r)]\,, \quad (2.3.1)$$

gdje su T_{ni} član kinetičke energije, V_{ne} nuklearno-elektronska interakcija, V_{ee} klasično elektronsko odbijanje ΔT je korekcija kinetičke energije zbog elektronske interakcije, ΔV_{ee} korekcija za sve ostale neklasične elektron-elektron interakcije [13-15]. Ako se jednadžba (2.3.1) raspiše u formatu Hamiltonijan operatora, odnosno supstituiranim Slaterovim determinantama χ_i izgledat će ovako

$$E[\rho_0(r)] = \sum_i^N \left(\langle \chi_i \left| \tfrac{1}{2}\nabla_i^2 \right| \chi_i \rangle - \langle \chi_i \left| \sum_k^{jezg} \frac{Z_k}{|r_i - r_k|} \right| \chi_i \rangle \right) \quad (2.3.2)$$

$$+ \sum_i^N \langle \chi_i | \tfrac{1}{2} \int \frac{\rho_0(r')}{|r_i - r'|} dr' \, | \chi_i \rangle + E_{xc}[\rho_0(r)]\,,$$

gdje je N broj elektrona promatranih u sustavu, χ_i je Slaterova determinanta, r su udaljenosti među česticama, Z je atomski broj za k-tu jezgru E_{xc} je energetski funkcional i predstavlja energiju izmjene i korelacije te usporedbom sa jednadžbi (2.3.1) i (2.3.2) vidi se da su se korekcijski članovi sadržani u njemu [13-15]. Puni sadržaj E_{xc} odnosi se na:

- energiju izmjene koja se odnosi na odbijanje elektrona uslijed istog spin-a,
- korelacijske energije uslijed istovremenih trenutačnih Couloumb interakcija među elektronima,

- samo-interakcijske energije uslijed spontanih i prividnih odbijanja elektrona koji imaju elektronsku interakciju samih sa sobom i
- korekcija kinetičke energije između referentnog sustava i realnog sustava u osnovnom energetskom stanju.

Moguće nepoznanice koje razlikuju sustav Kohn-Sham sustav od Schrodingerovog su sve sadržane u tom funkcionalu energije izmjene i korelacije. Teoretski odabirom ispravnog E_{xc} može se dobiti egzaktno rješenje sustava te egzaktnu vrijednost osnovne energetske razine. Problem se javlja što ne znamo koja vrijednost energije je točna pa su rezultati generalno uvijek aproksimativni doduše veliki trud se ulaže u razvoj funkcionala koji bi bili što precizniji. Neki od najkorištenijih metoda su *LDA, GGA, PBE i B3LYP* gdje se ili računaju empirjiski interpolirane funkcije ili se minimizira gradijent ili su neka hibridna kombinacija sa korektivnim konstantama [9-11].

Upravo zbog ovog zadnjeg teorema *DFT* je postao jedna od najčešće korištenih metoda za izračune u kvantnom modeliranju i tema je velikog broja radova koji se fokusiraju unapređenju modela koji aproksimiraju E_{xc} od kojih se u mnogima našlo strojno učenje. Naime najčešće u primjeni modeli strojnog učenja se koriste posredno uz klasične izračune na razini kvantne fizike i kemije. Tako se pojedini članovi funkcionala ili varijable aproksimiraju i uvrštavaju u formulu ili se aproksimiraju pojedinačni nepoznati članovi unutar samog $E_{xc.}$. Pregledom rada i teorije pojavila se ideja promatranja funkcionala gustoće i E_{xc} kao svojevrsne „crne kutije“ gdje bi se koristili algoritmi strojnog učenja u poziciji transformatora unutar „crne kutije“ za estimiranje osnovne energetske razine. Potrebno bi bilo izlučiti koje karakteristične vrijednosti elektronske strukture dostupne u bazi podataka dozvoljavaju treniranje takvog estimatora. Pokušaj bi bio ubrzati proces varijacijskog teorema direktnom transformacijom odnosno ubrzavanjem računanja energije, ili dobivanja energije iz neke od mjerljivih veličina sustava [1,10].

2.4. Repozitoriji

Popularnost korištenja *DFT*-a omogućila je sakupljanje velikog broja mjerenih i sintetiziranih podataka [12,16,17]. Koncipirana je ideja za izraditi računalni sustav za kvantne procese koji bi mogao brzo i precizno simulirati kvantne sustave direktno iz osnovnih načela. Za ovakav koncept potrebno je sakupiti što je više mogućih kvalitetno dobivenih setova podataka, ali potrebno je i isto tako stvoriti modele koji te podatke mogu iskoristiti za predviđanje. Ovakav model poželjno bi generalizirao nad cijelim poljem kvantne mehanike, ali zbog prirode treniranja modeli u svrhu strojnog učenja se strogo specijaliziraju inače gube svoja svojstva. Činjenica da je problematika složena, a alati su specifični proizlazi još jedan koncept projektiranja velikog sustava modela

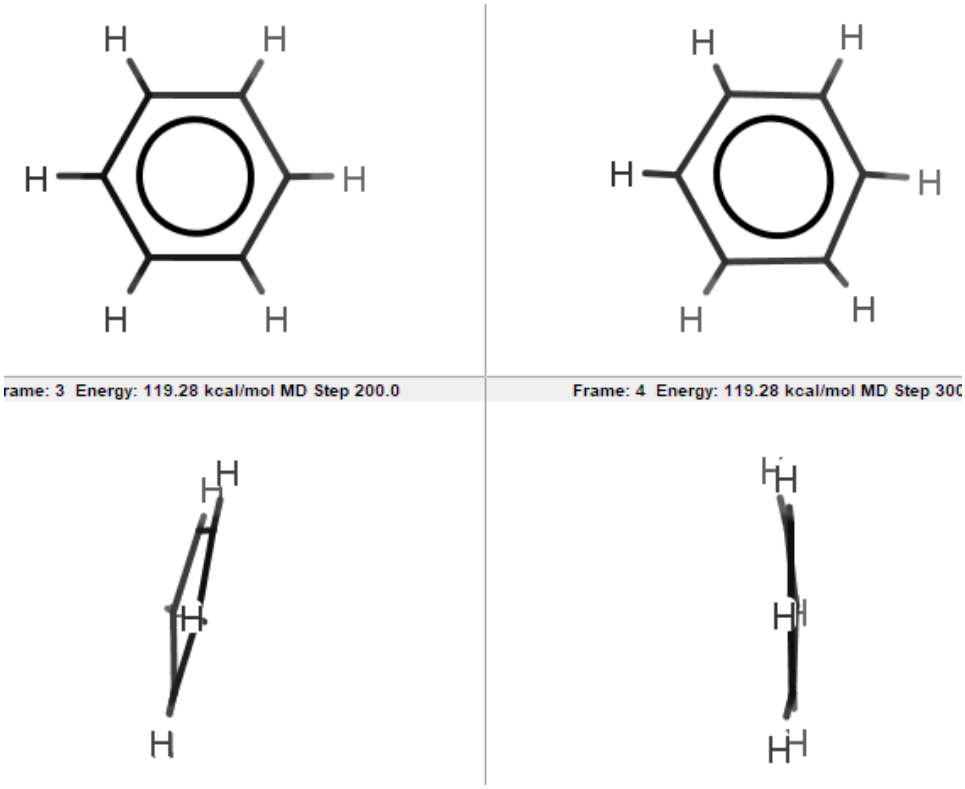
strojnog učenja koji bi se automatski odabirali i slagali u manje podsustave ovisno o problemu koji se rješava.

3. PRIKAZ I ANALIZA SETA PODATAKA

Nakon postavljanja teorijske osnove i smjera u kojem će se istraživanje odraditi, sljedeći korak za bilo koji rad baziran na strojnom učenju je odabir i obrada odgovarajućeg seta podataka. Bez adekvatnih informacija algoritmi strojnog učenja ne mogu izlučiti karakteristike sustava i generalno nisu korisni. Na drugu stranu korištenje bilo kakvih podataka također nije poželjno te je kritično obraditi podatke na takav način da se odaberu najefikasnije značajke. Obrada podataka se može svesti na automatsku koja se bazira generalno na metode transformacija i na ručnu obradu koja zahtjeva analizu predočenih podataka i odabir značajki na bazi intuicije istraživača. Za početak set podataka će se opisati i vizualizirati popraćeno početnom analizom i predobradom.

3.1. Opis seta podataka

Set podataka korišten u radu imenovan „*Densities dataset*“ proizašao je iz većeg *MD17* seta koji se koristi za proučavanje molekularne dinamike manjih molekula [12,16,17]. *Densities* set podataka odabran je prvenstveno zato što sadržava podatke o *DFT* elektronskoj gustoći koji nisu dio drugih setova. Sve vrijednosti u setu podataka dobivene su računanjem na standardnim programskim paketima za molekularnu dinamiku i računanja elektronske strukture. Set podataka podijeljen je na podatke pojedinačnih molekula. Podatci su generirani nasumičnim odabirom molekularnih konformacija i računanjem odgovarajućih vrijednosti elektronskih struktura za svaku konformaciju. Kemijske konformacije predstavljaju vrstu izomera gdje iste molekule mogu imati prostornu rotaciju oko dijelova molekula sa primjerom na Slici 3.1. Činjenica da imaju pomak u prostoru može utjecati na njihove elektronske karakteristike. *DFT* elektronska gustoća razložena je Fourierovom transformacijom u formatu baznih koeficijenata. *DFT* energija i *DFT* gustoća izračunate su korištenjem *PBE* aproksimacije E_{xc} funkcionala. Dodatno je izračunata i energija dobivena CC metodom koja služi za usporedbu sa *DFT* metodom pošto je *CC* metoda preciznija [1,2].



Slika 3.1 Primjer različitih izomera za molekulu benzena

3.2 Vizualizacija seta podataka

Podatci su spremljeni u formatu *.npy* koji zahtjeva korištenje biblioteke *numpy* [18] iz programskog jezika *Python* [19] koji će biti korišten za vizualizaciju i obradu. Učitani set podataka u programskom sučelju je matričnog zapisa sa pripadajućim vrijednostima za 1001 jedinstvenu konformaciju gdje su dimenzije tih matrica prikazane u Tablici 3.1. sa pridodanim mjernim jedinicama.

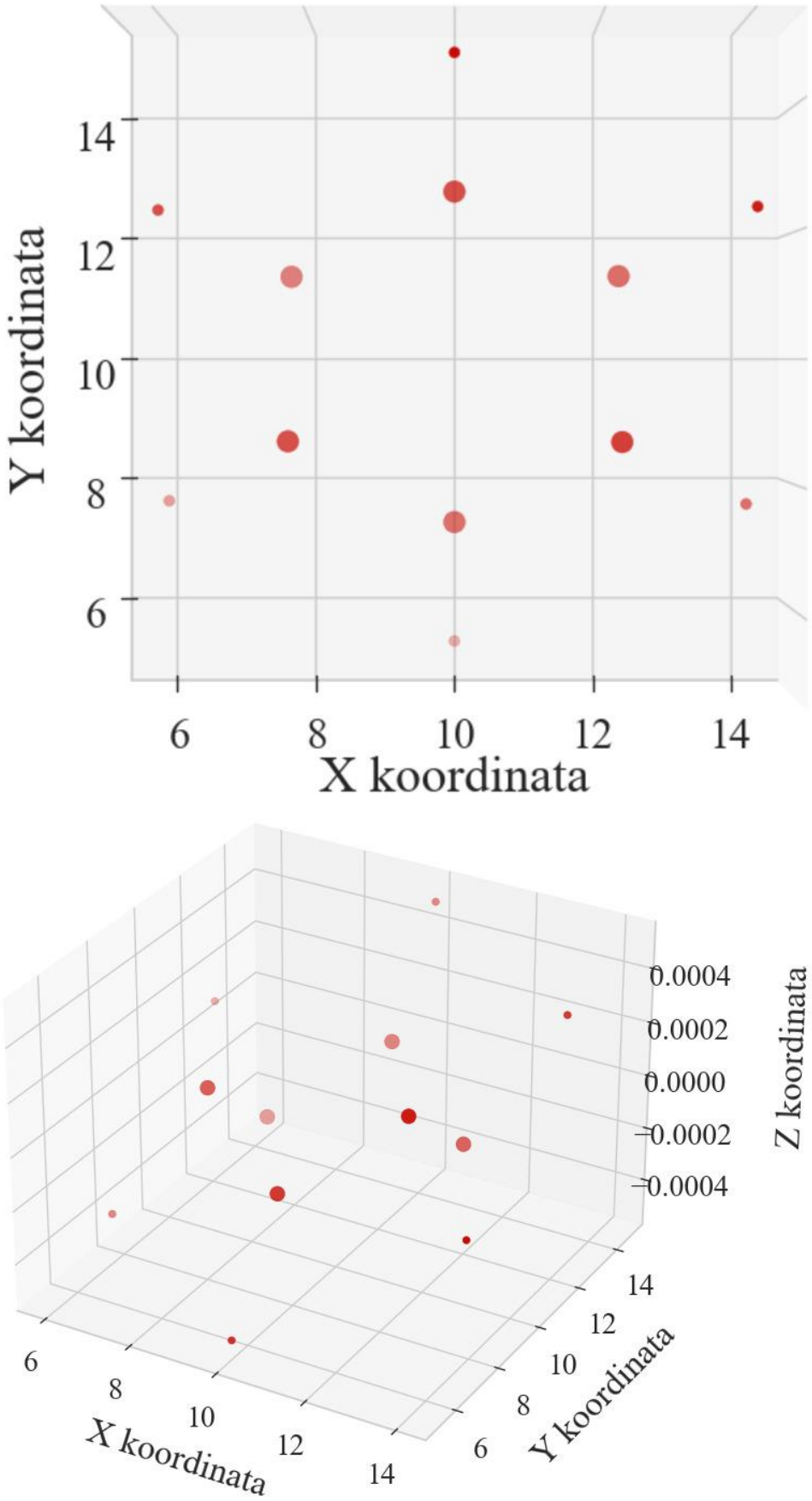
Tablica 3.1 Dimenzije matrica strukture i elektronske karakteristike za molekulu benzena

Naziv matrice	Dimenzija	Mjerna jedinica
Structures.npy	[1001 , 12 , 3]	*Bohr*
DFTdensity.npy	[1001, 125000]	/
DFTenergy.npy	[1001, 1]	*Kcal/mol*
CCenergy.npy	[1001, 1]	*Kcal/mol*

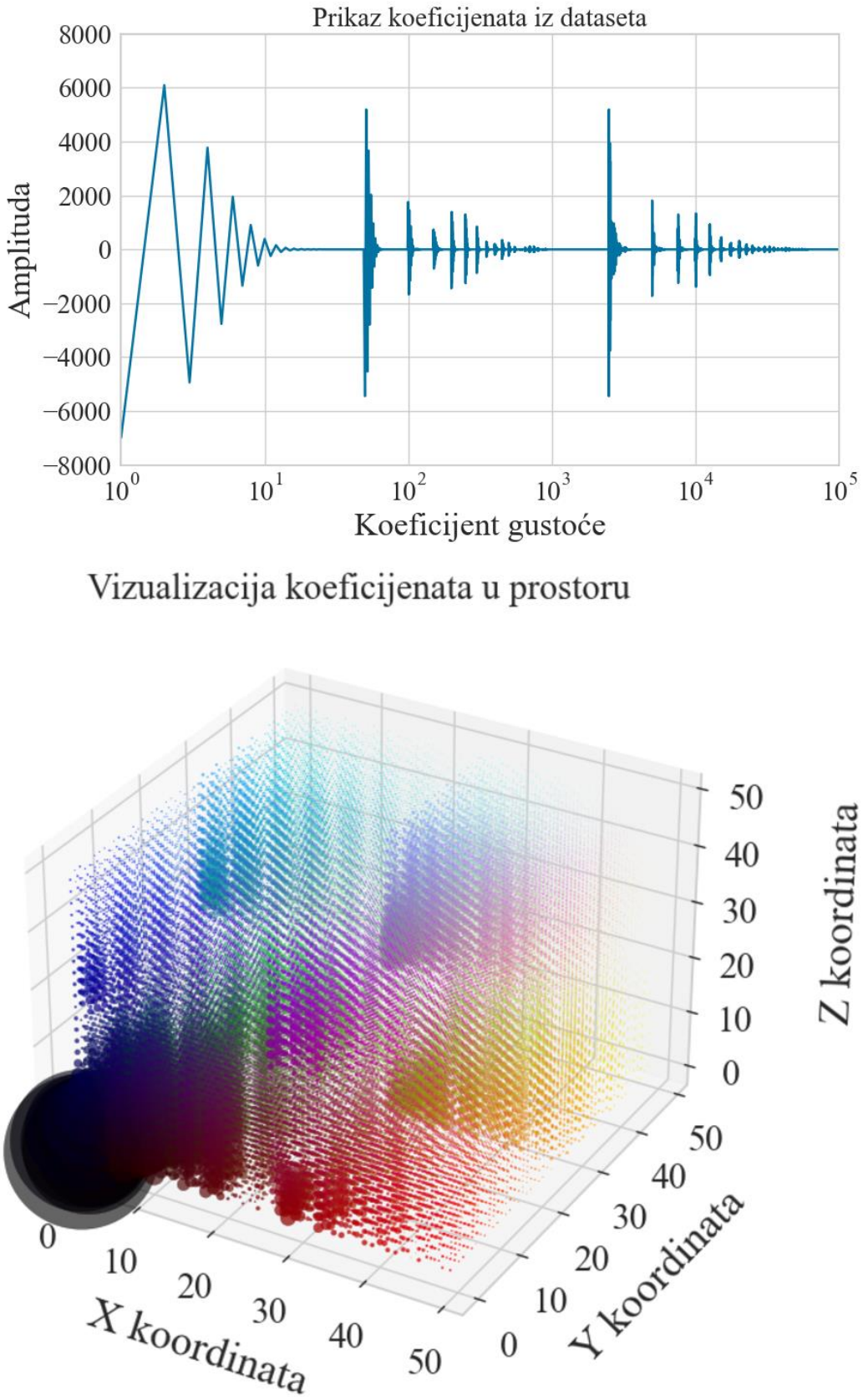
Da bi se bolje predočilo što te matrice znače i o kakvim vrijednostima je riječ ti podatci će se prikazati i grafički. Strukture benzena dane su u XYZ formatu koje označavaju pozicije svakog od 12 atoma. Izgled jedne od 1001 geometrije u prostoru prikazana je na Slici 3.2 gdje su veći krugovi mjesta atoma ugljika a manji atomi vodika. Na pogledu u izometriji se vidi varijacija pozicije atoma, a na pogledu iz tlocrta se vidi oblik molekule. Koeficijenti elektronske gustoće razloženi

su u jednoj dimenziji za svaku geometriju kao pojednostavljenje za primjenu u strojnom učenju. Ti koeficijenti su Fourierova transformacija prostorne vrijednosti, a originalno su dobiveni u 3D prostoru sa 50 koeficijenata po dimenziji prostora (50x50x50 matrica). Kako bi se najbolje vizualizirala situacija odrađeno je re-formatiranje matrice te su *1D* i *3D* koeficijenti prikazani na Slici 3.3. u *3D* prikazu veličina točke predstavlja intenzitet a boja predstavlja skupinu kojoj vrijednosti pripadaju. Zbog velike količine podataka oba prikaza imaju slabu vidljivost, ali se nazire priroda prostorne gustoće gdje je očito velika vrijednost bliže molekuli sa naglim padom intenziteta i da je gustoća kuglastog oblika.

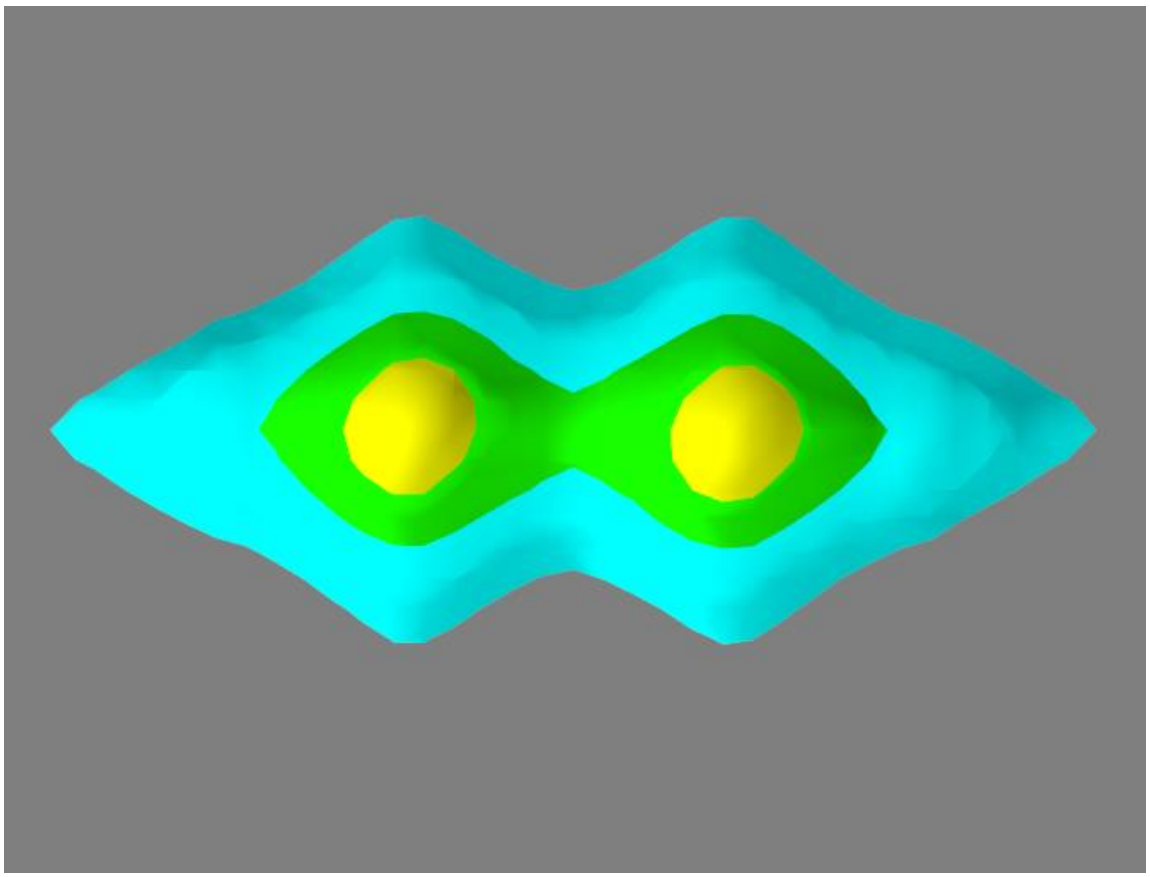
Da bi se vizualizirao pravi oblik gustoće odrađena je više-dimenzijska inverzna Fourierova transformacija i dobivena je matrica koja je renderirana u *3D* obliku i prikazana na Slici 3.4. *3D* prikaz je presječen i rastvoren kako bi unutarnji oblik bio vidljiv te prikazana priroda prostorne elektronske gustoće gdje je najveći intenzitet neposredno oko molekule. Preostalo je još prikazati energije koje su prikazane na Slici 3.4 za svaku geometriju. Pomoćne crte služe za lakše promatranje gdje se nalaze vrijednosti energije. Prikazana je samo *DFT* energija pošto *CC* energija identično prati putanju *DFT* energija samo za umanjene vrijednosti.



