Estimacija temperature elektromotora primjenom složenih algoritama strojnog učenja

Štimac, Tomislav

Master's thesis / Diplomski rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:126226

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2025-01-27**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

ESTIMACIJA TEMPERATURE ELEKTROMOTORA PRIMJENOM SLOŽENIH ALGORITAMA STROJNOG UČENJA

Rijeka, studeni 2023.

Tomislav Štimac
0069067674

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

ESTIMACIJA TEMPERATURE ELEKTROMOTORA PRIMJENOM SLOŽENIH ALGORITAMA STROJNOG UČENJA

Mentor: Prof. dr. sc. Zlatan Car

Rijeka, studeni 2023.

Tomislav Štimac
0069067674

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: **Tomislav Štimac (0069067674)**
Studij: Sveučilišni diplomski studij elektrotehnike
Modul: Automatika

Zadatak: **Estimacija temperature elektromotora primjenom složenih algoritama strojnog učenja**

Opis zadatka:

Napraviti pregled postojećih istraživanja u estimaciji temperature elektromotora električnog automobila primjenom umjetne inteligencije. Na javno dostupnom skupu podataka napraviti statističku analizu i pred-obradu podataka. Ispitati različite algoritme strojnog učenja te ustanoviti koji od njih imaju najveću točnost u estimaciji temperature elektromotora. Točnost estimacije algoritama strojnog učenja poboljšati sa izradom metode nasumičnog odabira hiperparametara i peterostruke unakrsne validacije. Ispitati da li se točnost estimacije temperature elektromotora može poboljšati izradom ansambla koji je sastavljen od više algoritama strojnog učenja.

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

kojom ja, Tomislav Štimac, JMBAG: 0069067674 student Tehničkog fakulteta Sveučilišta u Rijeci sukladno s člankom 9. Pravilnika o diplomskom radu, diplomskom ispitu i završetku sveučilišnih diplomskih studija izjavljujem da sam samostalno izradio svoj diplomski rad pod nazivom „Estimacija temperature elektromotora primjenom složenih algoritama strojnog učenja“. U radu sam primijenio metodologiju znanstvenoistraživačkog rada te sam koristio literaturu navedenu na kraju rada

Rijeka, studeni 2023. Tomislav Štimac

_____

ZAHVALA

Ovim putem želio bih se zahvaliti svom mentoru prof. dr. sc. Zlatanu Caru na prilici za istraživanje u području umjetne inteligencije što se pokazalo zahtjevno, ali i jako zanimljivo iskustvo. Posebno hvala v. asist. dr. sc. Nikoli Anđeliću na prenesenim znanjima i savjetima tijekom izrade ovog rada bez koga bi ovaj put bio znatno teži.

Hvala obitelji, prijateljima i kolegama koji su bili uz mene tijekom studiranja, a posebno dvjema ženama bez kojih ne bih bio ono što jesam. Nona, hvala ti što si mi bila majka, tvoje riječi pamtim kao da je bilo jučer, a tvoj značaj u mom životu nikada neću zaboraviti. Najveće hvala mojoj ljubavi uz čiju podršku sam uspio napisati ovaj rad.

SADRŽAJ

1. UVOD

Integracija električnih pogonskih sustava u moderne automobile donijela je revoluciju u automobilsku industriju, nudeći poboljšanu energetsku učinkovitost, smanjene emisije i poboljšane performanse. Ključna komponenta ovih sustava je sinkroni motor s permanentnim magnetom (eng. PMSM – *Permanent Magnet Synchronous Motor*), koji služi kao pokretačka snaga za različite podsustave vozila. PMSM je vrlo popularan i često korišten u industriji zbog superiornog momenta i gustoće snage. Sprječavanje kvara motora i osiguravanje dugotrajnog korištenja zajedno s visokom učinkovitošću je ključ, a to dovodi do problema s temperaturom, posebno u kontekstu primjene u automobilskoj industriji. Temperatura PMSM-a izravno utječe na njihovu učinkovitost, performanse i ukupnu trajnost, što je čini kritičnim parametrom za praćenje i kontrolu radi osiguravanja sigurnog i pouzdanog rada vozila. U tom kontekstu, kombinacija naprednih algoritama strojnog učenja i elektromehaničkog inženjerstva nudi obećavajući put za preciznu procjenu temperature, čime se pridonosi evoluciji automobilskih tehnologija [1,2].