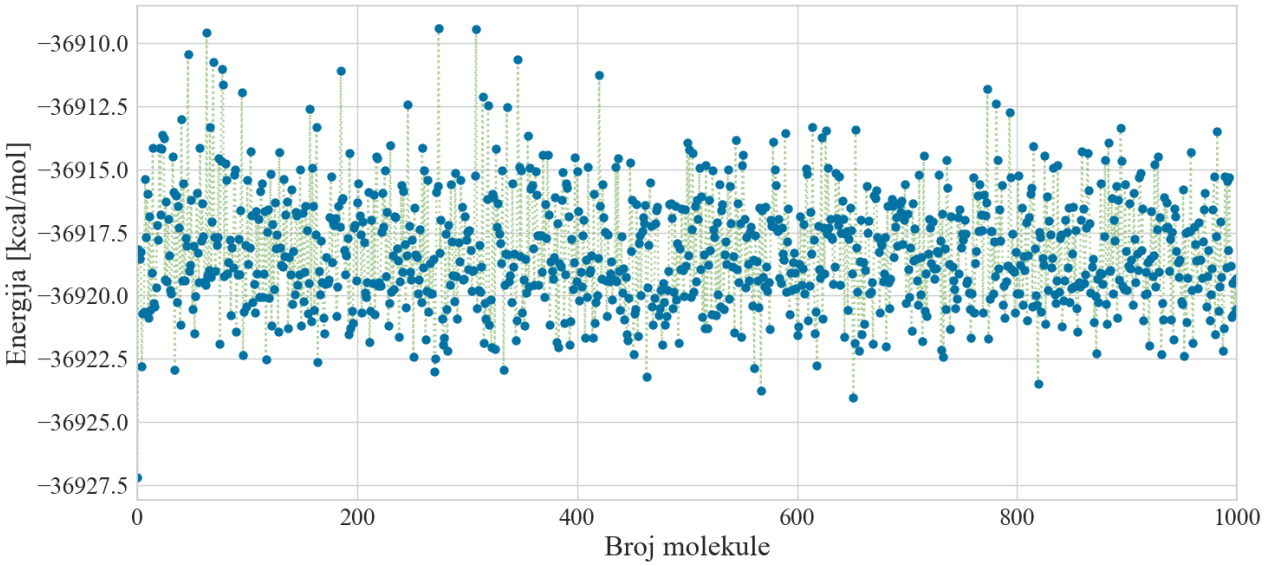
Slika 3.2 Geometrija molekule benzena po prostornim koordinatama



Slika 3.3 Prikaz elektronske gustoće u 1D obliku (gore) i 2D obliku (dole)



Slika 3.4 3D render površine elektronske gustoće



Slika 3.5 Graf DFT energije za pojedinačnu konformaciju

3.3 Analiza i pred-obrada seta podataka

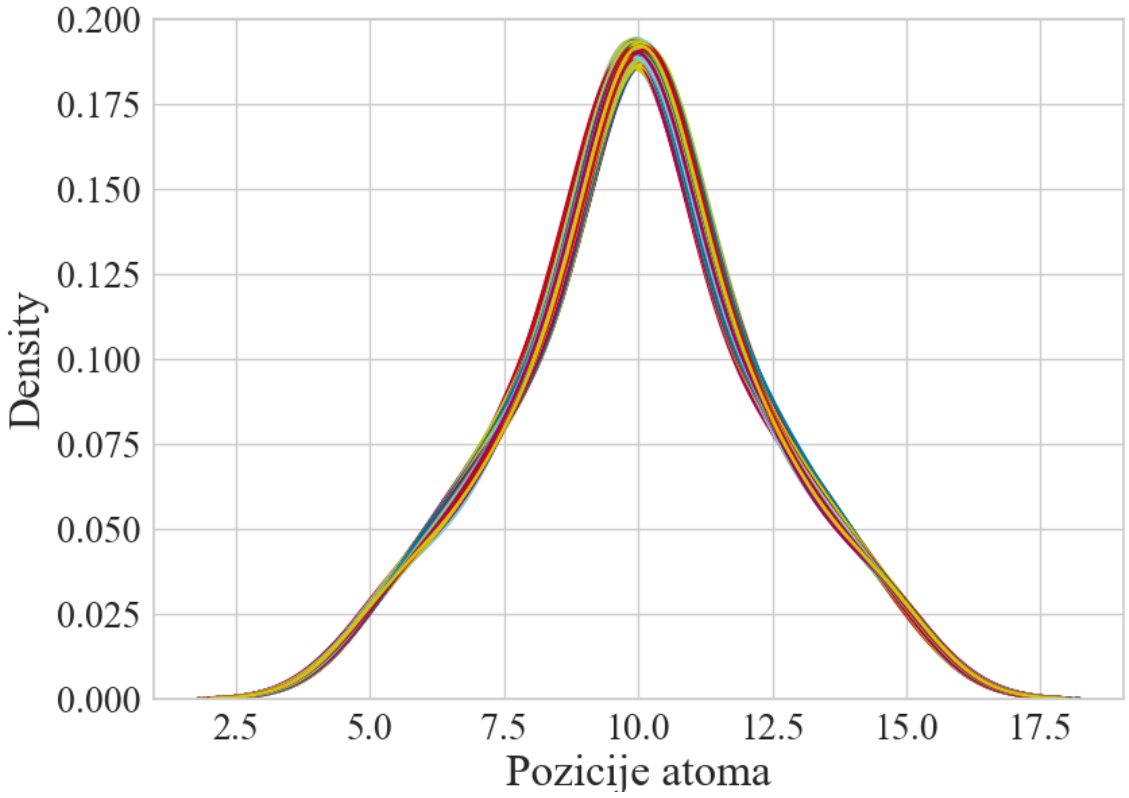
Vizualizacija podataka dala je dobar uvid u situaciju te se može iščitat nekoliko problema koje treba uzeti u obzir za pripremu podataka za bilo kakve modele strojnog učenja. Glavni problem je činjenica da ima značajno više značajki (stupaca) nego uzoraka (redova) odnosno javlja se $p(125000+) >> n(1001)$ problem [20,21]. Ovakva situacija otežava treniranje modela strojnog učenja zbog statističke prirode njihovih algoritama. Posljedica tog problema je najčešće nemogućnost generalizacije modela gdje algoritam uđe u zasićenje zbog prevelike količine statističkog šuma i počne imati lošije predikcije. Isto tako mala količina uzoraka može uzrokovati slabu statističku relevantnost značajki koja bi opet stvorila uvjete za lošu predikciju. Također još jedan problem koji nije za zanemariti je veliki broj značajki predstavlja veliki broj ulaznih varijabli koje treba u isto vrijeme obraditi što može izazvati eksponencijalno duže vrijeme treniranja modela.

U određenim slučajevima ovakav raspored značajki naspram uzoraka ne može se izbjeći te treba svjesno izabrati koje vrijednosti će se filtrirati. Iz prvog pogleda doduše vidi se da je potrebno izvršiti redukciju dimenzija nekom od dostupnih metoda i da je ovakav set podataka dobar kandidat za korištenje algoritama koji rade regularizaciju o čemu će biti više rečenu u sljedećim poglavljima.

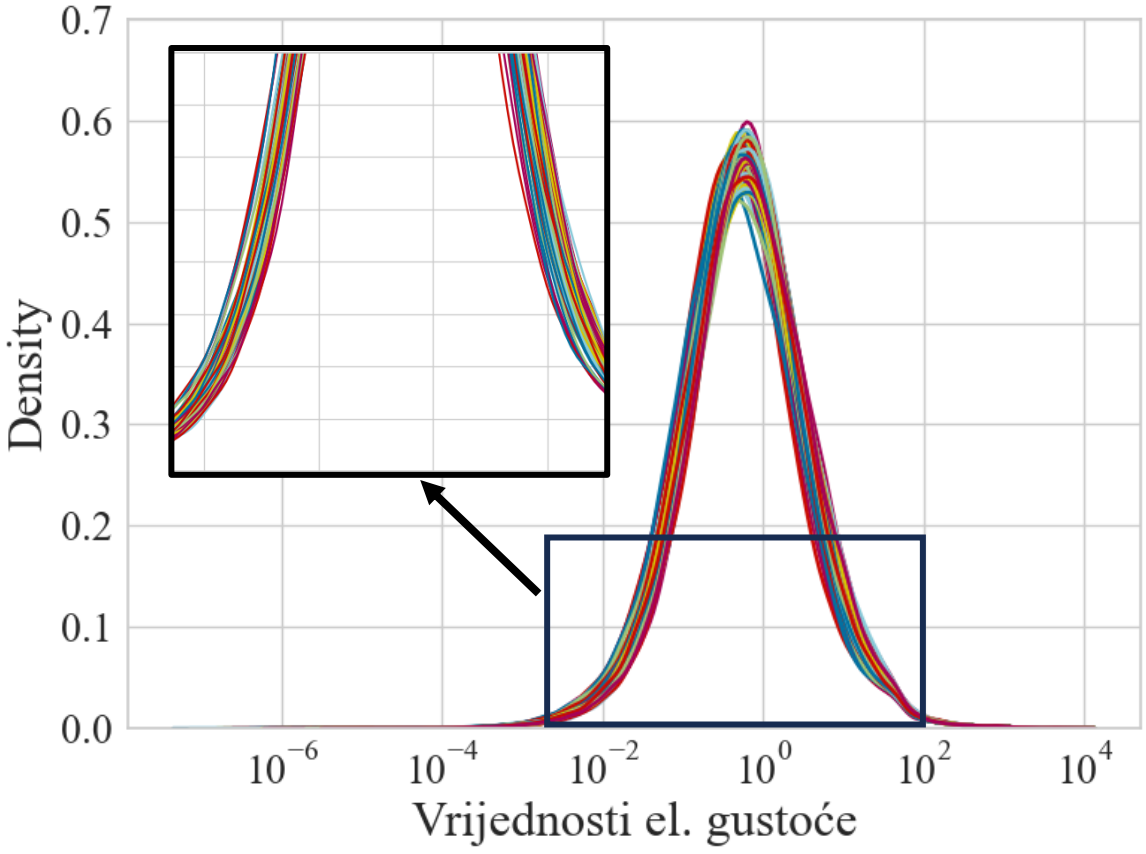
3.3.1 Netipične vrijednosti

Za prvi korak treba se provjeriti statistička distribucija seta kako bi se provjerila kvaliteta podataka koji se u njemu nalaze. Zbog velike količine podataka nepraktično je odraditi klasičnu statističku analizu minimuma, maksimuma, standardne devijacije i slično jer je to nepraktično vizualizirati i ručno provjeriti. Naime uzimajući pretpostavke da zbog same prirode kvantne fizike i statističkih metoda same vrijednosti varijabli morale bi imati generalno standardnu raspodjelu vrijednosti. Koristeći analizu gustoće vjerojatnosti može se vidjeti distribucija značajki te ako ona nalikuje standardnoj distribuciji može se zaključiti da su podatci kvalitetno dobiveni i da nema odstupanja i netipičnih vrijednosti [21,22].

Analiza gustoće vjerojatnosti za geometrije prikazana je na Slici 3.6, a za elektronsku gustoću sa logaritamskom skalom na Slici 3.7. Na grafovima se može vidjeti krivulja koja nalikuje standardnoj devijaciji iako na Slici 3.7 se vidi pomak krivulje prema desno i logaritamska skala ukazuje na to da šiljak prevladava nad ostalim vrijednostima. To ukazuje da ima jako puno značajki sa vrlo malim vrijednostima i da je omjer između najveće i najmanje ekstreman. Usprkos tome ne vidi se utjecaj ili prevladavanje netipičnih vrijednosti, a razlike u magnitudama mogu se transformirati skaliranjem i normalizacijom ako se pokaže potreba.



Slika 3.6 Estimacija gustoće vrijednosti značajki strukture molekule



Slika 3.7 Estimacija gustoće vrijednosti značajki elektronske gustoće

3.3.2 Korelacija

U svrhu statističke analize i uvida u utjecaj značajki uvodi se primjena analize Pearson korelacijskih koeficijenata [22]. Pearson korelacija ispituje snagu linearne asocijacije između dvije varijable. Vrijednosti Pearson korelacije nalaze se između -1 i 1 koji bi označavali savršenu negativnu korelaciju i savršenu pozitivnu korelaciju a 0 bi označavalo da nema korelacije. Za svrhu strojnog učenja poželjno je da značajke imaju tendenciju snažnije korelacije neovisno da li je ona pozitivna ili negativna. Pozitivna stvar kod Pearson korelacije je svojstvo neovisnosti o mjernim veličinama i skaliranju ili o tome ako je varijabla ovisna ili neovisna. Doduše to može biti i negativna stvar jer onda se iz toga ne može definitivno odrediti kauzalnost varijable. Jedna od metoda za vizualizaciju korelacijskih vrijednosti je korištenjem toplinske mape (engl. *heatmap*) koja iscrtava obojane ćelije u formatu matrice gdje je intenzitet boje povezan sa iznosom korelacijskog koeficijenta promatranih varijabli.

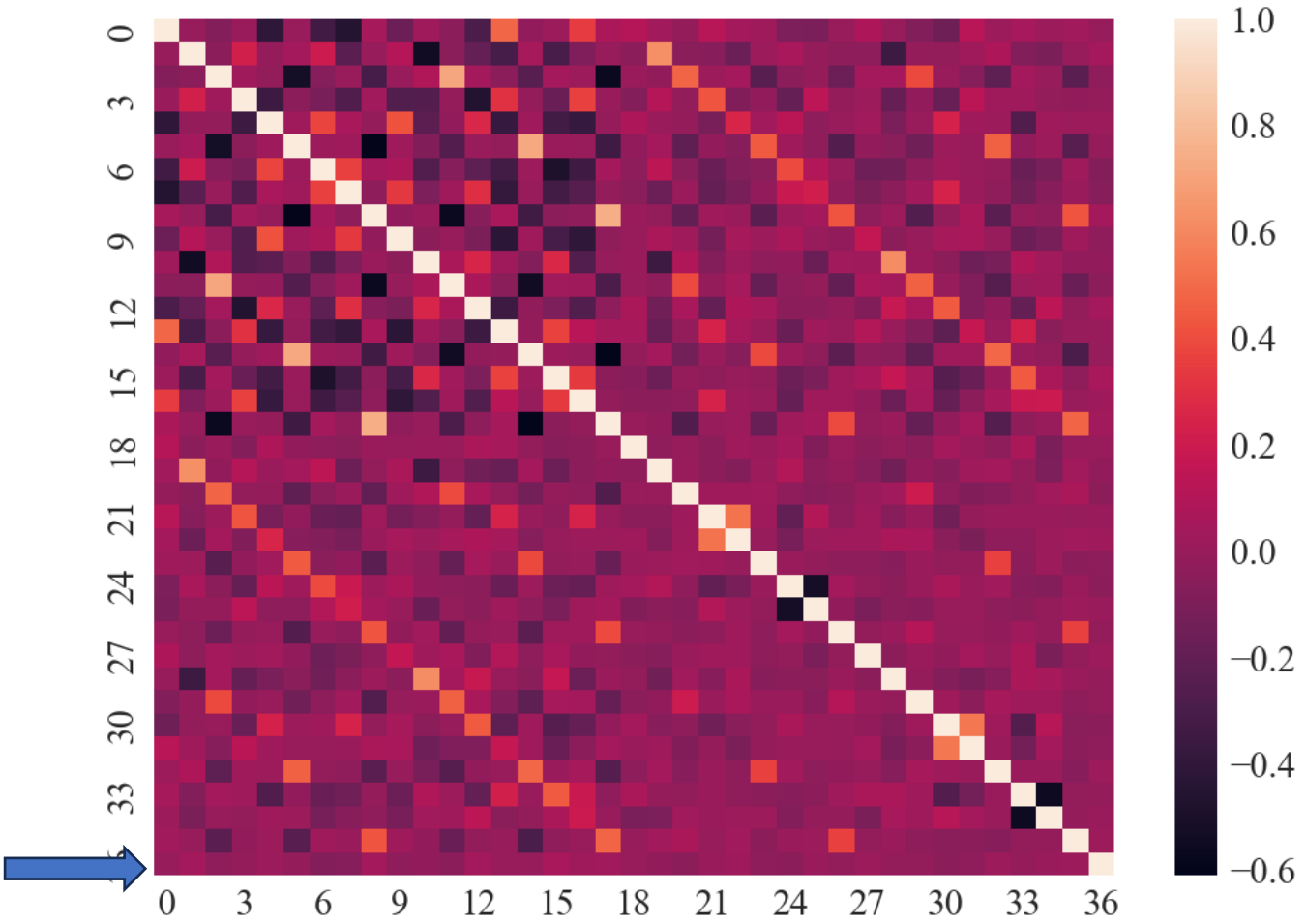
Korelacija je izvršena spajanjem promatranog seta podataka sa podatcima energije kako bi se usporedila korelacija između struktura i energije i elektronske gustoće i energije. Za izračunati korelaciju struktura naspram energije nije problem učiniti te je prikazana na Slici 3.8. Vrijednosti od interesa su na zadnjem redu odnosno stupcu te se vidi da je korelacija praktički nepostojeća pa je šansa da će te varijable dati koristan doprinos vrlo mala. Kako bi se pokušalo izvući što je više relevantnih informacija iz podataka odrađena je transformacija strukture u Couloumb matricu kao jednostavni globalni deskriptor koji proširuje polje strukture u polje elektrostatskih interakcija. Couloumb matrica se računa sljedećom formulom [2,8]

$$M_{ij} = \begin{cases} 0.5Z_i^{2.4} & za\ i = j \\ \frac{z_i z_j}{R_{ij}} & za\ i \neq j \end{cases}, \tag{3.1}$$

gdje je Z atomski broj jezgre (6 za C, 1 za H), R je pozicija atoma. Skripta za računanje matrice napisana je ručno te je iterirana za svaku strukturu kako bi se mogla izvršit korelacija deskriptora s energijom. Dobivena korelacijska matrica prikazana je na Slici 3.9 a vrijednosti od interesa su također u zadnjem redu odnosno stupcu. Iz prikazanoga se vidi da kodiranje strukture deskriptorom nije pomoglo te da još uvijek vrijednosti jako slabo koreliraju.

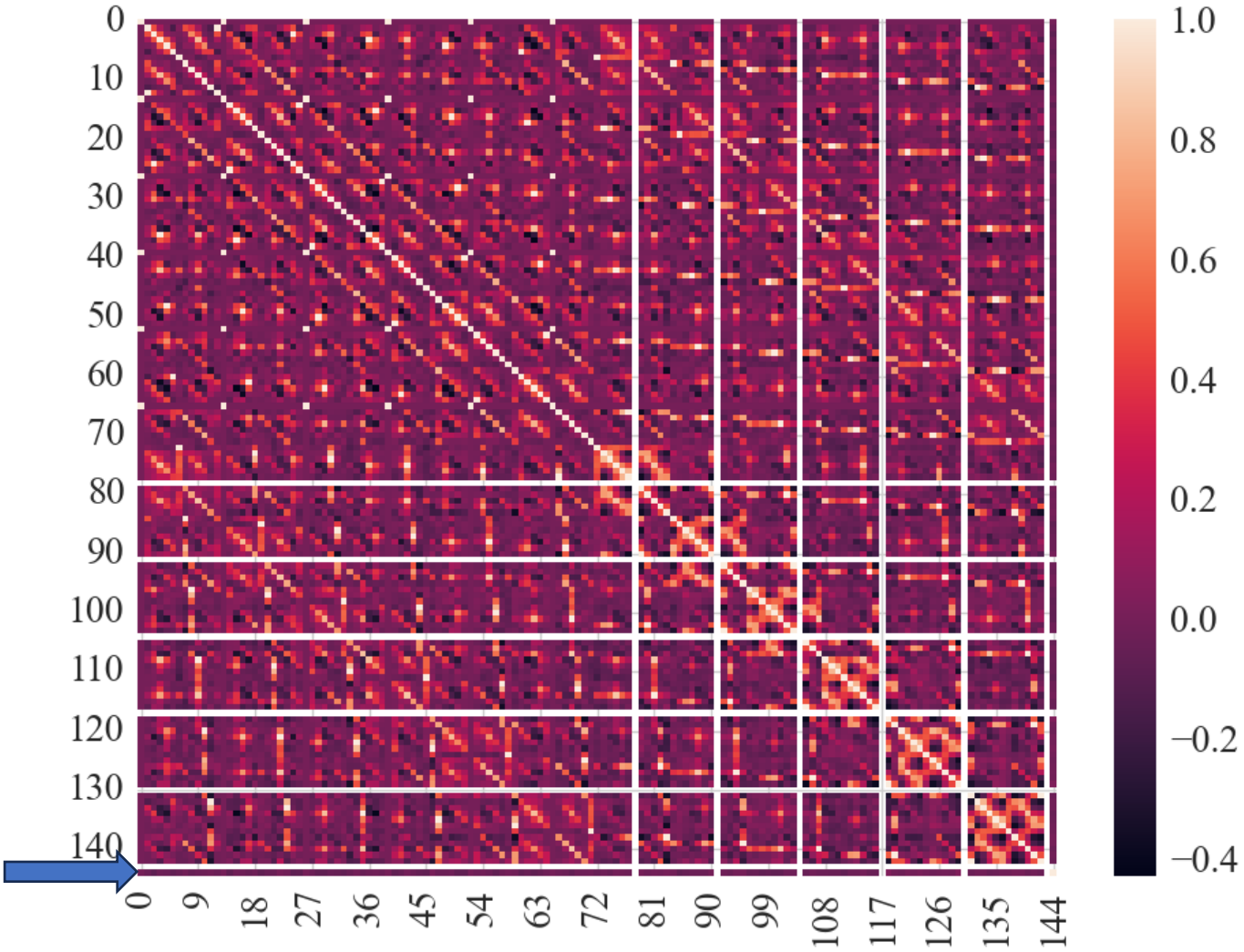
Preostalo je još provjeriti stanje sa elektronskom gustoćom. Teorijom koja je odrađena u prvom poglavlju pokazana je povezanost elektronske gustoće preko funkcionala sa svim drugim članovima te da sadrži sve potrebne informacije za računanje Hamiltonijana. Logično je za naslutiti da određena količina značajki mora direktno utjecati na izlaznu varijablu energije. Za odrediti korelacijsku matricu ručno je napisana skripta koja je podijelila set podataka na manje dijelove na kojima se iterativno izvršavala korelacijska analiza preko kojih su se izlučili indeksi

značajki koje koreliraju sa vrijednošću manje od -0.4 ili više od 0.4. Time se zaobišao zahtjev za 106 GB radne memorije i izuzetno nepreglednog *heatmap*-a. Izlučivanjem indeksa korelacije dobiveno je 1697 značajki koje koreliraju bolje od ±0.4. Jedan od manjih blokova korelacije prikazan je na Slici 3.10 kako bi se vidjela asocijacija među značajkama. Na prikazu se vidi još jedan potencijalni problem u vidu multi-kolinearnosti gdje je puno značajki linearno ovisno jedno o drugome što može stvoriti problem kod treniranja algoritama zbog velike osjetljivosti na ažuriranje koeficijenata [23,24].

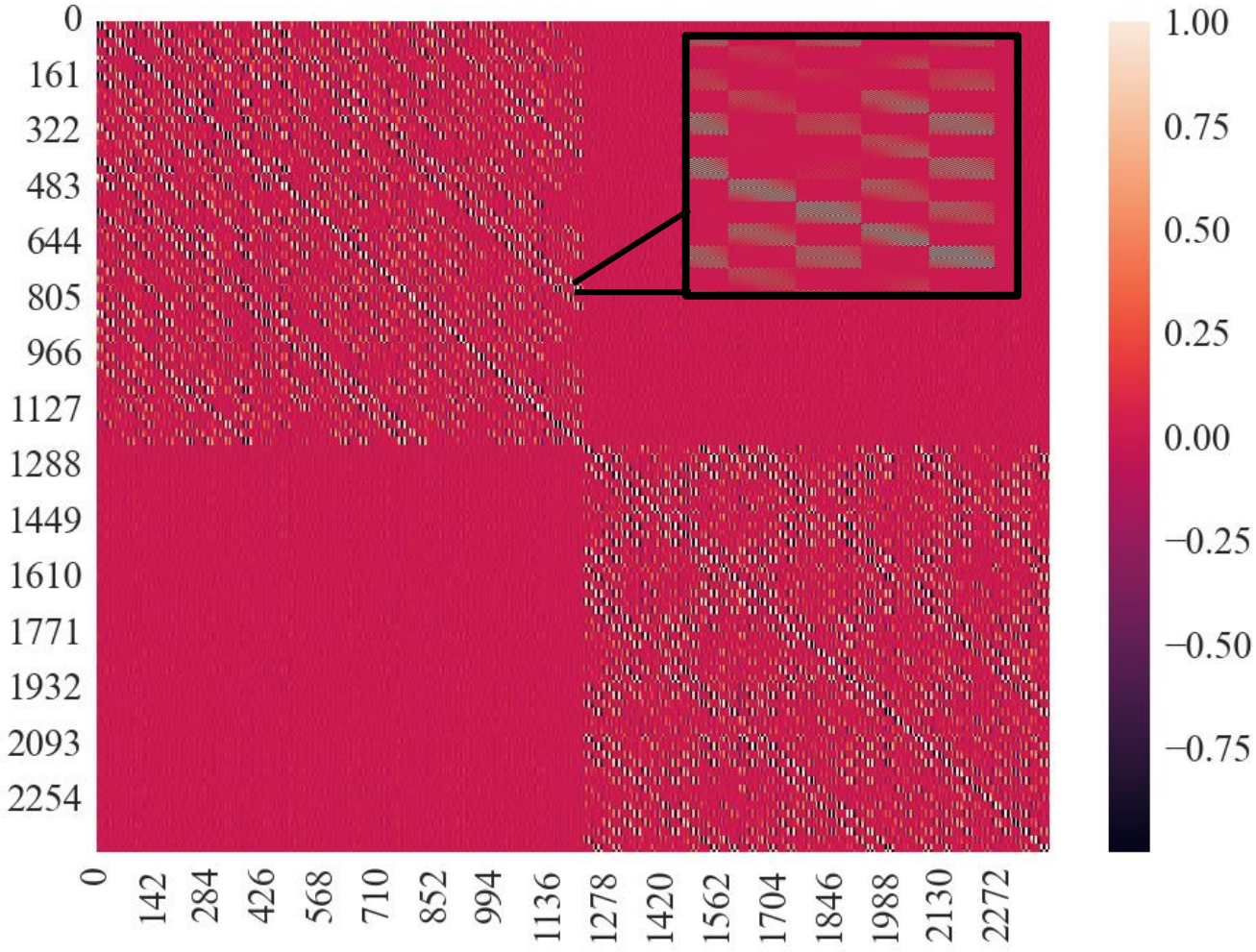


Slika 3.8 Korelacijski heatmap za pozicije atoma

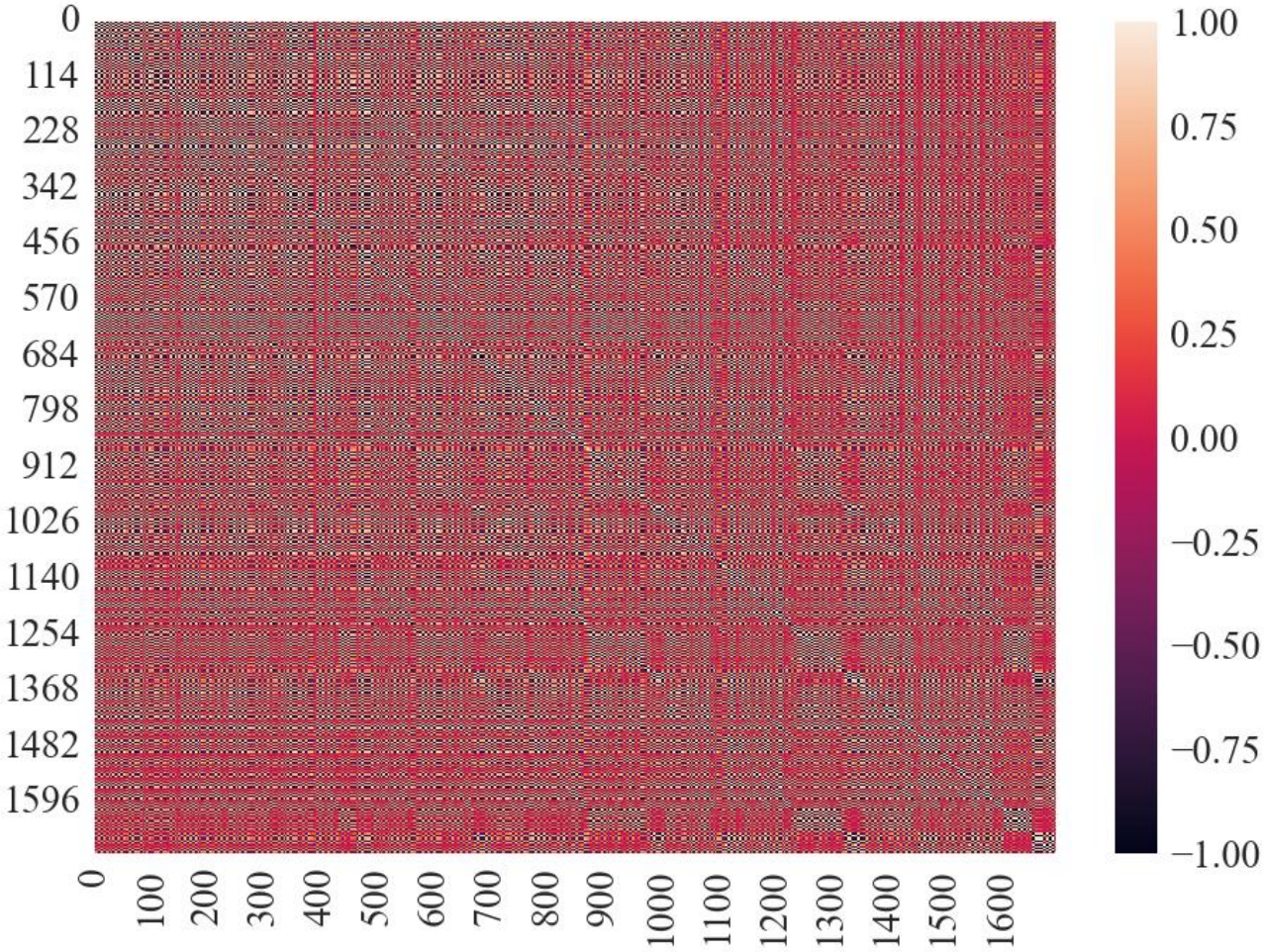
Podatci koji nisu prikazani su matrica *CCenergija* koja ima praktički savršenu linearnu korelaciju sa *DFT* energijom jer kao što je prethodno rečeno te dvije energije imaju identičnu krivulju samo su drugih amplituda. Korelacijska matrica za slučaj ±0.4 prikazana je na slici 3.11.



Slika 3.9 Korelacijski heatmap za Couloumb deskriptore



Slika 3.10 Korelacijski heatmap za gustoće u zadnjem bloku



Slika 3.11 Prikaz heatmap-a za vrijednosti korelacije energije ±0.4

4. REDUKCIJA DIMENZIJA I SKALIRANJE PODATAKA

Kao što je već spomenuto jedan od većih problema koja su se pojavila u radu i koji generalno opterećuju modele strojnog učenja je slučaj značajno većeg broja značajki od uzoraka ($P>>N$). U procesu analize seta podataka tokom odrađivanjem korelacijske analize pronađeno je da velika količina podataka nema nikakvu korelaciju sa varijablom koju se pokušava estimirati te je time odrađen svojevrsni korak redukcije dimenzija. Usprkos činjenici da korelacija traži samo linearnu ovisnost varijabli odlučeno je da će se minimalno korelirajuće varijable odbaciti kako se spriječilo učenje model na ekvivalentu šuma u podatcima [20,21].

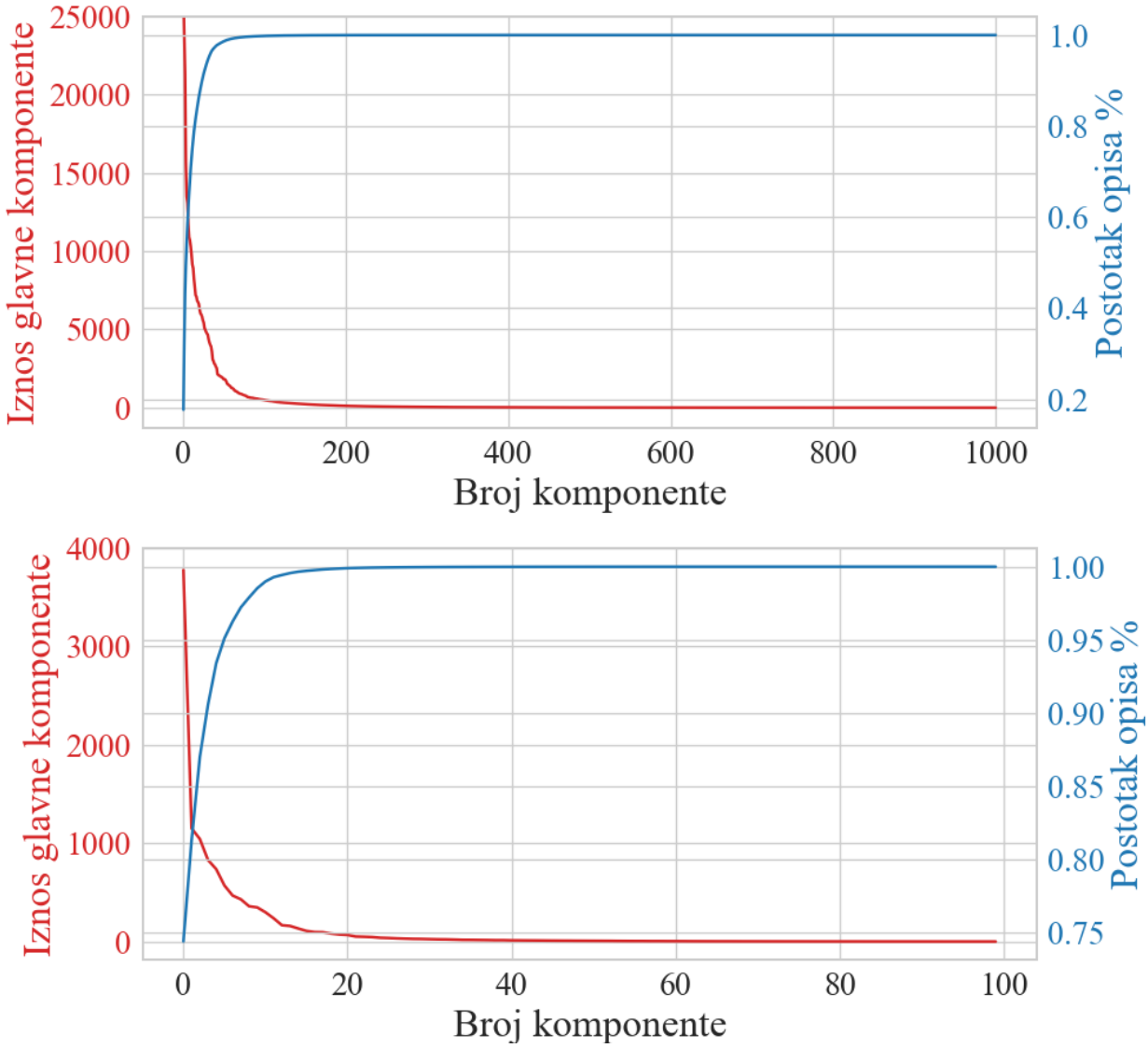
U literaturi je uvrženo mišljenje da je odabir korelacijskih varijabli vrijednosti veće/manje od ±0.6 u tu svrhu dobar početak [21,26]. Analizom je pronađeno da takvim odabirom broj ulaznih varijabli iznosi 41 značajku što je drastična razlika od 1697 značajki za korelaciju ±0.4. Isto tako u literaturi je rečeno da za modele nije najbolje da sadržavaju isključivo visoko koleirajuće značajke zbog mogućnosti pre-treniranja modela [25,26], a uz to baza podataka koja je velika i ima mogućnost nelinearnih karakteristika gube se informacije drastičnim oduzimanjem značajki [10]. Zbog ovih razloga uzet je set podataka sa više sadržanih značajki kako bi se omogućila šansa generalizacije modela. Uzevši u obzir da je još uvijek prisutna situacija p>n čak i nakon toga primijeniti će se vrsta redukcije dimenzija koja transformira podatke na takav način da pokuša izlučiti karakteristiku podataka i izvrši smanjenje dimenzija baziranu na tome.

4.1 Analiza glavnih komponenti (engl. *Principal component analysis - PCA*)

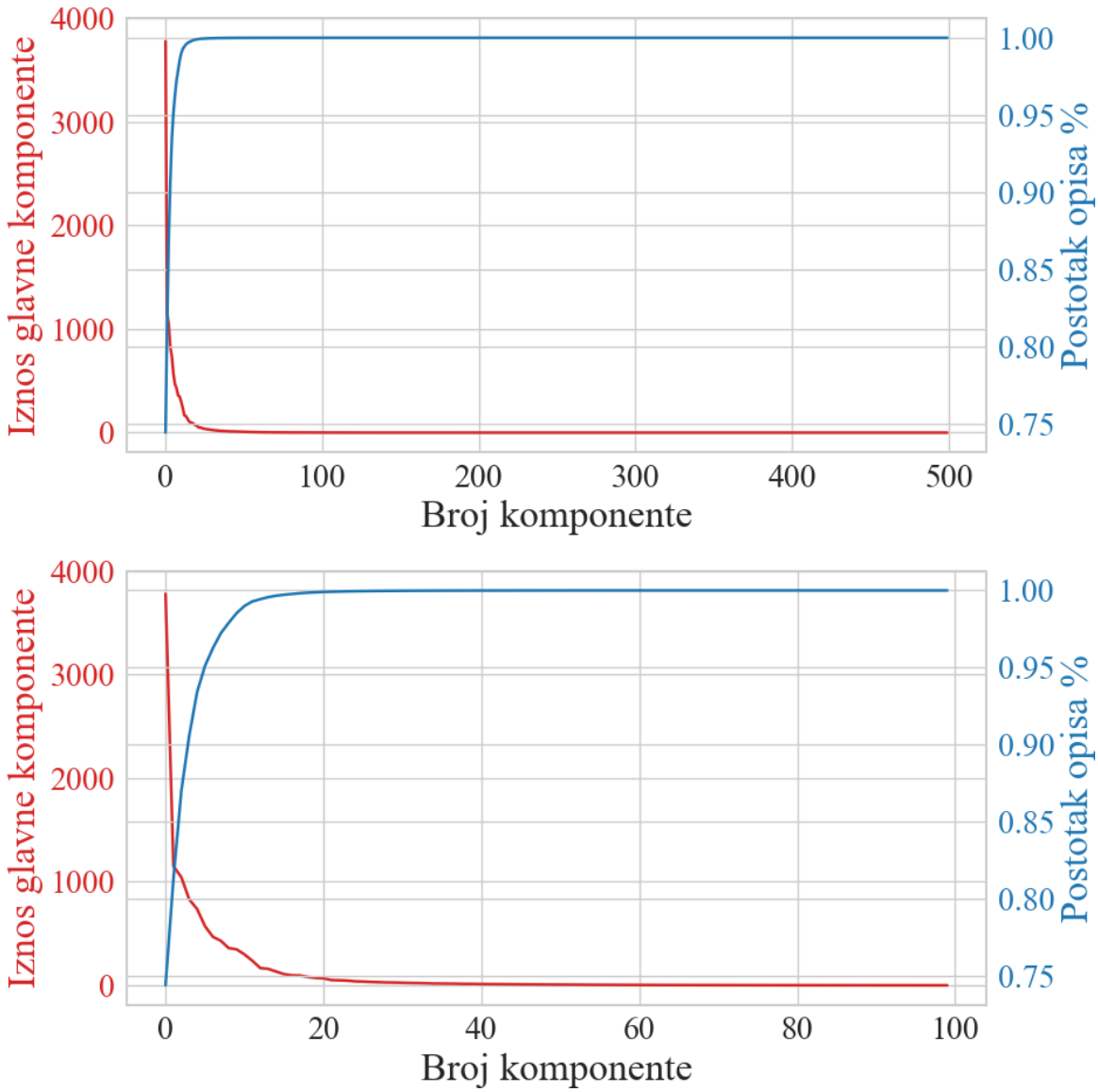
Jedna od glavnih metoda redukcije dimenzija je korištenje *PCA* te je dostupna ugrađena funkcija u programskom paketu korišten u ovom radu za strojno učenje. *PCA* je metoda nenadziranog traženja linearne transformacije koji koristi jedinične eigenvektore kako bi transformirao podatke u prostoru značajki. Karakteristika ove metode je što zadržava maksimalno informacija varijance značajki i dobro se skalira na podatcima koji imaju Gauss raspodjelu [21,26,27].

PCA radi tako da pokuša naći vezu između značajki u setu koreliranih podataka tako da pronađe set nekoreliranih značajki koji se zovu glavne komponente (engl. *principal components*). Te glavne komponente imaju svojstvo sadržavanja informacija o varijanci značajki. Prvi takav vektor glavnih komponenti sadržava najviše informacija o varijanci ukupnog seta podataka te je najutjecajniji. Takvih vektora ima *n* komada i svaki sljedeći vektor opisuje podatke određenim postotkom koji je manji od prethodnog i ima toliko komponenti koliko treba da se opiše cijeli set podataka. Odrađena je *PCA* analiza na cijeli set podataka i na korelacijom smanjeni set podataka gdje se vidi slična karakteristika za oboje na Slikama 4.2 i 4.3. Činjenica da je slična karakteristika,

odnosno da *PCA* metoda pronalazi slične glavne komponente na ukazuje da korelacija ne uništava pretežito korisne značajke i da korelirani set je opisan sa manje komponenti pa će se vršiti treniranje na modelima sa jednim i sa drugim setom podataka i usporediti će se rezultati. Najčešće se koristi samo dio glavnih komponenti, ali u ovom slučaju ima mali broj uzoraka te će se morat odradit pretraga broja komponenti ovisno o rezultatu treniranja.



Slika 4.1 Prikaz PCA komponenti za puni set podataka

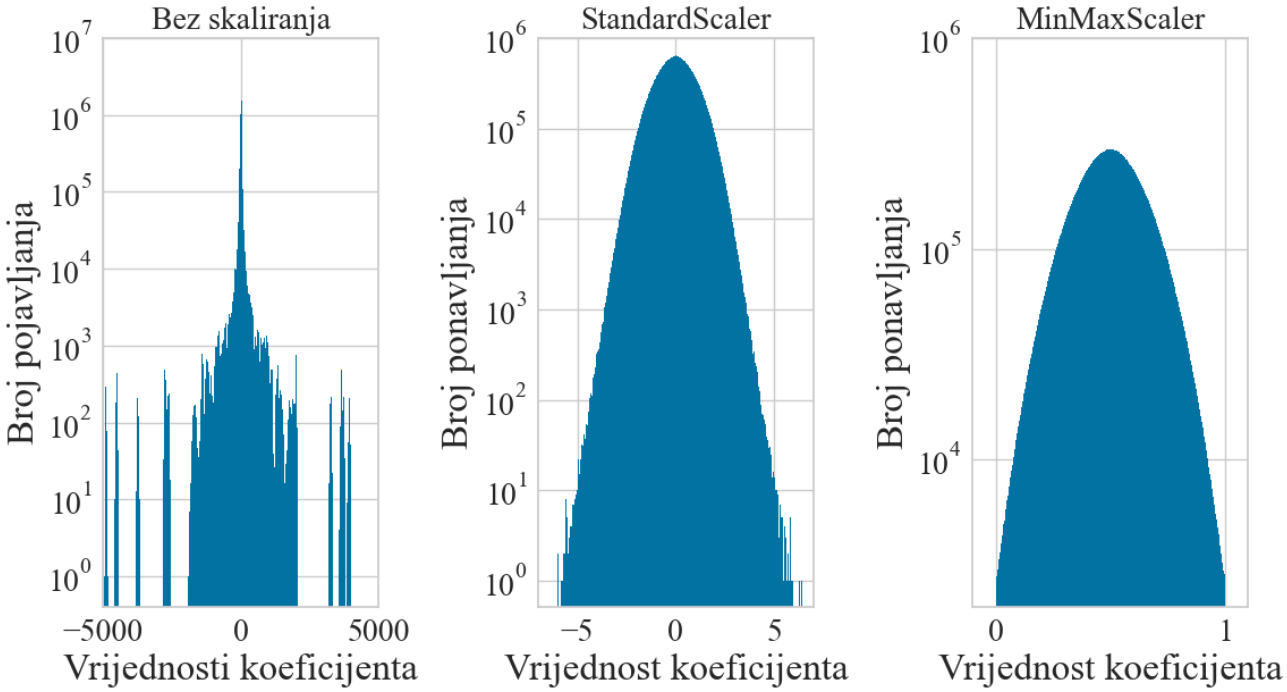


Slika 4.2 Prikaz PCA komponenti za korelirani set podataka

4.2 Skaliranje i normalizacija

Skaliranje i normalizacija nisu dio smanjenja seta podataka, ali su dio metoda obrade koje omogućuju algoritmima da bolje izvršavaju učenje[26]. Promatrane i iskorištene metode skaliranja u radu su dio ugrađenih funkcija biblioteke za strojno učenje te se zovu *StandardScaler* [28] i *MinMaxScaler* [28]. Naime uzevši u obzir da odabrani modeli imaju dobro svosjtvo regularizacije te da podatci odabrani za ulaze u model nisu u različitim mjernim veličinama utjecaj *StandardScaler* transformacije neće značajno utjecat na rezultate, ali je dobra praksa standardizirati podatke dokle god ne počne negativno utjecati na učenje modela. *MinMaxScaler* isključivo je korišten sa *SVR* algoritam jer je pokazao značajno poboljšanje u radu modela što je i preporučeno u uputama [28].