Rad PMSM-a pod različitim uvjetima i opterećenjima uvodi složenost u upravljanje temperaturom. Pregrijavanje ne samo da ugrožava performanse i učinkovitost motora, već također predstavlja sigurnosne rizike te ubrzava trošenje i habanje. U statoru postoji problem s propadanjem zaštitne izolacije što može dovesti do djelomičnog pražnjenja, smanjenja životnog vijeka i kvarova. Problemi s rotorom povezani su sa smanjenjem magnetskog toka permanentnog magneta koji može dovesti do nepovratne demagnetizacije magneta. I na kraju, senzori koje je teško implementirati, obično su nezamjenjivo ugrađeni te njihova funkcionalnost degradira s vremenom. Motori nisu jeftini i nije ih lako nabaviti pa je ključno iskoristiti što veći potencijal preopterećenja bez oštećenja ili skraćivanja njihovog životnog vijeka. Tradicionalne metode procjene temperature, koje se oslanjaju na toplinske modele i fizičke senzore, imaju ograničenja u hvatanju varijacija i prilagođavanju dinamičkim operativnim profilima [2-5].

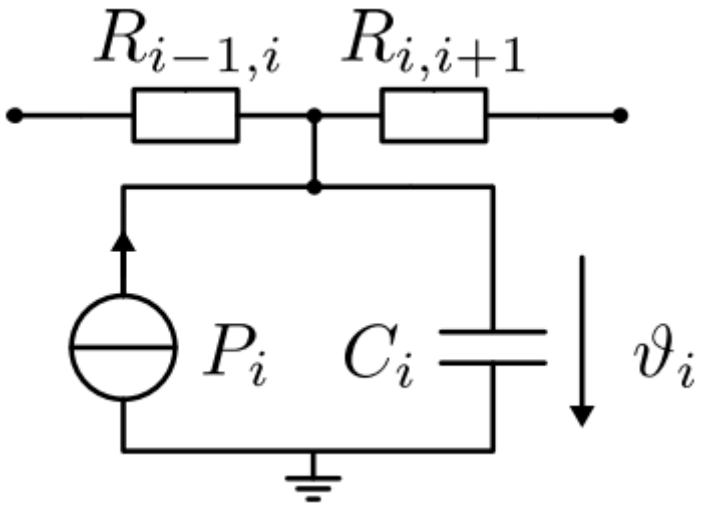
Nakon razmatranja problema dolazi se do pitanja je li moguće primijeniti strojno učenje na estimaciju temperature elektromotora, konkretnije rotora? Ukoliko je odgovor na to pitanje pozitivan, a dosadašnja istraživanja na to upućuju, onda je izuzetno važno ustvrditi je li moguće tu estimaciju dalje unaprijediti korištenjem nasumične pretrage hiperparametara, unakrsnom

validacijom i primjenom ansambl metoda. Odgovore na ove teze će se pokušati ponuditi kroz poglavlja ovog rada.

U idućem poglavlju pružit će se pregled postojećih istraživanja o tehnikama estimacije temperature, koje su problematike tih tehnika te u kojem smjeru idu aktualna istraživanja. Nadalje, istražit će se set podataka koji se koristi, veze između varijabli unutar seta podataka te koje su predložene metode za predobradu seta podataka. U četvrtom poglavlju opisana je najvažnija teorijska podloga, odnosno cjelovita metodologija istraživanja kroz više podpoglavlja. U petom poglavlju nalaze se rezultati koji se oslanjaju na sve obrađeno u prethodnom poglavlju, a nakon čega slijedi poglavlje u kojem se diskutira o predstavljenim rezultatima. Zatim slijedi zaključak u kojem su izloženi objedinjeni zaključci dobiveni analizom rezultata te osvrt na postavljene teze iz uvoda.

2. PREGLED POSTOJEĆIH ISTRAŽIVANJA

Poznavanje radne temperature u stvarnom vremenu važno je u sustavu upravljanja kako bi se namjestili parametri regulatora ovisni o temperaturi. Moguće je mjeriti temperaturu nepokretnih dijelova stroja sa senzorom temperature kontaktnog tipa, ali kod rotora je dodatan izazov zbog posebnosti strukture i naravno rotacije. Dostupne su metode bežične telemetrije, ali su skupe i utječu na robusnost stroja. Općenito, izravno mjerenje temperature je neizvedivo u automobilskoj primjeni zbog poteškoća kod održavanja ili zamjene ugrađenih senzora temperature u slučaju kvara. Kroz povijest su predložene različite tehnike procjene temperature koje su se razvijale desetljećima. Među najuspješnijima su toplinske mreže s grupiranim parametrima (eng. LPTNs – *Lumped parameter thermal networks*) koje označavaju ekvivalentne mreže nalik električnim krugovima. LPTN se sastoje od niza povezanih čvorova, gdje svaki čvor predstavlja komponentu s određenim toplinskim otporom i kapacitetom te se s njima lako aproksimiraju unutarnji prijenosi topline na temelju termodinamičke teorije [2-7]. Na slici 2-1 se nalazi termalni element koji je bazični blok LPTN-a.