Transformacija *StandardScaler* oduzima srednju vrijednost sa svih varijabli i tako centrira podatke nakon čega skalira podatke na jediničnu varijancu čime se podatci transformiraju da spadaju u standardnu Gaussijsku raspodjelu. *MinMaxScaler* skalira podatke tako da translatira podatke između određenih vrijednosti što je u ovom slučaju u rasponu [0,1]. Utjecaj skaliranja prikazan je na Slici 4.4 gdje se vidi histogram vrijednosti (1000 podjela) za podatke gustoće transformirane *StandardScalerom* i *MinMaxScalerom.*



Slika 4.3 Histogram transformacije seta podataka gustoće metodama skaliranja

5. METODE STROJNOG UČENJA

Nakon obrade teorije i analize seta podataka dobije se bolji uvid u problematiku zadatka. Zbog same karakteristike i vrste informacija koja je sadržana u setu podataka postavljaju se određeni zahtjevi na metode učenja koje će utjecati na odabir algoritama.

1) Cilj treniranja je dobiti vrijednost skalarnog broja iz drugih skalarnih brojeva što znači da će se koristiti modeli regresije [26].
2) Veliki broj značajki sa velikom razlikom između minimalnih i maksimalnih vrijednosti i problem kolinearnosti zahtjeva korištenje robusnih modela sa snažnom regularizacijom [23].
3) Mali broj uzoraka nagovještava korištenje unakrsne-validacije za poboljšanje rezultata uslijed relativno malog testnog uzorka [26].
4) Dobra standardna raspodjela podataka govori da normalizacija podataka možda nije nužna, ali da će bit korisno ispitati metode skaliranja.
5) Naizgled kompleksni funkcionali koji su osjetljivi na promjenu koeficijenata nagovještaju moguću osjetljivost na promjenu hiperparametara [8].
6) P>>N problem zahtjeva testiranje smanjenje broja dimenzija u svrhu poboljšanja rezultata [20].
7) Metrike evaluacije bi se trebale koristiti standardne (iz literature[1,2,5]) kako bi rezultati bili konzistentni.

5.1 Modeli regresije

U ovom radu primjenjuje se model strojnog učenja baziran na pojmu nadziranog učenja gdje se modeli treniraju tako što se u njega unose poznati ulazni i poznati izlazni podatci. Vrsta nadziranog učenja primijenjena u radu je vrste regresije gdje je cilj predviđanja kontinuirana izlazna varijabla (energija). Algoritmi strojnog učenja primijenjeni u radu sadržani su u *scikit-learn* [28] biblioteci za programski jezik *Python* te su kao takvi redom probrani oni koji zadovoljavaju zahtjeve nabrojane na početku poglavlja. Algoritmi sa najboljim i najbržim rezultatima su zadržani za nastavak istraživanja te su imenovani i opisani u nastavku.

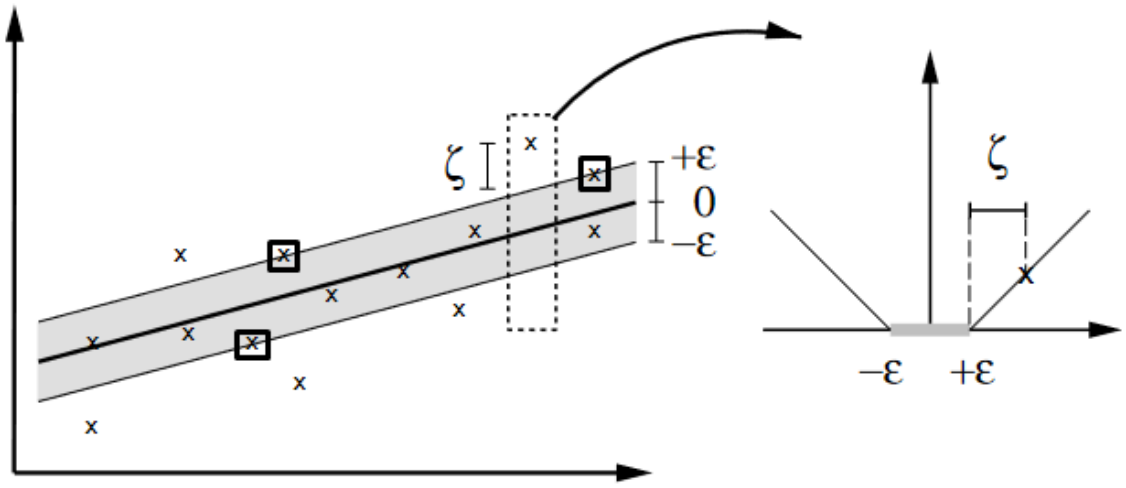
5.1.1 Višeslojni perceptron (engl. *Multilayer perceptron regressor – MLP*)

MLP [28] je vrsta primjene umjetne neuronske mreže koja se koristi za predikciju na temelju ulaznog seta vrijednosti. Model je građen od slojeva međusobno povezanih neurona gdje je svaki sljedeći neuron u sloju povezan sa svim neuronima u sloju prije njega. Na svaki neuron djeluje operacija sumiranja vrijednosti neurona iz prethodnog sloja popraćeno skaliranjem te vrijednosti

težinskim faktorom i na kraju provlačenje vrijednosti kroz aktivacijsku funkciju koja određuje ako će ta vrijednost nastaviti u neuron sljedećeg sloja. Treniranje se vrši iterativno tako da se mijenjaju vrijednosti težinskih faktora dokle god se ne minimizira odstupanje predviđene vrijednosti na izlazu od očekivane vrijednosti [26,28]. Ovakav model ima mogućnost učenja vrlo kompleksnih prijenosnih funkcija i ima veliku fleksibilnost pa je smatran generalnim modelom u primjeni strojnog učenja [26]. Zbog tog razloga odabran je kao početni model za preliminarno istraživanje sa kojim su drugi algoritmi uspoređivani. Snaga *MLP*-a zasjenjena je činjenicom da je to dosta kompleksan model koji sadrži puno hiperparametara i ne skalira se dobro sa brojem podataka što može dovesti do vremenski dugog treniranja ako parametri nisu dobro podešeni. Odabir hiperparametara doduše može se značajno skratiti intuitivnim odabirom polja nad kojim će se vršiti pretraga pa će se o tome više reći u Poglavlju 6.

5.1.2 Metoda potpornih vektora (engl. *Support vector machine - SVR*)

SVR [28,29] je metoda korištenja transformacijske jezgre u regresiji koja djeluje slično kao linearna regresija samo korištenjem transformacije prostora značajki. *SVR* radi tako da pokuša povući hiperravninu sa granicama na udaljenosti $\pm\varepsilon$ tako da sa tim granicama pokrije što je više moguće predikcija u prostoru. Ta hiperravnina postavlja se nasumičnim odabirom predikcija koji se nalaze na rubovima granica koji služe kao pomoćni vektori koji određuju smjer hiperravnine. Na slici 4.1 prikazan je primjer hiperravnine gdje su $\pm\varepsilon$ rubovi granica koji obuhvaćaju predikcije, a ζ predstavlja odmak od granica koji penalizira izračun hipperavnine i utječe na smjer te se ponaša kao član regularizacije. Kvadratom zaokružene predikcije na rubovima granice predstavljaju pomoćne vektore. Ova metoda izabrana je zbog svoje inherentne robusnosti zbog činjenice da ima snažnu regularizaciju i ne koristi sve podatke u isto vrijeme pa tako izvršava svojevrsno probiranje značajki [29].



Slika 5.1 Prikaz hiperravnine u primjeni SVR [29]

5.1.3 Metoda linearne grebenaste regresije (engl. *linear Ridge regression -Ridge*)

Ridge [26,28] regresija je metoda višestruke linearne regresije koja specifično smanjuje utjecaj kolinearnosti ulaznih varijabli na izlaz. Klasična linearna regresija računa koeficijente minimizirajući kvadrat pogreške linearne funkcije od predikcija za svaku nezavisnu varijablu. Naime ako ima više varijabli koje ovise jedna o drugoj koeficijenti pridružene tim varijablama mogu postati prevladavajući velike i time izlaz postane jako osjetljiv na sitne promjene tih varijabli. *Ridge* regresija računa kolinearnost i unosi svojstvo *L2* regularizacije koja vrši penalizaciju koeficijenata kolinearnih varijabli. *L2* regularizacija ne penalizira sve varijable istim vrijednostima nego je proporcionalna iznosu početne vrijednosti što znači da veliki *L2* brže penalizira velike koeficijente a sporije manje koeficijente prilikom minimizacije [26,28].

5.1.4 *TheilSen* metoda regresije

TheilSen [28] regresija je metoda višestruke linearne regresije koristeći statističke metode koje su se pokazale robusne i statistički nepristranih predikcija. Činjenica da je estimator bez pristranosti čini ovu metodu efikasnijom od klasične i čini ju otpornom na netipične vrijednosti i može predviđati čak sa koruptiranim podatcima. To ju također čini otpornom na kolinearnosti među varijablama. Metoda radi tako da izvršava više linearnih predikcija u isto vrijeme na podskupinama ulaznih varijabli te se uzima medijan vrijednosti izračunatih među njima te se rezultat toga koristi za predikciju[28,30].

5.2 Evaluacija modela

Za procjenu kvalitete estimacije metoda strojnog učenja i razumijevanja što rezultati predstavljaju potrebno je odabrati metode kojima će se ti isti rezultati vrednovati. Algoritmi koji se koriste za strojno učenje generalno samo minimiziraju grešku vrijednosti unutarnjih varijabli baziranih na razlici estimacije. To ima posljedicu da čak i kad algoritam konvergira u minimum to ne znači nužno da je estimacija točna i ispravna, a samim time osoba koja analizira točnost estimacije iz tih vrijednosti to ne može procijeniti. Za uvesti metrike potrebno je postaviti što će se promatrati i kako definirati uspješnost estimacije. U literaturi koriste se već definirane metrike te će u radu biti korištene iste zbog konzistentnosti rezultata i usporedbe [1,2,26,28].

5.2.1 Koeficijent determinacije (R^2)

Koeficijent determinacije je standardna statistička metoda provjere dobrote funkcije predikcije estimatora [26,28]. Sa njime se mjeri proporcionalna varijacija dobivenih i očekivanih vrijednosti. Vrijednost R^2 je očekivana u rasponu od [0,1] sa time da 1 znači da model savršeno prati očekivane vrijednosti. Vrijednost koeficijenta može biti negativna što znači da obična horizontalna linearna funkcija bolje predviđa od modela. Formula koeficijenta determinacije prikazana je u sljedećoj jednadžbi gdje je $\hat{y}$ predviđena vrijednost y_i stvarna vrijednost i $\bar{y}$ je srednja vrijednost stvarnih vrijednosti.

$$R^2 = 1 - \frac{\sum_i^n (y_i - \hat{y})^2}{\sum_i^n (y_i - \bar{y})^2} \quad . \tag{5.1}$$

5.2.2 Srednja apsolutna vrijednost (*MAE*)

MAE [26,28] je mjera apsolutne greške predviđenih vrijednosti od realnih tako što se izračuna suma aritmetičkih sredina razlike realne i predviđene vrijednosti. Ovakva metrika je korisna jer ovisi o skali vrijednosti koja se promatra za razliku od R^2 metrike [26]. *MAE* ima istu mjernu jedinicu kao i predviđeni podatak (u ovom slučaju kcal/mol) te praktički ukazuje na toleranciju odstupanja od realne vrijednosti što znači da je cilj smanjiti iznos odnosno rezultat bliže nuli je bolji. Formula za računanje *MAE* iznosa prikazana je formulom

$$MAE = \frac{1}{n} \sum_l^n |y_i - \hat{y}_i| \, . \tag{5.2}$$

5.2.3 Korijen srednje kvadratne pogreške (*RMSE*)

RMSE [26,28] je metrika koja prikazuje mjeru pogreške kao kvadrat sume aritmetičke sredine razlika između predviđene i realne vrijednosti. Koncept je sličan kao *MAE* gdje je razlika što je *RMSE* puno osjetljiviji na velika odstupanja greške. Gdje *MAE* ukazuje na opću preciznost modela *RMSE* ukazuje na magnitudu greške modela. Ova metrika također ovisi o skali promatrane varijable i rezultati koji teže ka 0 su bolji. Formula za računanje *RMSE* je u nastavku

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{n}(y_i - \hat{y})^2}\,. \tag{5.3}$$

5.2.4 Standardne devijacije *R2, MAE i RMSE*

Za ocjenjivanje efikasnosti modela poželjno je uključiti i metrike koji pokazuju razliku rezultata učenja u međukoracima iteracija kako bi se ocijenilo svojstvo generalizacije nad setom podataka. Svojstva generalizacije spadaju pod efekte pre-treniranja i pod-treniranja te razlika u rezultatima predviđanja ovisno o nasumičnom odabiru ulaznih varijabli u setu podataka [26]. Kako bi se analiziralo to ponašanje uvest će se mjera standardne devijacije R^2, *MAE* i *RMSE* metrika koje će opisati fluktuaciju rezultata. Standardna devijacija računa se sljedećim formulama gdje član sa crtom iznad metrike predstavlja srednju vrijednost.

$$R^2_{STD} = \sqrt{\frac{1}{N}\sum_{i}^{N}\left(R_i^2 - \overline{R^2}\right)^2}\,, \tag{5.4}$$

$$MAE_{STD} = \sqrt{\frac{1}{N}\sum_{i}^{N}(MAE_i - \overline{MAE})^2}\,, \tag{5.5}$$

$$RMSE_{STD} = \sqrt{\frac{1}{N}\sum_{i}^{N}(RMSE_i - \overline{RMSE})^2}\,. \tag{5.6}$$

5.3 Treniranje i testiranje modela

Kako bi se primijenile metode i počelo izvršavati učenje estimatora na pripremljenim podatcima potrebno je podijelit podatke na takav način koji će dati najveću mogućnost uspješnog učenja sa najboljom procjenom i evaluacijom tog modela. Podatci se dijele na set za treniranje i set za testiranje (validaciju) i u nastavku će se obraditi načini podjele. U svrhu boljeg treniranja i statistički relevantnijeg testiranja uvodi se pojam unakrsne validacije koji povećava kvalitetu učenja [26,28]. Na kraju uz pomoć unakrsne validacije dobije se mogućnost automatiziranog unaprjeđivanja modela korištenjem pretrage hiperparametara algoritama.

5.3.1 Podijela seta podataka na podsetove za trening i test

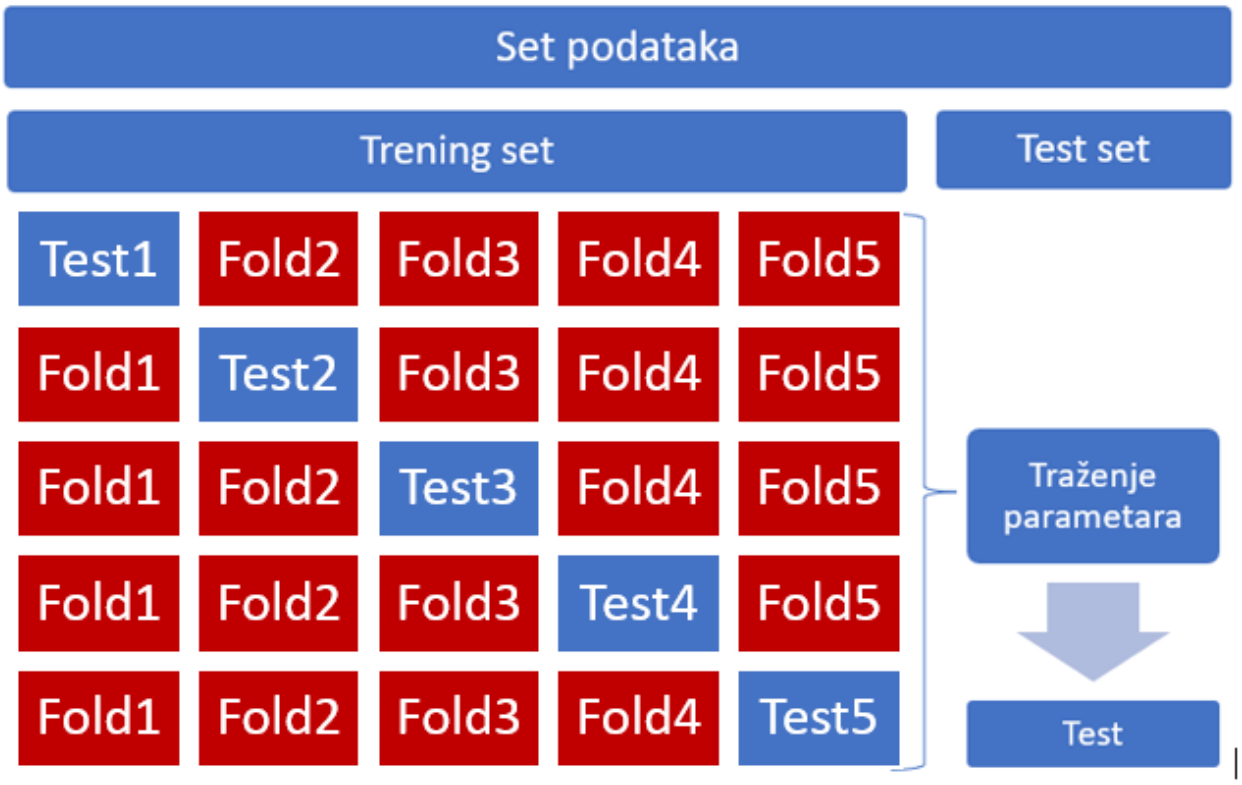
Potreba za podjelom podataka javlja se zbog toga što modeli strojnog učenja usavršavaju svoje estimacije isključivo na podatcima koji su im servirani. U slučaju da se unesu svi podatci iz seta ne može se procijeniti svojstvo generalizacije modela nad podatcima već samo imamo informaciju o estimaciji na tim specifičnim podatcima. Ako se ne može prepoznati svojstvo generalizacije ne može se procijeniti ako je model pretreniran na podatke, odnosno ako model može estimirati točne vrijednosti za drugu kombinaciju ulaznih varijabli. Kako bi se to izbjeglo uvodi se metoda razdvajanja podataka na set za treniranje i testiranje [26,28] gdje model u prvom koraku učita podatke za treniranje, izvrši učenje i tek nakon toga unese podatke iz seta za testiranje kako bi

procijenio svoj stupanj estimacije. Ovime se postiže smanjenje statističke pristranosti modela bez uništavanja varijance podataka. Nema specifičnih pravila za odabir omjera podjele na trening i test pa se u ovom slučaju odabrala podjela 80/20 (trening/test) kako bi se zadržala dovoljna količina podataka za treniranje uzevši u obzir relativno mali broj uzoraka [20,21]. Ujedno je još iskorišteno svojstvo nasumične podjele varijabli ulaza i izlaza na setove kako bi se još više povećala nepristranost i omogućilo kvalitetno učenje modela dobivenih iz informacija umjesto raspoznavanja oblika. Za primjenu podjele na trening i test iskorištene su ugrađene funkcije biblioteke za strojno učenje.

5.3.2 Unakrsna validacija (engl. *cross validation* - CV)

Treniranje modela na velikom setu za treniranje/testiranje može funkcionirati dobro dok se ne dođe do točke gdje se moraju birati hiperparametri regresijskih modela. U tom slučaju probiranjem hiperparametara ručno na ukupnom setu podataka može izazvati iterativno poboljšanje modela jer je testiran na prethodno viđenim podatcima. Usprkos činjenici da model daje bolje rezultate ti rezultati nisu statistički relevantni jer se ponovno može javiti pretreniranje zbog situacije u kojoj nasumičnim odabirom varijabli se mogu pojaviti podatci iz seta za testiranje koji je trebao ostati neviđen. Takva situacija naziva se curenje podataka koji nisu trebali uči u model tokom treniranja [26,28].

Za rješavanje tog problema uvodi se metoda unakrsne validacije koja trajno odvoji testni set podataka, a trening set raspodjeli na *n* količinu manjih podskupina te se oni iterativno izmjenjuju za treniranje i testiranje te se taj postupak ponavlja u *k* koraka. Nakon što se izvrši *n* brojeva raspodjela u *k* koraka model se evaluira odabranim metrikama. Takva metoda zove se *k-fold CV* [28] i shema metode prikazana je na Slici 5.2. Odrađivanje unakrsne validacije korisno je za procjenu generalizacije modela nad podatcima, optimizaciju hiperparametara algoritama i u slučajevima setova podataka koji imaju relativno malo uzoraka jer se time virtualno povećava broj uzoraka. Negativna strana unakrsne validacije je činjenica da se treniranje mora izvršiti *k* puta pa to treba uzeti u obzir kako bi se spriječilo dugoročno treniranje kada se pretražuju hiperparametri.



Slika 5.2 Shema k-Fold unakrsne validacije

5.4 Ansambl metode

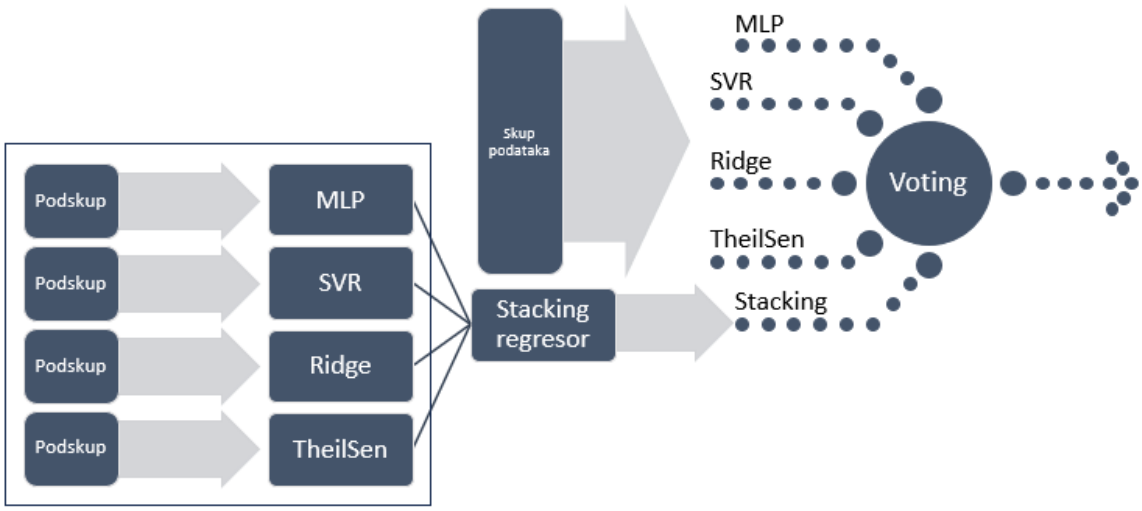
Učenje ansambl metodama je proces koji koristi setove drugih modela strojnog učenja istreniranih na istim podatcima u cilju estimiranja istog rezultata. Takav set modela se integrira na neki način kako bi se unaprijedila moć estimacije zajedničkog zadatka. Za izvršiti ansambl generalno se prvo počinje sa treniranjem nekoliko različitih estimatora na istom setu podataka sa istim izlaznim varijablama. Nakon što se pripremi veći broj estimatora njih se onda spaja u odabranu metodu ansambla, taj ansambl se isto tako trenira nakon čega se rezultati evaluiraju i ako je potrebno ukloni jedan ili više ulaznih estimatora koji narušavaju rezultat pa se treniranje ansambla izvršava ponovno [26,28,31,32].

5.4.1 Ansambl slaganjem regresora (engl. *Stacking regressor*)

Slaganje regresora [28,31] je metoda ansambla gdje se pripremi set baznih estimatora čiji rezultati se koriste za učenje izlaznog meta-estimatora. Meta-estimator je praktički izučen na vrijednostima nakon transformacije što je ekvivalentno učenju modela na novom prostoru značajki. Zadnja predikcija vrši se odabranim krajnjim regresorom. Ideja je spojiti najbolje značajke pojedinih modela kako bi se poboljšao rezultat.

5.4.2 Glasački ansambl regresora (engl *Voting ensemble*)

Korištenje glasačkog ansambla [28,31] se razlikuje od slaganja prvenstveno što se u glasačkom ansamblu svi bazni estimatori treniraju na cijelom setu podataka. Nakon treniranja na rezultate se glasa te se na taj način predikcije usrednjavaju. Ovakav model može imati bolje predikcije, ali glavno svojstvo je smanjenje utjecaja pretreniranja na baznim estimatorima. U ovom radu trenirat će se obje vrste regresora (*voting* i *stacking*) s time da će se uspoređivat rezultati jednog i drugog ansambla s time da će se na kraju *stacking* ansambl biti jedan ulaza u glasački. Shematski prikaz izgleda takvog ansambla je na Slici 5.3.



Slika 5.3 Diagram serijske ansambl metode

6. PRETRAŽIVANJE HIPERPARAMETARA

Kod nadziranog učenja pridodaje se velika važnost odabiru hiperparametara korištenim u algoritmima strojnog učenja i načinu pretraživanja istih. Optimalni set hiperparametara utjecat će na preciznost rezultata i brzinu konvergencije i uopće šansi da se algoritma izuči na podatcima[26,28,33]. Hiperparametri za razliku od unutarnjih koeficijenata modela se ne ažuriraju automatski prilikom učenja modela nego se definiraju kao globalna varijable prije početka treniranja te se moraju mijenjati prije pokretanja modela. U ovom poglavlju prikazat će se hiperparametri koji su u prethodnom poglavlju imenovani i opisat način na koji su pretraživani.

Postoji više metoda pretraživanja optimalnog seta hiperparametara koji daje najbolje rezultate. Početni korak je definirati prostor značajki koji se sastoji od odabranih hiperparametara preko kojeg se radi pretraživanje. Nakon toga definira se metoda pretraživanja od kojih su najčešće nasumično pretraživanje i pretraživanje po mreži. U ovom radu koriste se ugrađene funkcije programa za izvršiti pretraživanje i koristilo se za pronalaženje optimalnog broja komponenti u *PCA*. Zbog prirode problema koristili su se robusni algoritmi kod kojih većinu hiperparametara se ili nije trebalo mijenjat ili su ih imali malo za namještanje.

6.1 Hiperparametri odabranih modela

U nastavku će biti prikazani hiperparametri modela koji su odabrani za korištenje u ovom radu. Određeni modeli imaju automatski odabir pojedinačnih parametara ovisno o obliku matrice ulaznih podataka. Neki od drugih parametara nemaju direktni utjecaj na performanse modela pa će se nakon prikaza hiperparametara biti zadan popis parametara koji će se optimizirati te će time biti definiran prostor pretrage.

6.1.1 Hiperparametri *Ridge* regresije

U Tablici 5.1 je popis parametara [28] *Ridge* regresije gdje su definirane gornja i donja granica parametara koji će se optimizirati. Ova metoda regresije u biblioteci ima verziju sa ugrađenom pretragom parametara preko unakrsne validacije *RidgeCV* tako što se definira raspon *alpha* parametra. Obzirom da drugi parametar koji ima smisla mijenjat (*solver*) ima automatski odabir ovisno o dimenzijama ulazne matrice potrebno je samo varirati *alpha* parametar. Zbog toga ovaj model se u kodu poziva samostalno preko *RidgeCV* i unošenjem polja [1e-5…1e+2].

Tablica 5.1 Hiperparametri modela Ridge regresije

Hiperparametar	**Vrijednost**	**Donja granica**	**Gornja granica**
alpha	default = 1.0	0.00001	100
fit_intercept	default = *True*	/	
copy_X	default = *True*	/	
max_iter	default = None	/	
tol	default = 1e-4	/	
solver	default = *'auto'*	*'svd', 'cholesky', 'sparse', 'lsqr', 'sada', 'lbfgs'*	
positive	default = *False*	/	

6.1.2 Hiperparametri *SVR*

U tablici 5.2 prikazani su hiperparametri [28] *SVR* modela regresije gdje stupci gornja i donja granica predstavljaju granice polja pretraživanja, a stupac vrijednost predstavlja baznu vrijednost te ako u stupcu desno od nje je kosa crta to znači da nije mijenjana. Parametri od interesa su *C* koji predstavlja jačinu regularizacije, *epsilon* koji određuje granicu tolerancije, *kernel* koji određuje vrstu transformacije, *gamma* koji određuje hyperparametar unutar *kernel* parametra. Korištenje osnovnog kernela *rbf (radial basis function)* je preporučeno i pokazalo se kao najbolji izbor jer sa drugim kernelima algoritam nebi konvergirao. *Gamma* kao parametar se može definirati da se automatski računa u dvije metode (*'auto'* i *'scale'*) ili se unosi konstantni broj. Odlučeno je ostaviti da se računa automatski sa metodom pod *'auto'* jer osnovna definicija *'scale'* računa varijancu svakog člana te to suviše produžuje vrijeme treniranja u slučaju velikog broja značajki.

Tablica 5.2 Hiperparametri SVR regresora

Hiperparametar	**Vrijednost**	**Donja granica**	**Gornja granica**
kernel	default = *'rbf'*	*'linear', 'poly', 'rbf', 'sigmoid'*	
degree	default = 3	/	
gamma	default = *'scale*	*'scale', 'auto'*	
coef0	default = 0.0	/	
tol	default = 1e-3	/	
C	default = 1.0	0.001	10000
epsilon	default = 0.1	0.00001	10
shrinking	default = *True*	/	
cache_size	default = 200	1000	
max_iter	default = -1	/	

6.1.3 Hiperparametri *TheilSen* regresora

U tablici 5.3 prikazani su hiperparametri [28] *TheilSen* modela regresije gdje stupci gornja i donja granica predstavljaju granice polja pretraživanja, a stupac vrijednost predstavlja baznu vrijednost te ako u stupcu desno od nje je kosa crta to znači da nije mijenjana. Parametri od interesa su *max_subpopulation* koji određuje broj podsetova na kojima će se model izvršavati i *n_subsamples* koji određuju broj značajki od kojih će se računati unutarnji parametri. Doduše hiperparametre *TheilSen* regresora nije specifično potrebno mijenjati jer se samostalno prilagođavaju podatcima osim u specifičnim slučajevima gdje se želi smanjiti populacija u svrhu bržeg treniranja.

Tablica 5.3 Hiperparametri TheilSen regresora

Hiperparametar	**Vrijednost**	**Donja granica**	**Gornja granica**
Fit_intercept	default = *True*	/	
Copy_X	default = *True*	/	
Max_subpopulation	default = 1e4	10	1000
N_subsamples	default = None	Broj značajki	
Max_iter	default = 300	/	
tol	default = 1e-3	/	
N_jobs	default = None	N = -1	

6.1.4 Hiperparametri MLP regresora

U Tablici 5.4 prikazani su hiperparametri [28] *MLP* regresora gdje stupci gornja i donja granica predstavljaju granice polja pretraživanja, a stupac vrijednost predstavlja baznu vrijednost te ako u stupcu desno od nje je kosa crta to znači da nije mijenjana. Od izabranih modela *MLP* ima najviše parametara za optimizaciju i može imati veliku osjetljivost na svaku od njih. Većina parametara doduše su uvjetovani parametrom '*solver'* koji predstavlja algoritam korišten za minimizaciju greške. Sa nekoliko pretpostavki i savjeta iz uputa za korištenje polje pretraživanja može se znatno smanjiti kako bi se ubrzao proces sprječavanjem nepotrebnih ili neuspješnih treniranja. Za setove podataka sa malim brojem uzoraka (<10000) kao što je set koji se koristi u ovom radu preporučeno je koristiti *solver: lbfgs* [28] od kojeg se i krenulo pa je broj parametara za pretraživanje drastično pao. Parametri od interesa su

- *hidden_layer_sizes* koji određuje broj neurona u *n*-tom sakrivenom sloju gdje je *n* broj slojeva odnosno količina brojeva odvojenih zarezom (npr. 10,10,10 je tri sloja po 10 neurona),

- *activation* je vrsta aktivacijske funkcije na krajevima neurona ,
- *solver* je optimizacijski algoritam težinskih koeficijenata (odabran *lbfgs*),
- *alpha* stupanj regularizacije podataka

Tablica 5.4 Hiperparametri MLP regresora

Hiperparametar	**Vrijednost**	**Donja vrijednost**	**Gornja vrijednost**
hidden_layer_sizes	default = (N_1,…,N_i)	1	10
N_i *(neuroni)*	default = 100	10	200
activation	default = *relu*	*'relu', 'identity', 'tanh', 'logistic'*	
solver	default = *adam*	*lbfgs*	
alpha	default = 1e-4	0.000001	1
max_iter	default = 200	/	
tol	default = 1e-4	/	
max_fun	default = 15000	/	

6.2 Metode pretraživanja

Za izvršiti optimizaciju hiperparametara potrebna je napraviti metodu sadržanu od sljedećih koraka[26,28]:

1) Odabrat estimator,
2) definirati prostor pretraživanja hiperparametara,
3) odabrati metodu kojom se pretražuju hiperparametri,
4) izvršiti odabir parametara i unesti ih u estimator,
5) izvršiti unakrsnu validaciju na novim parametrima,
6) evaluirati rezultate te ako nije završena pretraga vratiti se na korak 4).

6.2.1 Metoda nasumičnog odabira (engl. *Random search*)

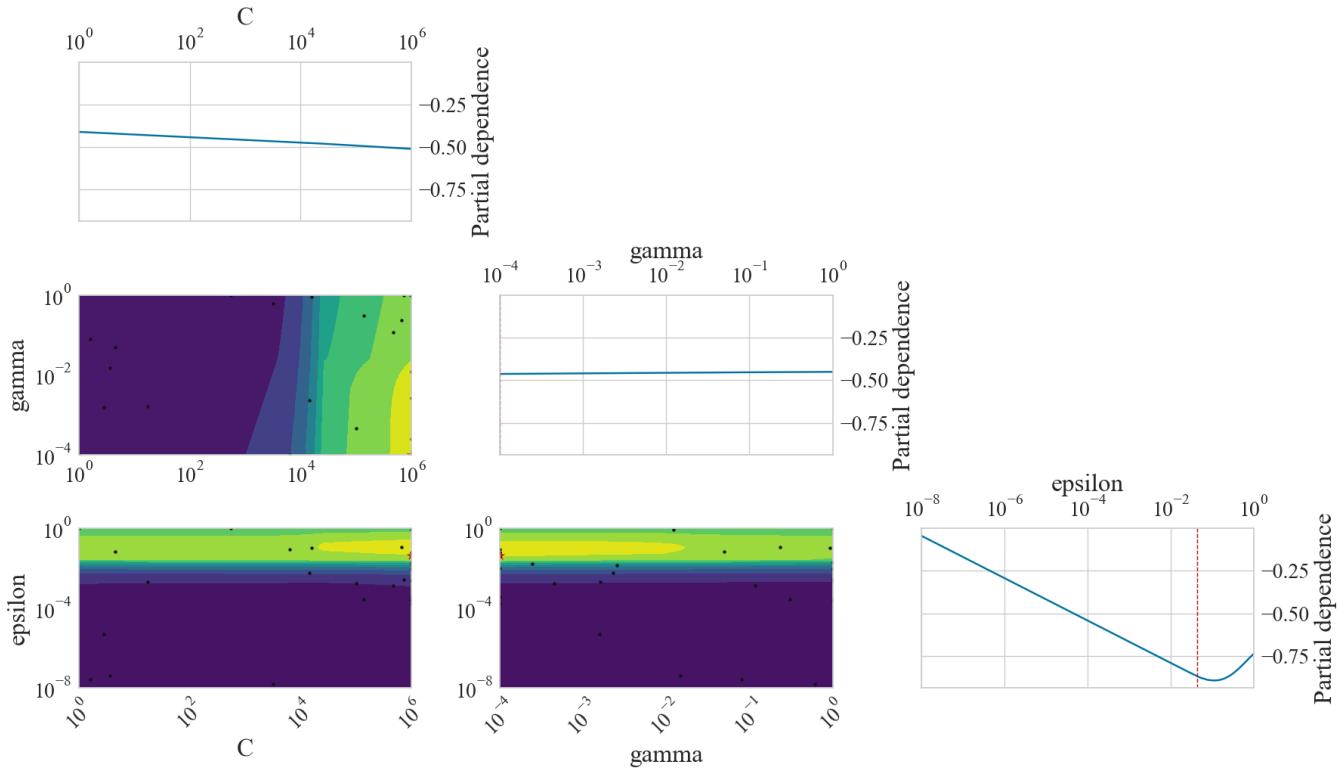
Pretraživanje parametara nasumičnim odabirom [28] daje potencijalno brže i efikasnije pretraživanje od pretraživanja kroz mrežu. Način na koji funkcionira je da se definira raspon vrijednosti određene distribucije iz koje će algoritam nasumično birati vrijednosti. Ovako se osigura bolja raspodjela tako što se statistički pojedini hiperparametri češće pojavljuju u točkama minimuma pa se pretraga vrši efikasnije. To je poželjna karakteristika u početcima pretraživanja kada se nema ideja o mjerama hiperparametara pa se vrlo brzo mogu odbaciti veliki dijelovi prostora i ograničiti prostor pretraživanja. U svrhu ovog rada odrađeno je početno pretraživanje korištenjem ove metode dok se nije dobila ideja o ponašanju modela.

6.2.2 Metoda pretraživanja mreže (engl. *Grid search*)

Pretraživanje mreže [28] smatra se metodom temeljite potrage gdje se pretraživanje izvršava za sve moguće kombinacije odabranih parametara kroz cijelu mrežu definiranih vrijednosti. Najčešće se koristi ako specifično treba pretražiti skupinu parametara, ili se već unaprijed zna koji raspon vrijednosti se traži. Posljedica ove metode je što se svaka pojedina vrijednost pretraži sa svakom drugom pojedinom vrijednošću te se jako loše vremenski skalira sa brojem parametara. Usprkos tome u slučaju potrebe za preciznim odabirom parametara metoda može dati bolje rezultate i lako ih je protumačiti.

6.2.3 Metoda pretraživanja na temelju Bayes-ovog algoritma

Pretraživanje metodom Bayesovog algoritma [33] je dobra optimizacijska metoda za slučajeve gdje se ne poznaje prostor pretraživanja i za sustave s puno šuma. Ova metoda djeluje tako da se gradi probabilistički model koji gradi funkciju baziranu na prethodno odabranim parametrima uspoređenim sa dobivenim vrijednostima. Metoda kao takva je svojevrsni optimizacijski model koji minimizira funkciju dok ne dođe u točku minimuma za sve dimenzije prostora. Ovaj model pokušava uskladiti eksploraciju sa eksploatacijom odnosno pokušava naći vrijednosti hiperparametara u točki u kojoj nije bio a da pritom nađe minimum. Ova metoda u radu je primijenjena preko biblioteke *scikit-optimize* te radi tako da se definira estimator i parametri definirani distribucijom vrijednosti u prostoru te se izvrši optimizacija korištenjem unakrsne validacije. Primjer grafa djelomične zavisnosti hiperparametara za *SVR* treniranog na reduciranom setu podataka prikazana je na slici 5.1. Grafovi na dijagonali pokazuju krivulje vrijednosti hiperparametara kroz iteracije. Unutarnji osjenčani grafovi prikazuju prostor u obliku gradijenta na kojem su ucrtane točke koje predstavljaju pomicanje po prostoru sa zvijezdom koja je finalna točka u kojoj je algoritam konvergirao. Algoritam ne konvergira uvijek u istoj točki i ima mogućnost odabira koja nije idealna, ali zbog dobre vizualizacije lagano je ručno usavršit pretragu



Slika 6.1 Graf djelomične zavisnosti za pretraživanje hiperparametara SVR modela

6.2.3 Pretraživanje broja komponenti u *PCA*

U ovom radu odradilo se pretraživanje parametara broja komponenti u *PCA* koje je izvršeno u preliminarnom dijelu istraživanja kako bi se što prije odradilo smanjenje dimenzija seta podataka jer bi inače traženje parametara trajalo predugo. Također primijećena je bila jako velika oscilacija u rezultatima za malu promjenu parametara zbog suviše velikog broja značajki. Tek nakon što su se modeli generalno izučili je ponovno odrađeno pretraživanje parametara *PCA* kako bi se probalo dobiti efikasniji sustav za unos u ansambl metode.

7. REZULTATI

U nastavku će biti prikazan proces istraživanja te rezultati dobiveni tokom ispitivanja i treniranja modela dok se pokušavalo dobiti što bolja estimacija energije iz seta podataka. Od početka odrađeno je preliminarno istraživanje na punom setu podataka, testiranje metoda redukcije dimenzija i smanjeni set nakon korelacije. Utvrđeni su modeli regresije koji su testirani na punom setu nakon čega su se složili u cijevovod (engl. *pipeline)* gdje se odradilo pretraživanje broja komponenti za *PCA* i skaliranje. Nakon toga odrađeno je pretraživanje hiperparametara Bayesovom metodom pretrage te su modeli optimizirani. Nakon optimizacije odrađen je ansambl na modelima pojedinačni i serijski te su uspoređeni rezultati. Treniranje se izvršavalo na osobnom računalu sa procesorom AMD Ryzen 5 2600 3.4 GHz sa 6 fizičkih (12 logičkih) jezgri, 32GB RAM, m.2 SSD-om i grafickom karticom nVidia GTX1050 koja nije bila korištena.

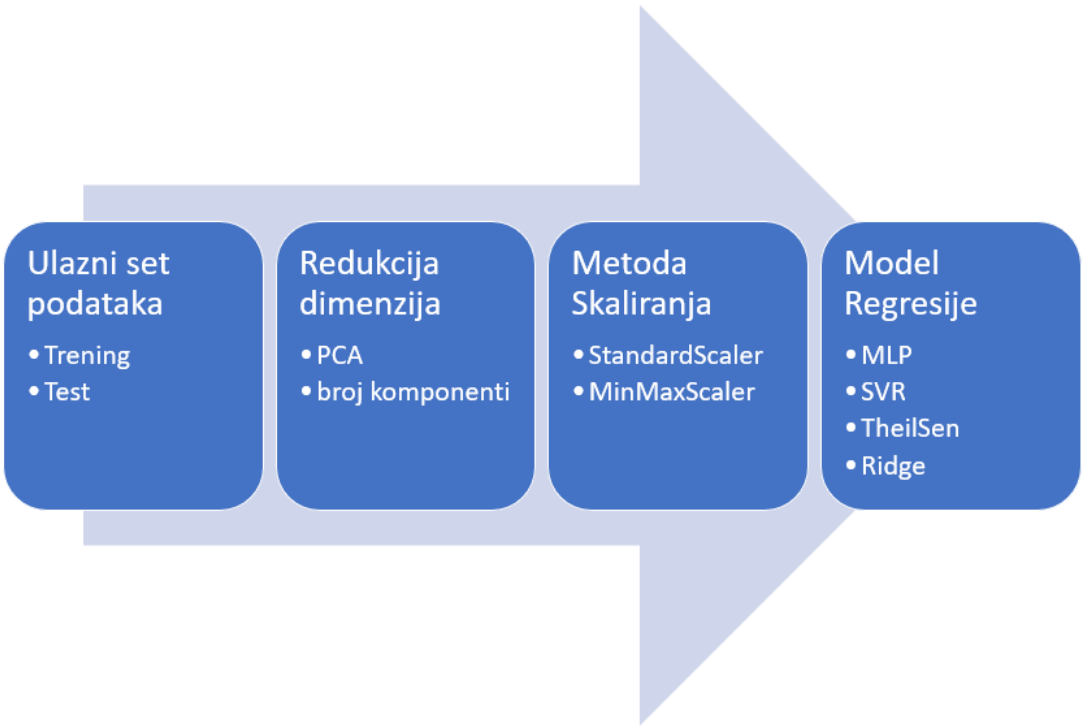
7.1 Početno istraživanje

Od programskog sučelja odabran je programski jezik *Python* u *Spyder* programskoj okolini, a glavna biblioteka koja sadrži većinu funkcija i algoritama za strojno učenje i njegovu implmentaciju je *scikit-lean.* Zbog generalnog korištenja u literaturi za početak istraživanja odabran je *MLP* kao bazni algoritam na kojem će se testirati set podataka. Set podataka učitan je preko *Numpy* biblioteke nakon čega je analiziran i odrađena je početna obrada izbacivanjem nul-vrijednosti (7351 značajka) te se set podataka podijelio na trening i test. *MLP* regresor je učitan sa postavkom *'lbfgs'* u *solveru* i *'identity'* u aktivacijskoj funkciji dok je ostalo ostavljeno u tvorničkim postavkama te je pokrenuto treniranje.

Treniranje MLP na cijelom setu trajalo je preko 12 sati, ali je dalo relativno dobre rezultate za prvi put te su postavljeni drugi modeli regresije kako bi se istestiralo njihovo ponašanje na set podataka. Rezultati skupine modela prikazani su u tablici 7.1 u kojoj su prikazani rezultati i vrsta skaliranja koja je korištena na setu podataka. Ostali modeli imali su značajno brže treniranje od *MLP* iako su svejedno trajali od pola sata do par sati osim *Ridge* koji je trajao par minuta Za nastavak istraživanja zadržani su modeli *MLP, Ridge, TheilSen* i SVR koji su za lakšu primjenu i održavanje konzistentnosti rezultata spojeni u *pipeline* čiji je dijagram toka prikazan na Slici 7.1. U nastavku istraživanja kada god se spominje neki model regresije to se odnosi na cijeli *pipeline* za taj model regresije.

Tablica 7.1. Rezultati preliminarnog treniranja na setu podataka

Model	***R2***	***MAE***	***RMSE***	***Skaliranje***
MLP()	0.815640	1.980164	3.45142	*Standard*
Ridge()	0.905954	0.508327	0.68612	*/*
TheilSen()	0.735281	0.655429	0.92236	*Standard*
Lasso()	0.608712	2.354820	3.30837	*Standard*
ElasticNet()	0.695162	1.647475	2.45113	*Standard*
SVR()	0.751204	1.070848	1.70480	*MinMax*
KNeighbours()	0.543362	2.469141	3.73725	*Standard*
DecisionTree()	0.451124	2.981221	3.87916	*Standard*
GaussianProcess()	0.321431	7.89521	9.04862	*Standard*



Slika 7.1 Prikaz diagrama toka grupiranja modela pipeline metodom

7.2 Redukcija dimenzija i pretraživanje parametara *PCA*

Sljedeće odradilo se pretraživanje mreže korištenjem *CV* metode kako bi se pronašle vrijednosti broja glavnih komponenti koji daju najbolji rezultat kada se izvrši redukcija dimenzija *PCA* metodom. Nakon toga odrađuje se smanjenje dimenzija biranjem značajki korelacijskom analizom te se vrši treniranje modela na smanjenom setu podataka uslijed korelacije za usporedbu sa rezultatima iz tablice 7.1. Na tako smanjenom setu izvršena je *PCA* transformacija te je ponovno korištenjem pretraživanja mreže tražen optimalni broj komponenti. Najbolji rezultati tih koraka su prikazani u tablicama 7.2-7.4. sa odgovarajućom vrijednošću *n_components* za svaki regresor, a na slikama 7.2-7.3 je prikazana promjena rezultata ovisno o broju komponenti za redukciju cijelog seta i umanjenog seta. Raspon *n_components* je bio [50,100,200,250,300,400]

Tablica 7.2 Najbolji rezultati za redukciju dimenzija PCA po n_components

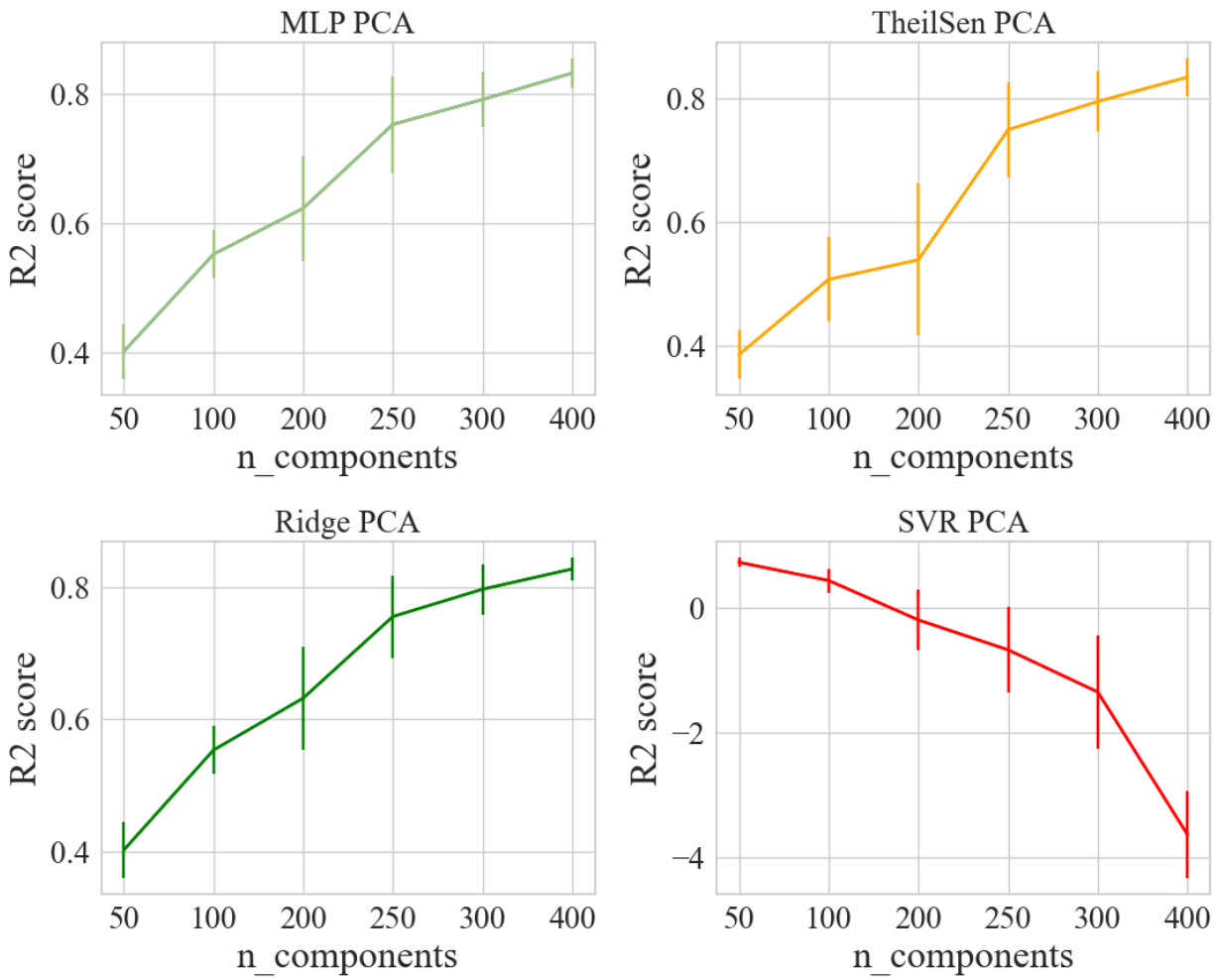
Regresija	***R^2_{test}(std)***	***MAE(std)***	***RMSE(std)***	***n_components***
MLP	0.8325925 (0.0227)	0.7004746 (0.0191)	0.9377927 (0.0493)	400
TheilSen	0.8347448 (0.0303)	0.6896042 (0.0245)	0.9295369 (0.0691)	400
Ridge	0.8273032 (0.0173)	0.6985578 (0.0239)	0.9544234 (0.0471)	400
SVR	0.7372399 (0.0779)	0.8470067 (0.0436)	1.15776911 (0.1319)	50

Tablica 7.3 Rezultati treniranja za redukciju dimenzija korelacijom (CV)

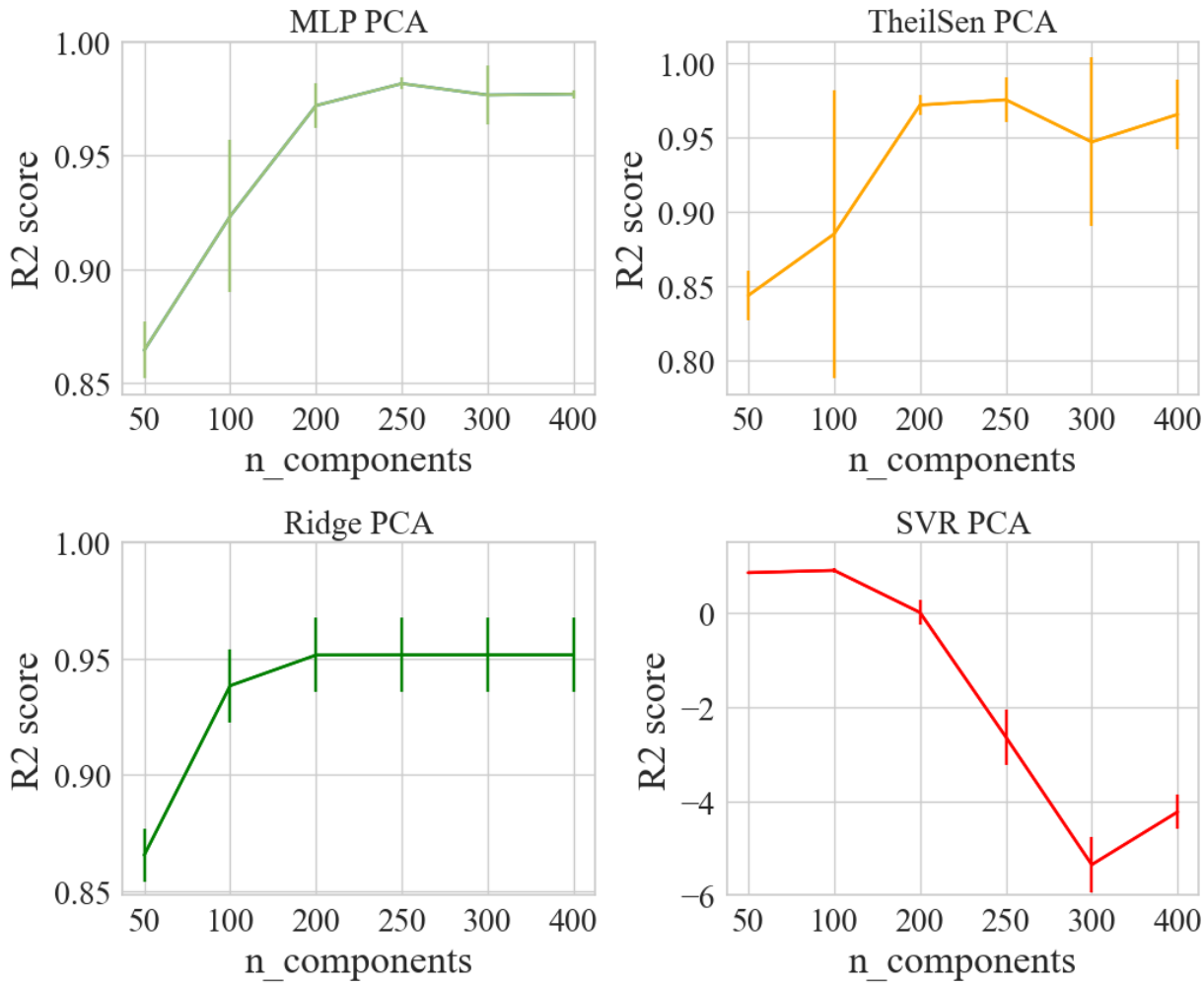
Regresija	***R^2_{test}(std)***	***MAE(std)***	***RMSE(std)***	***skaliranje***
MLP	0.9637961(0.0114)	0.2528598(0.0115)	0.4318440(0.0759)	Standard
TheilSen	0.8946660(0.0875)	0.3515224(0.0368)	0.6859668(0.2346)	Standard
Ridge	0.9626121(0.0091)	0.3195799(0.0087)	0.4393927(0.0324)	/
SVR	0.8170117(0.0218)	0.7545262(0.0532)	0.9807585(0.0666)	Standard

Tablica 7.4 Najbolji rezultati za korelaciju i redukciju dimenzija PCA po n_components

Regresija	***R^2_{test}(std)***	***MAE(std)***	***RMSE(std)***	***n_components***
MLP	0.9815256 (0.0026)	0.1980348 (0.0090)	0.3129415 (0.0353)	250
TheilSen	0.9754343 (0.0151)	0.1911846 (0.0137)	0.3408694 (0.0919)	250
Ridge	0.9515338 (0.0160)	0.3469368 (0.0148)	0.4978195 (0.0652)	400
SVR	0.9053805 (0.0507)	0.3925446 (0.0196)	0.6795929 (0.1607)	100



Slika 7.2 Grafički prikaz rezultata ovisno o broju komponenti u PCA na velikom setu



Slika 7.3 Grafički prikaz rezultata ovisno o broju komponenti u PCA za umanjeni set

U prethodnim tablicama uočljivo je poboljšanje rezultata estimatora za svaki korak koji se odradio. Rezultati od *PCA* na punom setu (Tablica 7.2) su bolji od treniranja na setu bez obrade (Tablica 7.1) . Odrađeno treniranje unakrsnom validacijom na setu koji je smanjen korelacijom dao je još bolje rezultate od prethodna dva, a kada se na to još dodao *PCA* dobiveni su rezultati koji su svi bili bolji od 0.9.

7.3 Optimizacija hiperparametara

Nakon što se obradio set podataka, modeli su se testirali dobili su se zadovoljavajući rezultati. Sljedeći korak bio je odradit optimizaciju hiperparametara kako bi se pokušalo poboljšat modele. U prethodnom poglavlju (Poglavlje 6) navedeni su hiperparametri od interesa pa se na temelju njih napravilo polje pretraživanja koje je iskorišteno za optimizaciju *BayesSearchCV* funkcijom. Modeli koji su optimizirani sa tom funkcijom su samo *SVR* i *MLP* jer *TheilSen* ne zahtjeva optimizaciju, a *Ridge* ima ugrađeni *CV*. Rezultati optimizacije, odnosno optimizirani parametri prikazani su u Tablici 7.5. Dobiveni parametri uvršteni su u pripadajuće modele i pokrenuto je učenje modela korištenjem *CV* sa rezultatima prikazanim u tablici 7.6. te se vidi ponovno

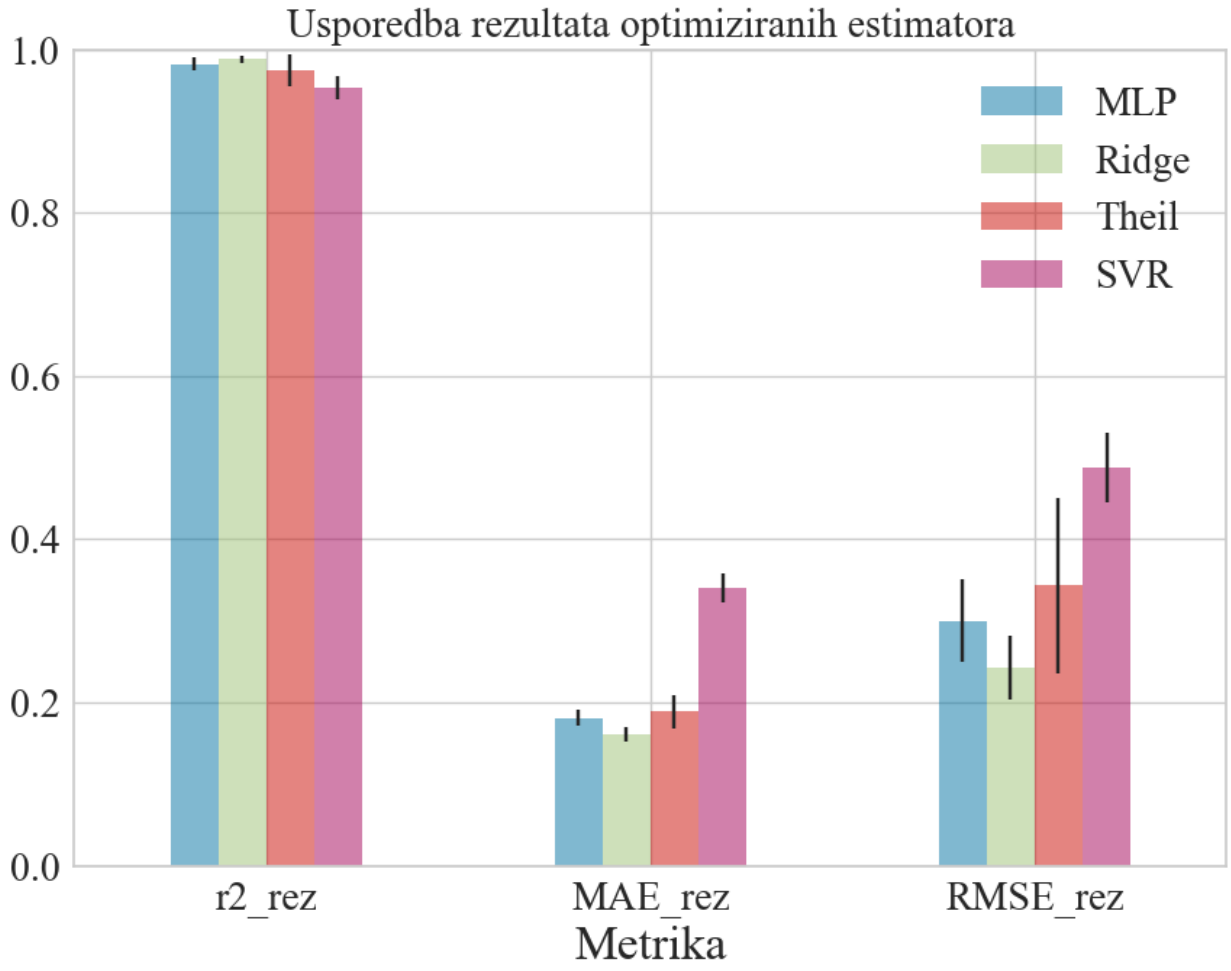
unapređenje rezultata sa grafom na Slici 7.4 gdje se vide rezultati pojedinačnih estimatora sa oznakom standardne devijacije rezultata.

Tablica 7.5 Parametri optimiziranih modela

SVR				
Parametar	**Raspon**	**Skala**	**Rezultat**	**R_2**
C	1 - 1e6	Log	698.1814745879186	0.9534070552298388
Epsilon	1e-8 - 1	Log	0.03342728611844894	
Gamma	1e-4 - 1	Log	0.04904559995067583	
MLP				
Neuroni1	0 - 100	Int	86	0.9826349534201644
Neuroni2	0 - 100	Int	14	
Neuroni3	0 - 100	Int	20	
alpha	1e-8 - 10	log	0.15053789557715627	

Tablica 7.6 Rezultati sa optimiziranim modelima sa CV

Regresija	***R^2_{test}(std)***	***MAE(std)***	***RMSE(std)***	***n_components***
MLP	0.9819710(0.0079)	0.1812598(0.0098)	0.2998901(0.0507)	250
TheilSen	0.9742089(0.0192)	0.1885655(0.0211)	0.3432104(0.1069)	250
Ridge	0.9880693(0.0052)	0.1606839(0.0089)	0.2432530(0.0389)	400
SVR	0.9534957(0.0137)	0.3400988(0.0174)	0.4875581(0.0423)	100



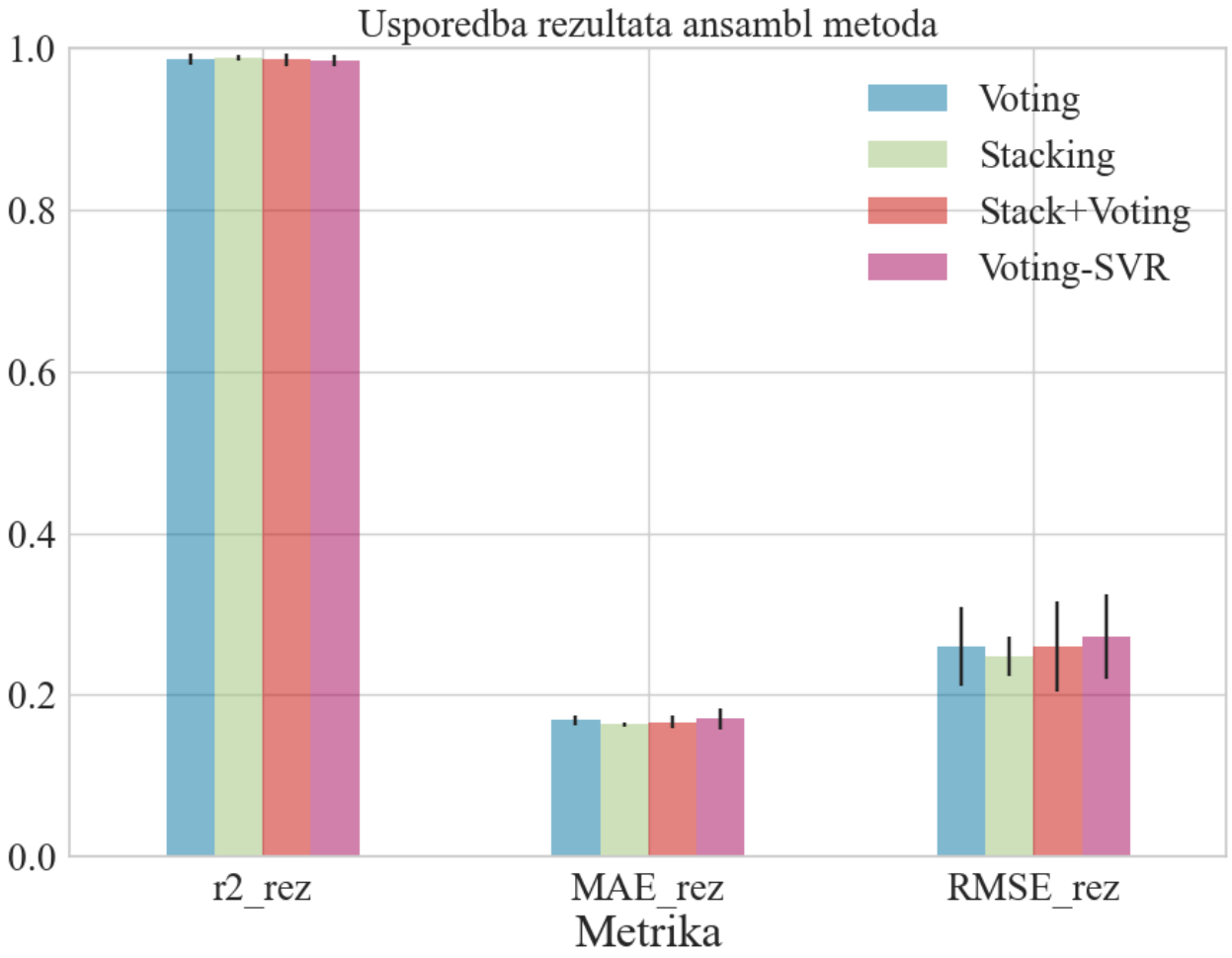
Slika 7.4 Usporedba rezultata optimiziranih estimatora sa oznako standardne devijacije

7.4 Rezultati ansambl

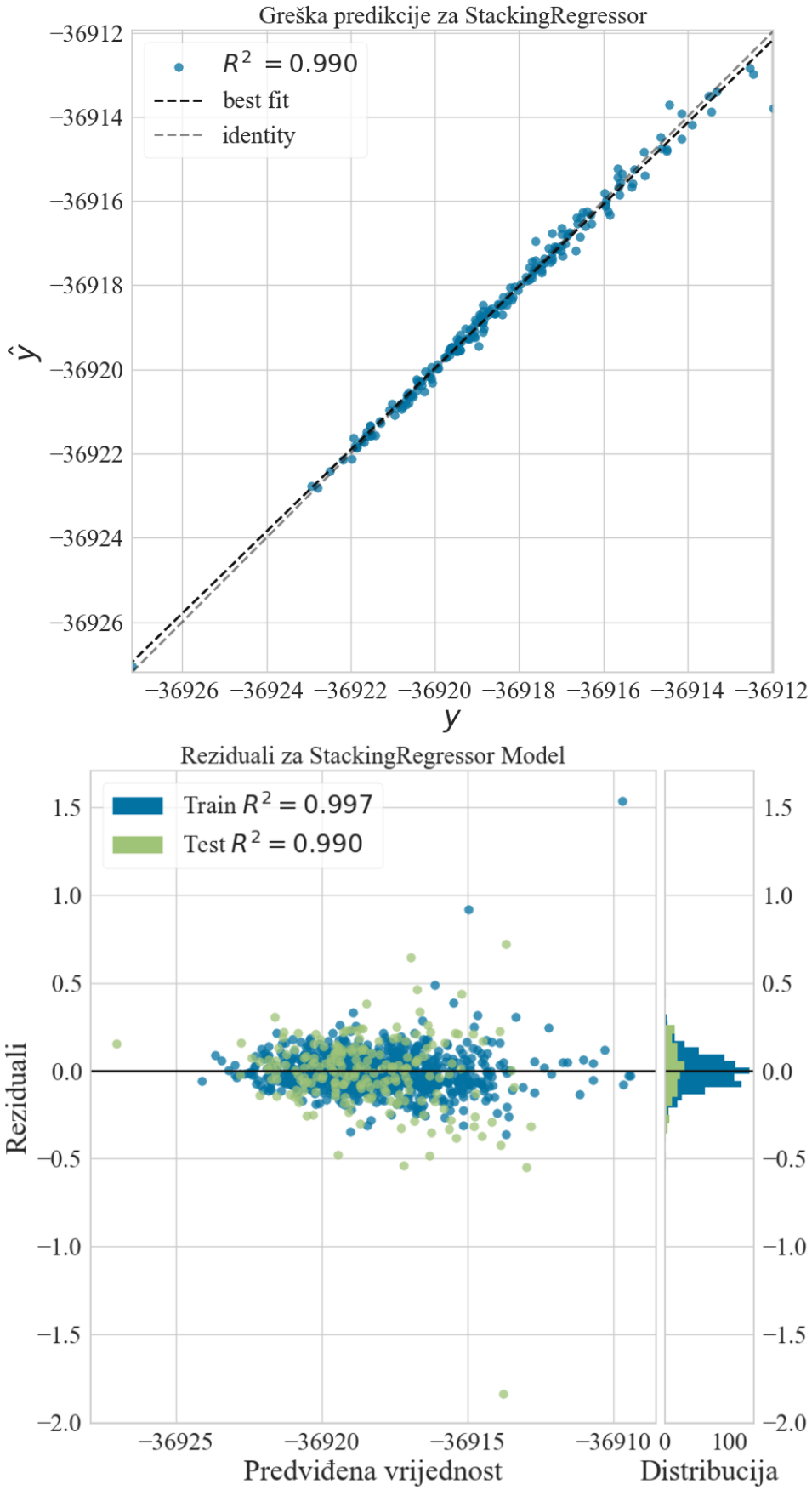
Modeli su se optimizirali te je zadnji korak bio izvršiti ansambl u cilju provjere mogu li se rezultati poboljšati spajanjem modela u cjelinu. Odrađeno je treniranje ansambla modela u *voting*, modela u *stacking*, modeli i *stacking* u *voting*, te modeli u *voting* izbacivanjem *SVR* uzevši u obzir da je prosječno imao najslabije rezultate i najveću devijaciju u odnosu na druge modele. Rezultati su prikazani u tablici 7.7 te se iz rezultata može očitati da *stacking* ansambl daje najbolji rezultat sa najmanjom standardnom devijacijom. Prikaz tablice 7.7 u grafičkom obliku može se vidjeti na Slici 7.5 sa oznakama standardne devijacije. Na slici 7.6 prikazani su grafovi predikcijske greške i graf reziduala koji prikazuju kvalitetu estimacije te se vidi dobro prijanjanje oko linije predikcije i samo dvije estimacije izvan krivulje standardne raspodjele. Na slici 7.7 prikazana je estimacija stacking ansambla na setu podataka koji je bio odvojen te je neviđen za model sa *MAE* = 0.15637.

Tablica 7.7 Rezultati za ansambl modela sa CV

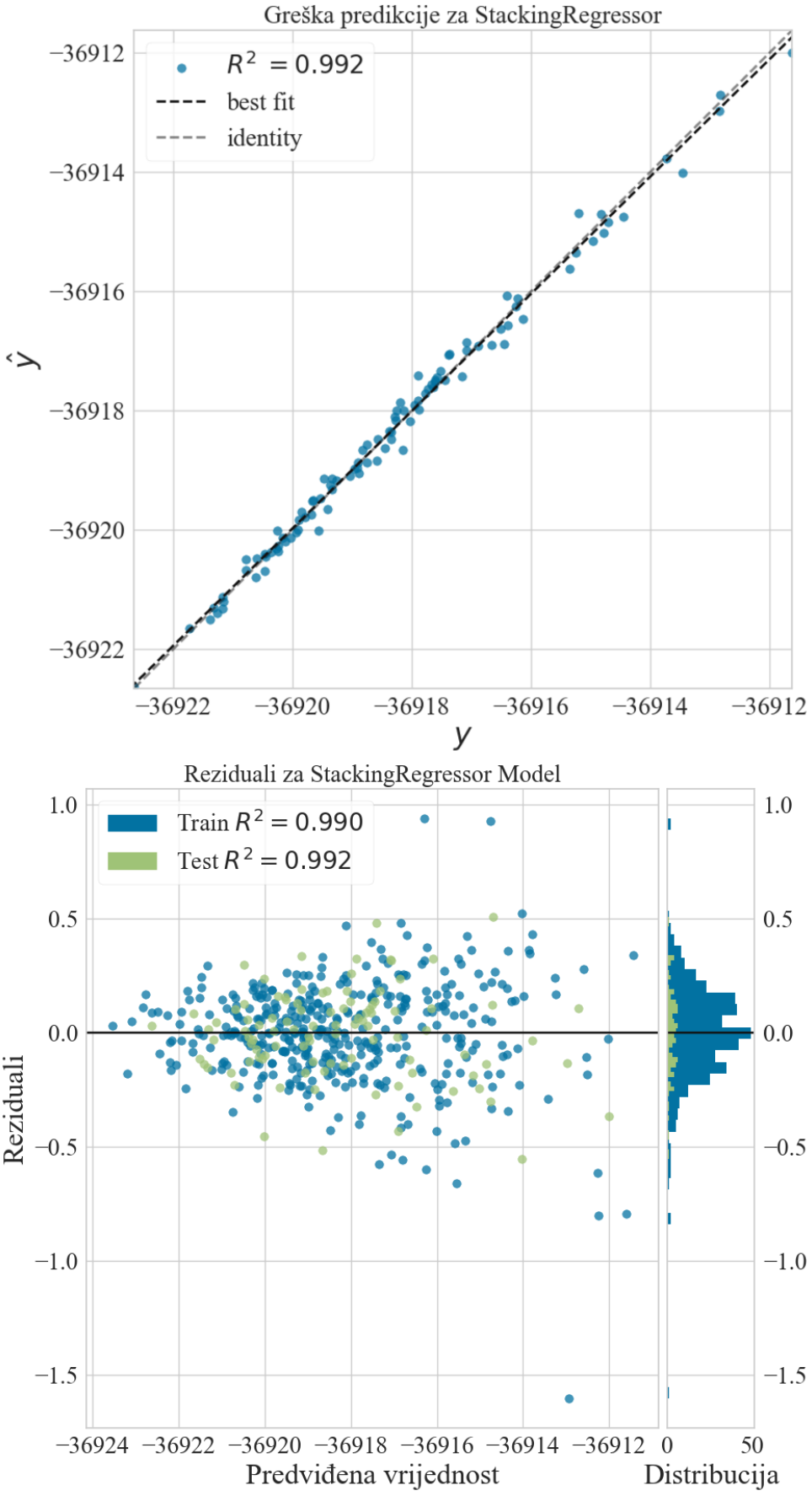
Regresija	***R^2_{test}(std)***	***MAE(std)***	***RMSE(std)***	***R^2_{test}najbolji***
Voting	0.9863428 (0.0066)	0.1684644(0.0059)	0.2598201(0.0488)	0.991226
Stacking	0.9880908(0.0032)	0.1634299(0.0028)	0.2474444(0.0242)	0.991354
Stack + voting	0.9859631(0.0083)	0.1663547(0.0083)	0.2596103(0.0561)	0.991563
Voting - SVR	0.9849871(0.0074)	0.1700650(0.0134)	0.2715770(0.0521)	0.991013



Slika 7.5 Prikaz rezultata ansambl metoda sa oznakom devijacije



Slika 7.6 Vizualizacija rezultata estimatora za stacking ansambl na treniranim podatcima



Slika 7.7 Prikaz estimacije stacking ansambla na neviđenim podatcima

8. DISKUSIJA

Postavljene hipoteze su ispitane te se došlo do zadovoljavajućeg rezultata korištenjem *stacking* ansambla robusnih modela regresije. U nastavku će se opisati tijek pokusa te dati komentari za proces dobivanja rezultata.

Problem koji se pojavio od početka je činjenica da je trebalo nekako preliminarno ispitati ako je set podataka podatan za treniranje i ako se isplati vršiti pretraživanje hiperparametara za modele koji bi potencijalno mogli imati poželjne rezultate. Zbog toga postavile su se pretpostavke na temelju korelacijske analize da modeli koji imaju robusnu regularizaciju bi mogli imati bolje performanse. Uzevši u obzir da je bilo kritično smanjiti trajanje treniranja čim su se drugi dijelovi seta podataka pokazali da imaju lošu korelaciju odmah su bili izbačeni.

Drugi veliki problem je bio treniranje *MLP-a* zbog velike količine hiperparametara, a na početku vrijeme treniranja mu je bilo najduže od svih drugih modela. Naime pokusom je utvrđeno da koristeći *adam* ili *sgd* kao *solver* nije bilo korisno jer je model ili imao snažne oscilacije koje bi prekinule treniranje ili bi asimptotski počeo konvergirat što znači da bi ušao u neki od lokalnih minimuma. Korigiranje parametara kao što su stopa učenja ili regularizacijski parametar također nebi pomogli jer bi generalno ili počeo naglo oscilirati povečanjem stope učenja ili bi opet našao neki lokalni minimum promjenom regularizacije. Uzevši u obzir da je jedino sa *lbfgs solverom* dobivao rezultate treniranje se značajno pojednostavilo jer on zahtjeva najmanje hiperparametara za izvršiti. U procesu pretraživanja hiperparametara isprobano je korištenje nasumičnog pretraživanja, ali ono nije dalo dobre rezultate, a pretraživanje mreže je bilo prekompleksno za ručno namještanje. Ispalo je da nije bilo potrebno raditi ručna podešavanja jer je Bayes metoda bila vrlo precizna, a vremenski izuzetno efikasna. Doduše problem se javio kod primjene u učenju *MLP-a* te se morala ručno napisat funkcija koja bi omogućila pretragu.

Prilikom treniranja uvedena su skaliranja iako nisu značajno utjecala na treniranje modela osim u slučaju *SVR* algoritma koji nije bio u stanju napraviti nikakvu estimaciju dok se nije uveo *MinMaxScaler*, a *Ridge* regresor je čak radio bolje bez ikakvog skaliranja. Na kraju jedina stvar koja je utjecala na kvalitetu učenja u gotovo svim modelima je bila mogućnost obrade seta podataka te je provedeno dosta vremena testirajući utjecaj *PCA* na modele. Zato je bilo kritično uvesti *pipeline* metodu spajanja estimatora sa transformatorima kako bi se tok podatka uvijek odvijao isto i da se ne pojavi curenje informacija.

Prilikom testiranja ansambl metoda nijedan od metoda *boosting-a* ili *extra tree-a,* a ni *bagging* nije mjerljivo utjecao na rezultate, što je popratilo početne testove gdje su ispitani regresori

bazirani na stablima te nijedan nije bio koristan u primjeni. Na kraju *stacking* i *voting* metode ansambla su se pokazale najboljima te su bile spajane u raznim oblicima da bi na kraju ispalo da je najkvalitetniji estimator bio čisti *stacking*. Kako bi se provjerilo funkcionira li program na generalno ili je model samo istreniran na podatcima koju su dostupni skinut je još jedan set podataka molekule benzena koji je imao pola uzoraka kao ovaj koji se koristio. Podatci o elektronskoj gustoći su učitani te su uzete značajke sa indeksa koji prate indekse originalnog seta podataka te su direktno povučeni u estimator bez učenja i dali su gotovo istu preciznost rezultata što se može vidjeti na Slikama 7.5 i 7.6.

9. ZAKLJUČAK

U procesu izrađivanja rada odrađeno je istraživanje na polju prirodnih znanosti i računalnih tehnologija kako bi se napravilo idejno rješenje i pokušaj modeliranja sustava računalnih metoda strojnog učenja u svrhu estimacije energetske razine. Na temelju aktualne literature pronađen je i odabran javno dostupni set podataka u tu svrhu. Set podataka je analiziran i na temelju pretpostavki proizašlih iz teorije i drugih radova postavljene su smjernice po kojima se set podataka obradio u svrhu korištenja u strojnom učenju. Istom procedurom analizirani su i odabrani algoritmi regresije te su testirani i optimizirani na setu podataka. Set podataka uspješno se obradio smanjivanjem veličine dimenzija te standardiziranjem i normaliziranjem vrijednosti. U tijeku izrađivanja rada i vršenja pokusa ispunjavala su se istraživačka pitanja postavljena na početku i rezultati su bili zadovoljavajući te su se svakom sljedećom iteracijom unapređivali. Uspješno se izvršila pretraga hiperparametara i optimizacija modela strojnog učenja što se vidjelo na rezultatima dobivenim unakrsnom validacijom. Na kraju ispitane su metode spajanja modela u ansambl te su dalje unaprijeđeni rezultati. Postignute su vrijednosti u razini $R_2 \geq 0.98$, $MAE \leq 0.2$, $RMSE \leq 0.3$ i standardnim devijacijama u trećoj decimali. Približne vrijednosti dobivene su učitavanjem seta podataka koje nisu bile korištene u procesu istraživanja te je pokazano da je moguće estimirati osnovnu energetsku razinu uz dostupnu elektronsku gustoću molekule.

10. LITERATURA

[1] Bogojeski, Mihail, et al. "Quantum chemical accuracy from density functional approximations via machine learning." *Nature communications* 11.1 (2020): 5223.

[2]Cramer, Christopher J. Essentials of computational chemistry: theories and models. John Wiley & Sons, 2013.

[3] Zhang, Xuan, et al. "Artificial intelligence for science in quantum, atomistic, and continuum systems." arXiv preprint arXiv:2307.08423 (2023).

[4] Hermann, Jan, Zeno Schätzle, and Frank Noé. "Deep-neural-network solution of the electronic Schrödinger equation." Nature Chemistry 12.10 (2020): 891-897

[5] Bartók, Albert P., et al. "Machine learning unifies the modeling of materials and molecules." Science advances 3.12 (2017): e1701816.

[6] Chmiela, Stefan, et al. "Machine learning of accurate energy-conserving molecular force fields." *Science advances* 3.5 (2017): e1603015.

[7]Behler, Jörg, and Michele Parrinello. "Generalized neural-network representation of high-dimensional potential-energy surfaces." Physical review letters 98.14 (2007): 146401.

[8]Hansen, Katja, et al. "Assessment and validation of machine learning methods for predicting molecular atomization energies." Journal of Chemical Theory and Computation 9.8 (2013): 3404-3419.

[9] Brockherde, F., Vogt, L., Li, L., Tuckerman, M. E., Burke, K., & Müller, K. R. (2017). Bypassing the Kohn-Sham equations with machine learning. Nature communications, 8(1), 872.

[10]Snyder, John C., et al. "Finding density functionals with machine learning." Physical review letters 108.25 (2012): 253002.

[11]Dick, S., & Fernandez-Serra, M. (2020). Machine learning accurate exchange and correlation functionals of the electronic density. Nature communications, 11(1), 3509.

[12]Bowman, Joel M. et al. “The MD17 datasets from the perspective of datasets for gas-phase "small" molecule potentials.” The Journal of chemical physics 156 24 (2022): 240901 .

[13]Slater, J.C. (1960) Quantum Theory of Atomic Structure. Vol. 1, McGraw-Hill Book Company, New York.

[14]Kohn, Walter, and Lu Jeu Sham. "Self-consistent equations including exchange and correlation effects." *Physical review* 140.4A (1965): A1133.

[15]Hohenberg, P., & Kohn, W. (1964). Inhomogeneous electron gas. *Physical review*, *136*(3B), B864.

[16]Chmiela, S., Sauceda, H. E., Poltavsky, I., Müller, K.-R., Tkatchenko, A., ,“sGDML: Constructing Accurate and Data Efficient Molecular Force Fields Using Machine Learning“ Computer Physics Communications, 240, 2019, pp. 38-45.

[17]sa interneta http://www.quantum-machine.org/datasets/ zadnji pristup 3.3.2024

[18] Harris, Charles R., et al. "Array programming with NumPy." Nature 585.7825 (2020): 357-362.

[19] Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

[20] Hastie, Trevor & Tibshirani, Robert. (2003). Expression Arrays and the p ≫ n Problem.

[21] Johnstone, Iain M., and D. Michael Titterington. "Statistical challenges of high-dimensional data." Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367.1906 (2009): 4237-4253.

[22] Lazic, S. E. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn."

[23] Chan, Jireh Yi-Le, et al. "Mitigating the multicollinearity problem and its machine learning approach: a review." Mathematics 10.8 (2022): 1283.

[24]Wan, Zhongyu, et al. "Effectively improving the accuracy of PBE functional in calculating the solid band gap via machine learning." Computational Materials Science 198 (2021): 110699.

[25]Hall, Mark A. Correlation-based feature selection for machine learning. Diss. The University of Waikato, 1999.

[26]Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.

[27] Ayesha, Shaeela, Muhammad Kashif Hanif, and Ramzan Talib. "Overview and comparative study of dimensionality reduction techniques for high dimensional data." Information Fusion 59 (2020): 44-58.

[28] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[29] Smola, Alex J., and Bernhard Schölkopf. "A tutorial on support vector regression." Statistics and computing 14 (2004): 199-222.

[30] Wang, X., Dang, X., Peng, H., & Zhang, H. (2009). THE THEIL-SEN ESTIMATORS IN A MULTIPLE LINEAR REGRESSION MODEL.

[31] Mendes-Moreira, J., Soares, C., Jorge, A. M., & Sousa, J. F. D. (2012). Ensemble approaches for regression. ACM Computing Surveys, 45(1), 1–40.

[32] Chen, S., & Luc, N. M. (2022). RRMSE Voting Regressor: A weighting function based improvement to ensemble regression. arXiv preprint arXiv:2207.04837.

[33] Louppe, Gilles. "Bayesian optimisation with scikit-optimize." PyData Amsterdam. 2017.

11. SAŽETAK I KLJUČNE RIJEČI

Tema rada je estimacija osnovnog energetskog stanja molekula primjenom modela strojnog učenja. Napravljen je uvod u polje kvantne kemije i hipoteze su definirane te je odrađen pregled literature. Pregled se sastojao od uvoda i definicija iz kvantne mehanike kako bi se postavila temeljna razina znanja. Nakon toga je analiziran i obrađen set podataka dobiven korištenjem teorije funckionala gustoće. U toku rada nabrojane su metode strojnog učenja i opisan je generalni postupak kojim se takvi modeli izrađuju i treniraju. Obradom seta podataka korištenjem metoda redukcije dimenzija kao što su *PCA* sukcesivno su trenirani i optimizirani modeli estimacije te su u svakom tom koraku prezentirani rezultati.

Kada se set podataka pročistio i modeli estimatora se optimizirali definirane su ansambl metode koje su se istrenirale u nekoliko kombinacija nakon čega su se prikazali rezultati i odabran je najuspješniji model. Metode su se evaluirale standardnim metrikama R^2, *MAE* i *RMSE* i standardnom devijacijom istih. Odabirom najboljeg modela ansambla učitan je set podataka koji nije bio korišten za treniranje kako bi se potvrdilo uspješno istraživanje. U zadnjem poglavlju prokomentiran je postupak istraživanja i odluke iza određenih metoda.

Ključne riječi: umjetna inteligencija, strojno učenje, teroija funkcionala gustoće, estimacija osnovnog energetskog stanja, Python, Principal Component Analysis, PCA, redukcija dimenzija, optimizacija hiperparametara, scikit-learn, multivarijatna regresija

12. SUMMARY AND KEYWORDS

The theme of this paper was estimating ground state energies of a molecule using machine learning methods. At the start of the paper the reader is introduced into the field of quantum chemistry and the hypothesis was defined followed by a brief look at the state of the art. Some ground rules were set in the way of theory and formulas to help in understanding the process. This was followed by an analysis and visualisation of the dataset that was generated on the basis of density functional theory. Methods of machine learning were described and the process of model selection and training was defined. After setting up the dataset methods of dimensionality reduction via *PCA* was explained and the methods were applied with succesive training and optimisation of the models followed by displaying the results.

When the dataset was sufficienty reduced and estimators were well optimised ensemble methods were defined. These were trained and cross valdiated in a few different combination and the most efficient one was selected. The methods were evaluated by the standard evaluation metrics R^2, *MAE*, *RMSE* and their standard deviation. After selecting the best ensemble method a fresh dataset was loaded and applied to the estimator to confirm a succesful research. Last chapter was dedicated to discussion and commentary on the process and reasoning behind certain decisions.

Keywords: artificial intelligence, machine learning, density functional theory, ground state energy estimation, Python, scikit-learn, Principal Component Analysis, PCA, dimensionality reduction, hiperparameter optimisation, multivariate regression

POPIS SLIKA

POPIS TABLICA

DODATAK A – PROGRAMSKI KOD

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn import svm
from sklearn.linear_model import  RidgeCV, TheilSenRegressor
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, cross_validate

#%%Ucitavanje dataseta
dftEnergije = np.load("dft_energies.npy").ravel()
ccEnergije = np.load("cc_energies.npy")
dftGustoce = np.load("dft_densities.npy")
strukture = np.load("structures.npy")
##uklanjanje nula iz seta elektronske gustoce
index_zero = np.argwhere(np.all(dftGustoce[:,:] == 0, axis=0))
dataDensity=np.delete(dftGustoce,index_zero,axis=1)
X = dataDensity
Y = dftEnergije
#X_train, X_test, y_train, y_test = train_test_split(X, Y,random_state=40,
test_size=0.2)
print("Dataset loaded")
#%%Petlja za racunanje korelacije za elektronsku gustocu

blok= np.hsplit(dataDensity,49)
corr_maska=[]
corr_koef=[]
i=0

for i in range(49):
    tmp_blok=np.corrcoef(blok[i], dftEnergije, rowvar=False)
    tmp_corr=tmp_blok[2401, :2401]
    corr_koef=np.append(corr_koef,tmp_corr)
    tmp_corr=(0.4 < tmp_corr) | (tmp_corr < -0.4)
    corr_maska=np.append(corr_maska,tmp_corr)

index_corr = np.argwhere(corr_maska == 0)
corDensity=np.delete(dataDensity,index_corr,axis=1)
#plt.plot(corr_koef)
print("Korelacija gotova")

#%% PCA
pca1 = PCA()
X_pca1 = pca1.fit_transform(corDensity)
# print(pca.explained_variance_ratio_)
# print(np.cumsum(pca.explained_variance_ratio_))
# print(pca.singular_values_)
pca_vrijednosti1=pca1.singular_values_
pca_opis1=np.cumsum(pca1.explained_variance_ratio_)
pca2 = PCA(n_components=100)
X_pca2 = pca2.fit_transform(corDensity)
# print(pca.explained_variance_ratio_)
# print(np.cumsum(pca.explained_variance_ratio_))
# print(pca.singular_values_)
pca_vrijednosti2=pca2.singular_values_
pca_opis2=np.cumsum(pca2.explained_variance_ratio_)
```

```
fig=plt.figure()
# fig, ax1= plt.subplots()
ax1 = fig.add_subplot(2,1,1)
plt.grid(True)
color = 'tab:red'
ax1.set_xlabel('Broj komponente')
ax1.set_ylabel('Iznos glavne komponente', color=color)
ax1.plot(pca_vrijednosti1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Postotak opisa %', color=color)
ax2.plot(pca_opis1, color=color)
ax2.tick_params(axis='y', labelcolor=color)

#fig, ax3= plt.subplots()
ax3 = fig.add_subplot(2,1,2)
plt.grid(True)
color = 'tab:red'
ax3.set_xlabel('Broj komponente')
ax3.set_ylabel('Iznos glavne komponente', color=color)
ax3.plot(pca_vrijednosti2, color=color)
ax3.tick_params(axis='y', labelcolor=color)
ax4 = ax3.twinx()
color = 'tab:blue'
ax4.set_ylabel('Postotak opisa %', color=color)
ax4.plot(pca_opis2, color=color)
ax4.tick_params(axis='y', labelcolor=color)

print("PCA gotov")

#%%Definiranje modela regresora
#X_train, X_test, y_train, y_test = train_test_split(corDensity,
Y,random_state=40, test_size=0.2)


MLPreg =
MLPRegressor(hidden_layer_sizes=(86,14,20),activation="identity",solver='lbfg
s',

alpha=0.15053789557715627,max_iter=150000,max_fun=150000)#.fit(X_train,
y_train)

# y_predict_mlp = MLPreg.predict(X_test)
# print("R2 train mlp",MLPreg.score(X_train,y_train))
# print("R2 test MLP",MLPreg.score(X_test,y_test))
# print("MAE MLP =", mean_absolute_error(y_test, y_predict_mlp))
# print("RMSE MLP =",np.sqrt(mean_squared_error(y_test, y_predict_mlp)))

SVR=svm.SVR(C=698.1814745879186,gamma=0.04904559995067583,cache_size=10000,ep
silon=0.03342728611844894, kernel='rbf')#.fit(X_train,y_train)
# y_SVR_predict = SVR.predict(X_test)
# print("R2_train SVR =", SVR.score(X_train,y_train))
# print("R2_test SVR =", SVR.score(X_test,y_test))
# print("MAE SVR =", mean_absolute_error(y_test, y_SVR_predict))
#y_full_SVR = SVR.predict(X)

theilreg = TheilSenRegressor(max_subpopulation=1e4, random_state=0,n_jobs=-
1)#.fit(X_train, y_train)
# y_theilreg_predict = theilreg.predict(X_test)
# print("R2_train theil =", theilreg.score(X_train,y_train))
```

```
# print("R2_test theil =", theilreg.score(X_test,y_test))
# print("MAE theil =", mean_absolute_error(y_test, y_theilreg_predict))
#y_full_theilreg = theilreg.predict(X)

linear_ridge=RidgeCV(alphas=[1e-4,1e-3,1e-2,1e-1,1,10])#.fit(X_train,y_train)
# y_ridge_predict = linear_ridge.predict(X_test)
# print("R2_train ridge =", linear_ridge.score(X_train,y_train))
# print("R2_test ridge =", linear_ridge.score(X_test,y_test))
# print("MAE ridge =", mean_absolute_error(y_test, y_ridge_predict))
#y_full_ridge = linear_ridge.predict(X)
print("Regresori fitani")
#%% Spajanje regresora u pipeline sa scalerom i PCA
#
#X_train, X_test, y_train, y_test = train_test_split(X, Y,random_state=40,
test_size=0.2)
#X_train, X_test, y_train, y_test = train_test_split(corDensity,
Y,random_state=40, test_size=0.2)

pipe_theil = Pipeline([("redukcija",PCA(n_components=250)),
                       ("skaliranje",StandardScaler()),
                       ("regresor",theilreg)])
# pipe_theil.fit(X_train, y_train)
# print("R^2 train theil", pipe_theil.score(X_train, y_train))
# print("R^2 test theil", pipe_theil.score(X_test,y_test))
# print("MAE theil =", mean_absolute_error(y_test,
pipe_theil.predict(X_test)))

pipe_ridge = Pipeline([("redukcija",PCA(n_components=300)),
                       #("skaliranje",StandardScaler()),
                       ("regresor",linear_ridge)])
# pipe_ridge.fit(X_train, y_train)
# print("R^2 train ridge", pipe_ridge.score(X_train, y_train))
# print("R^2 test ridge", pipe_ridge.score(X_test,y_test))
# print("MAE ridge =", mean_absolute_error(y_test,
pipe_ridge.predict(X_test)))

pipe_mlp = Pipeline([("redukcija",PCA(n_components=250)),
                     ("skaliranje",StandardScaler()),
                     ("regresor",MLPreg)])
# pipe_mlp.fit(X_train, y_train)
# print("R^2 train mlp", pipe_mlp.score(X_train, y_train))
# print("R^2 test mlp", pipe_mlp.score(X_test,y_test))
# print("MAE MLP =", mean_absolute_error(y_test, pipe_mlp.predict(X_test)))
#print("RMSE MLP =",np.sqrt(mean_squared_error(y_test,
pipe_mlp.predict(X_test))))

pipe_svr = Pipeline([("redukcija",PCA(n_components=100)),
                     ("skaliranje",MinMaxScaler()),
                     ("regresor",SVR)])
# pipe_svr.fit(X_train, y_train)
# print("R^2 train svr", pipe_svr.score(X_train, y_train))
# print("R^2 test svr", pipe_svr.score(X_test,y_test))
# print("MAE svr =", mean_absolute_error(y_test, pipe_svr.predict(X_test)))

scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
score_theil = cross_validate(pipe_theil, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
score_ridge = cross_validate(pipe_ridge, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
score_mlp = cross_validate(pipe_mlp, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
score_svr = cross_validate(pipe_svr, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
```

```
cv_res_theil_final=score_theil
cv_res_ridge_final=score_ridge
cv_res_mlp_final=score_mlp
cv_res_svr_final=score_svr
st_dev_final={
    'MLP_r2': np.std(np.array(list(cv_res_mlp_final['test_r2']))),
    'MLP_MAE':
np.std(np.array(list(cv_res_mlp_final['test_neg_mean_absolute_error']))),
    'MLP_RMSE':
np.std(np.array(list(cv_res_mlp_final['test_neg_root_mean_squared_error']))),
    'theil_r2': np.std(np.array(list(cv_res_theil_final['test_r2']))),
    'theil_MAE':
np.std(np.array(list(cv_res_theil_final['test_neg_mean_absolute_error']))),
    'theil_RMSE':
np.std(np.array(list(cv_res_theil_final['test_neg_root_mean_squared_error']))
),
    'Ridge_r2': np.std(np.array(list(cv_res_ridge_final['test_r2']))),
    'Ridge_MAE':
np.std(np.array(list(cv_res_ridge_final['test_neg_mean_absolute_error']))),
    'Ridge_RMSE':
np.std(np.array(list(cv_res_ridge_final['test_neg_root_mean_squared_error']))
),
    'SVR_r2': np.std(np.array(list(cv_res_svr_final['test_r2']))),
    'SVR_MAE':
np.std(np.array(list(cv_res_svr_final['test_neg_mean_absolute_error']))),
    'SVR_RMSE':
np.std(np.array(list(cv_res_svr_final['test_neg_root_mean_squared_error']))),
        }


print("Pipeline slozen")
#%%Pretrazivanje PCA n_components za X i corDensity
#X_train, X_test, y_train, y_test = train_test_split(X, Y,random_state=40,
test_size=0.2)
X_train, X_test, y_train, y_test = train_test_split(corDensity,
Y,random_state=40, test_size=0.2)

pipe_mlp = Pipeline([ ("redukcija",PCA()),
                      ("skaliranje",StandardScaler()),
                      ("regresor",MLPreg)])
param_grid={"redukcija__n_components":[50,100,200,250,300,400]}
scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
search_mlp=GridSearchCV(estimator=pipe_mlp,cv=3,
param_grid=param_grid,n_jobs=-
1,scoring=scoring,refit='r2',return_train_score=True)
search_mlp.fit(X_train,y_train)
print("mlp",search_mlp.best_params_)
print("mlp ",search_mlp.best_score_)
print("mlp
MAE",min(abs(search_mlp.cv_results_['mean_test_neg_mean_absolute_error'])))
print("mlp
RMSE",min(abs(search_mlp.cv_results_['mean_test_neg_root_mean_squared_error']
)))
#df_mlp_result_full = pd.DataFrame(search_mlp.cv_results_)
df_mlp_result_kor = pd.DataFrame(search_mlp.cv_results_)

pipe_theil = Pipeline([ ("redukcija",PCA()),
                      ("skaliranje",StandardScaler()),
                      ("regresor",theilreg)])
param_grid={"redukcija__n_components":[50,100,200,250,300,400]}
scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
```

```
search_theil=GridSearchCV(estimator=pipe_theil,cv=3,
param_grid=param_grid,n_jobs=-
1,scoring=scoring,refit='r2',return_train_score=True)
search_theil.fit(X_train,y_train)
print("theil",search_theil.best_params_)
print("theil",search_theil.best_score_)
print("theil
MAE",min(abs(search_theil.cv_results_['mean_test_neg_mean_absolute_error'])))
print("theil
RMSE",min(abs(search_theil.cv_results_['mean_test_neg_root_mean_squared_error
']))))
#df_theil_result_full = pd.DataFrame(search_theil.cv_results_)
df_theil_result_kor = pd.DataFrame(search_theil.cv_results_)

pipe_ridge = Pipeline([ ("redukcija",PCA()),
                        #("skaliranje",StandardScaler()),
                        ("regresor",linear_ridge)])
param_grid={"redukcija__n_components":[50,100,200,250,300,400]}
scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
search_ridge=GridSearchCV(estimator=pipe_ridge,cv=3,
param_grid=param_grid,n_jobs=-
1,scoring=scoring,refit='r2',return_train_score=True)
search_ridge.fit(X_train,y_train)
print("ridge",search_ridge.best_params_)
print("ridge",search_ridge.best_score_)
print("ridge
MAE",min(abs(search_ridge.cv_results_['mean_test_neg_mean_absolute_error'])))
print("ridge
RMSE",min(abs(search_ridge.cv_results_['mean_test_neg_root_mean_squared_error
']))))
#df_ridge_result_full = pd.DataFrame(search_ridge.cv_results_)
df_ridge_result_kor = pd.DataFrame(search_ridge.cv_results_)

pipe_svr = Pipeline([ ("redukcija",PCA()),
                      ("skaliranje",MinMaxScaler()),
                      ("regresor",SVR)])
param_grid={"redukcija__n_components":[50,100,200,250,300,400]}
scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
search_svr=GridSearchCV(estimator=pipe_svr,cv=3,
param_grid=param_grid,n_jobs=-
1,scoring=scoring,refit='r2',return_train_score=True)
search_svr.fit(X_train,y_train)
print("svr n_komp",search_svr.best_params_)
print("svr R2",search_svr.best_score_)
print("svr
MAE",min(abs(search_svr.cv_results_['mean_test_neg_mean_absolute_error'])))
print("svr
RMSE",min(abs(search_svr.cv_results_['mean_test_neg_root_mean_squared_error']
)))
#df_svr_result_full = pd.DataFrame(search_svr.cv_results_)
df_svr_result_kor = pd.DataFrame(search_svr.cv_results_)



#%% Pretrazivanje hiperparametara preko BayesianCV()
from skopt import BayesSearchCV
from skopt.space import Real, Categorical, Integer
from skopt.plots import plot_objective, plot_histogram
np.int = int
from sklearn.model_selection import cross_val_score
from skopt.space import Real, Integer, Categorical
from skopt.utils import use_named_args
from skopt import gp_minimize
```

```
from sklearn.base import BaseEstimator, ClassifierMixin

rez_opt={'R2_train_best':[],
          'R2_test':[],
          'model_param':[]
        }
pipe_bayes = Pipeline([("redukcija",PCA(n_components=100)),
                       ("skaliranje",MinMaxScaler()),
                       ("model",svm.SVR())])

svr_search={
    'model': Categorical([svm.SVR()]),
    'model__C': Real(1e0, 1e+6, prior='log-uniform'),
    'model__gamma': Real(1e-8, 1e0, prior='log-uniform'),
    'model__epsilon': Real(1e-4, 1, prior='log-uniform'),
    'model__kernel': Categorical(['rbf'])
}


opt = BayesSearchCV(
    pipe_bayes,
    [(svr_search, 30)],
    cv=5)
opt.fit(X_train, y_train)
print("val. score: %s" % opt.best_score_)
print("test score: %s" % opt.score(X_test, y_test))
print("best params: %s" % str(opt.best_params_))

## MLP wrapper za korištenje s Bayes search
class MLPWrapper(BaseEstimator, ClassifierMixin):
    def __init__(self, layer1=10, layer2=10, layer3=10,alpha=1e-
4,activation='identity',solver='lbfgs'):
        self.layer1 = layer1
        self.layer2 = layer2
        self.layer3 = layer3
        self.alpha = alpha
        self.activation=activation
        self.solver=solver

    def fit(self, X, y):
        model = MLPRegressor(
            hidden_layer_sizes=[self.layer1, self.layer2, self.layer3],
            alpha=self.alpha,
            activation=self.activation,
            solver=self.solver
        )
        model.fit(X, y)
        self.model = model
        return self

    def predict(self, X):
        return self.model.predict(X)

    def score(self, X, y):
        return self.model.score(X, y)

pipe_bayes_mlp = Pipeline([("redukcija",PCA(n_components=250)),
                       ("skaliranje",StandardScaler()),
                       ("model",MLPWrapper())])

opt = BayesSearchCV(
    estimator=pipe_bayes_mlp,
    search_spaces={
```

```
        'model__layer1': Integer(10, 100),
        'model__layer2': Integer(10, 100),
        'model__layer3': Integer(10, 100),
        'model__alpha':Real(1e-8, 10, prior='log-uniform'),
        'model__solver':Categorical(['lbfgs']),
        'model__activation':Categorical(['identity'])
    },
    n_iter=11
)

opt.fit(X_train, y_train)
opt.score(X_test,y_test)
print("val. score: %s" % opt.best_score_)
print("test score: %s" % opt.score(X_test, y_test))
print("best params: %s" % str(opt.best_params_))

rez_opt['R2_train_best'].append(opt.best_score_)
rez_opt['R2_test'].append(opt.score(X_test, y_test))
rez_opt['model_param'].append(str(opt.best_params_))


h = plot_objective(opt.optimizer_results_[0],
                   #dimensions=["C", "gamma", "epsilon"],
                   dimensions=["layer1","layer2","layer3","alpha"],
                   n_minimum_search=int(1e8)
                   )
plt.show()


#%%Primjena ansambl metoda
from sklearn.ensemble import VotingRegressor, StackingRegressor
modeli=[('mlp', pipe_mlp), ('ridge', pipe_ridge), ('theil', pipe_theil)
,('svr', pipe_svr),]

stacking=StackingRegressor(estimators=modeli,final_estimator=None,n_jobs=-
1,verbose=1,passthrough=False).fit(X_train, y_train)
y_predict_stacking = stacking.predict(X_test)
print("R^2 train stacking", stacking.score(X_train, y_train))
print("R^2 test stacking", stacking.score(X_test,y_test))
print("MAE", mean_absolute_error(y_test, y_predict_stacking))
print("RMSE", np.sqrt(mean_squared_error(y_test, y_predict_stacking)))
print("Stacking gotov")

#modeli=[('mlp', pipe_mlp), ('ridge', pipe_ridge), ('theil', pipe_theil)
,('svr', pipe_svr),('stack',stacking)]
voting = VotingRegressor(estimators=modeli,n_jobs=-
1,verbose=True).fit(X_train, y_train)
y_predict_voting = voting.predict(X_test)
print("R^2 train voting", voting.score(X_train, y_train))
print("R^2 test voting", voting.score(X_test,y_test))
print("MAE", mean_absolute_error(y_test, y_predict_voting))
print("RMSE", np.sqrt(mean_squared_error(y_test, y_predict_voting)))
print("Voting gotov")

scoring=['neg_mean_absolute_error','r2','neg_root_mean_squared_error']
score_voting = cross_validate(voting, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
score_stacking = cross_validate(stacking, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
stacking_u_voting=cross_validate(stacking, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
score_voting_noSVR = cross_validate(voting, X_train, y_train,
cv=5,scoring=scoring,return_train_score=True,n_jobs=-1)
```

```
y_predict_full_voting = voting.predict(corDensity)
print("MAEfull voting", mean_absolute_error(Y, y_predict_full_voting))
# #
y_predict_full_stacking = stacking.predict(corDensity)
print("MAEfull stacking", mean_absolute_error(Y, y_predict_full_stacking))



#%%Vizualizacija greske predikcije preko reziduala i prikaza estimacije
from yellowbrick.regressor import PredictionError
from yellowbrick.regressor import ResidualsPlot
model = stacking

fig, (ax1,ax2)=plt.subplots(1,2)
visualizer_pred = PredictionError(model,ax=ax1)
visualizer_res = ResidualsPlot(model,ax=ax2)

visualizer_res.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer_res.score(X_test, y_test)  # Evaluate the model on the test data
visualizer_res.show()

visualizer_pred.fit(X_train, y_train)  # Fit the training data to the
visualizer
visualizer_pred.score(X_test, y_test)  # Evaluate the model on the test data
visualizer_pred.show()
#%% Plot za vizualizaciju i analizu podataka

## Prikaz utjecaja StandardScaler()
scaler=StandardScaler()
minmax=MinMaxScaler()
#X_pca=PCA().fit_transform(X)
X_scale_std=scaler.fit_transform(X)
X_scale_minmax=minmax.fit_transform(X)
fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(5,5))
ax1.hist(X.flatten(),bins=1000)
ax2.hist(X_scale_std.flatten(),bins=1000)
ax3.hist(X_scale_minmax.flatten(),bins=1000)
plt.show()


##Prikaz atoma benzena u prostoru
struk= strukture[0,:,:]
x = struk[:,0]
y = struk[:,1]
z = struk[:,2]
r=[100, 100, 100, 100, 100, 100, 25, 25, 25, 25, 25, 25] #skaliranje za C i H
atome

fig = plt.figure()
fig.suptitle('Struktura molekule benzena u xyz formatu')
ax = fig.add_subplot(1, 2, 1,projection='3d')
ax.set_zticks([])
ax.set_xlabel('X koordinata')
ax.set_ylabel('Y koordinata')
ax.scatter3D(x, y, z,c='r',s=r)
ax.view_init(elev=90, azim=-90, roll=0)
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20

ax = fig.add_subplot(1, 2, 2,projection='3d')
ax.scatter3D(x, y, z,c='r',s=r)
```

```
ax.set_xlabel('X koordinata')
ax.set_ylabel('Y koordinata')
ax.set_zlabel('Z koordinata')
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20
ax.zaxis.labelpad = 40
plt.show()


##Pretvaranje 1D koeficijente elektronske gustoce u 3D prikaz inverznim FFT
volumen_koef = dftGustoce[1,:].reshape(50,50,50)
prostorna_gustoca=abs(np.fft.ifft2(volumen_koef))
#Prikaz koeficijenata elektronske gustoce u 1D i 3D
from itertools import product
from matplotlib import pyplot as plt
N = 50
fig = plt.figure()
ax = fig.add_subplot(1,2,1,projection="3d")
ax.set_title('Vizualizacija koeficijenata u prostoru')
prostor = np.array([*product(range(N), range(N), range(N))])
ax.scatter(prostor[:,0], prostor[:,1], prostor[:,2], c=prostor/50,cmap='hsv',
s=volumen_koef)
ax.set_xlabel('X koordinata')
ax.set_ylabel('Y koordinata')
ax.set_zlabel('Z koordinata')
ax.xaxis.labelpad = 20
ax.yaxis.labelpad = 20
ax.zaxis.labelpad = 30
ax = fig.add_subplot()
ax.set_title("Prikaz koeficijenata iz dataseta")
plt.plot(dftGustoce[0,:])
ax.set_xlabel('Koeficijent gustoće')
ax.set_ylabel('Amplituda')

plt.show()
# 3D prikaz elektronske gustoce
from mayavi import mlab
3D_gustoca = mlab.contour3d(abs(prostorna_gustoca))
#%% Subplot za sve modele PCA
modeli={
    'MLP': CV_MLP_KOR[['mean_test_r2']].copy().to_numpy().ravel(),
    'theil': CV_THEIL_KOR[['mean_test_r2']].copy().to_numpy().ravel(),
    'Ridge': CV_RIDGE_KOR[['mean_test_r2']].copy().to_numpy().ravel(),
    'SVR': CV_SVR_KOR[['mean_test_r2']].copy().to_numpy().ravel(),
}

devijacije={
    'MLP': CV_MLP_KOR[['std_test_r2']].copy().to_numpy().ravel(),
    'theil': CV_THEIL_KOR[['std_test_r2']].copy().to_numpy().ravel(),
    'Ridge': CV_RIDGE_KOR[['std_test_r2']].copy().to_numpy().ravel(),
    'SVR': CV_SVR_KOR[['std_test_r2']].copy().to_numpy().ravel(),

    }

tikovi=('50','100','200','250','300','400')
x = np.arange(len(tikovi))  # the label locations

from matplotlib.gridspec import GridSpec

gs = GridSpec(2, 2)
fig = plt.figure()
fig.suptitle('Rezultati za različite vrijednosti n_components za cijeli
dataset')
```

```
axe1 = fig.add_subplot(gs[0, 0])
axe2 = fig.add_subplot(gs[0, 1])
axe3 = fig.add_subplot(gs[1, 0])
axe4 = fig.add_subplot(gs[1, 1])

axe1.plot(modeli['MLP'])
axe1.errorbar(x,modeli['MLP'], yerr=devijacije['MLP'])
axe1.set_title('MLP PCA')

axe2.plot(modeli['theil'], color='orange')
axe2.errorbar(x,modeli['theil'], yerr=devijacije['theil'],color='orange')
axe2.set_title('TheilSen PCA')

axe3.plot(modeli['Ridge'], color='green')
axe3.errorbar(x,modeli['Ridge'], yerr=devijacije['Ridge'],color='green')
axe3.set_title('Ridge PCA')

axe4.plot(modeli['SVR'],color='red')
axe4.errorbar(x,modeli['SVR'], yerr=devijacije['SVR'],color='red')
axe4.set_title('SVR PCA')

for axe in [axe1,axe2,axe3,axe4]:
    axe.grid(True)
    axe.set(xlabel='n_components', ylabel='R2 score')
    axe.set_xticks(x,tikovi)

#%%
##računannje Couloumb matrice
potencijal=np.zeros((12,12))
for n in range(1001):
    struk=strukture[n,:,:]
    cij = np.zeros((12, 12))
    potencijalarray=np.array([6,6,6,6,6,6,1,1,1,1,1,1])
    for i in range(12):
        for j in range(12):
            if i==j:
                cij[i][j]=0.5*potencijalarray[i]**2.4
            else:
                    distance=np.linalg.norm(np.array(struk[i])-
np.array(struk[j]))

cij[i][j]=(potencijalarray[i]*potencijalarray[j])/distance
    potencijal=np.dstack([potencijal,cij])
potencijal=np.delete(potencijal.T,0,0)

##korelacija za potencijale naspram energiji
potencijal2=np.reshape(potencijal,[1001,144])
with np.errstate(divide="ignore", invalid="ignore"): #preskace dijeljenje s
nulom zbog vrijednosti stdev=0
    potencijal_corr=np.corrcoef(potencijal2, dftEnergije, rowvar=False)
##korelacija za geometriju naspram energiji
geometrija=np.reshape(strukture,[1001,36])
geometrija_corr=np.corrcoef(geometrija, dftEnergije, rowvar=False)

import seaborn as sns
##Korelacijski heatmapovi
sns.heatmap(tmp_blok)
sns.heatmap(potencijal_corr)
sns.heatmap(geometrija_corr)
sns.heatmap(np.corrcoef(corDensity,dftEnergije, rowvar=False))
```