Slika 2-1 Termalni element LPTN-a [7]

Metoda je neinvazivna, ne zahtjeva dodatni hardver te se postiže visoka točnost s relativno malo parametara modela. Međutim, za projektiranje LPTN-a je potrebno stručno poznavanje geometrije motora i sustava hlađenja. Postoje i tehnike estimacije temperature koje koriste električne parametre koji su temeljeni na obzerverima magnetskog toka (eng. *Wave flux observers*) te ubrizgavanje visokofrekventnog signala (eng. *High-frequency signal injection*). Obzerveri magnetskog toka estimiraju temperaturu analizom magnetskog toka u zračnim

procjepima, gdje je osnovno načelo da se magnetske karakteristike motora mijenjaju s temperaturom, utječući na distribuciju magnetskog toka. Mjerenjem napona i struja statora, zajedno s brzinom motora i drugim radnim parametrima, obzerver procjenjuje temperaturu rotora na temelju varijacija u magnetskom toku. Ova metoda zahtijeva točno poznavanje parametara motora i pažljivu kalibraciju kako bi se uzele u obzir neidealnosti i nesigurnosti. Ubrizgavanje visokofrekventnog signala uključuje ubrizgavanje visokofrekventnog signala u namote motora i analiziranje odziva kako bi se izvukle informacije o električnim i toplinskim karakteristikama motora. Ova se metoda temelji na činjenici da se impedancija namota motora mijenja s temperaturom. Mjerenjem promjena impedancije uzrokovanih ubrizganim signalom, metoda može procijeniti otpor namota motora, koji se pak može povezati s temperaturom. Ovaj pristup zahtijeva preciznu mjernu opremu i pažljive tehnike obrade signala za točno izdvajanje relevantnih informacija. Nedostatci navedenih metoda su što je potrebno opsežno poznavanje konstrukcijskih parametara elektromotora te velika stručnost [2–7].

Nedavni napredak strojnog učenja potaknut je razvojem novih algoritama učenja, teorije umjetne inteligencije te dostupnosti podataka i računala koji te podatke mogu obraditi. Strojno učenje je svoju primjenu našlo u svim sferama života, a posebice u znanosti i tehnologijama. Pristup neuronske mreže (eng. NN – *neural network*) je neovisan o informacijama o motoru jer je prilagođen isključivo empirijskim mjerenjima te iz tog razloga ne trpi od degradacije procjene ukoliko su pretpostavke fizičkog modela pogrešne. Uvođenje strojnog učenja prilikom estimacije temperature u ugrađenim sustavima, naravno, dolazi i s novim izazovima. Za razliku od npr. LPTN, mnogi hiperparametri zajedno sa stvarnim parametrima modela određuju ishod treninga. Oni se moraju namjestiti kako se ne bi izgubila autonomija samoprilagodbe za razliku od „dizajna stručnjaka“. Nadograđivanje modela poboljšava njegovu estimaciju, ali zahtijeva dodatne računalne resurse te duže traje [4].

Temelj ovog rada daju uspješne estimacije temperature iz znanstvenog članka autora Kirchgässner i Wallscheid [4] na kojem se zasniva korišteni set podataka te istraživanja sa stranice *Kaggle* s kojih je set podataka preuzet [8]. Korištene metode i metrike evaluacije variraju od rada do rada te su prisutni drugačiji pristupi problemu, stoga nema relevantnih i lako usporedivih metrika s ovim radom, ali tema je itekako aktualna i upravo primjena strojnog učenja će zasigurno dovesti do novih postignuća u estimaciji temperature PMSM.

3. OPIS SETA PODATAKA I PREDOBRADA

Točna estimacija temperature ima golemu važnost za osiguranje optimalnih performansi, dugovječnosti i sigurnosti PMSM-a. Međutim, prije nego što se krene s estimacijom temperature, neophodno je predstaviti alate koji će se koristiti u radu te kroz poglavlje napraviti analizu mjerenih podataka, što čini početnu točku ovog istraživanja. Nakon što se predstavi programski jezik Python i ostali bitni alati, opisat će se korišteni set podataka (eng. *Dataset*) jer cijeli rad počinje od razumijevanja seta podataka. Podaci prikupljeni od PMSM-a su složeni i obuhvaćaju različite varijable koje pridonose temperaturnim fluktuacijama. Detaljan opis seta podataka nudi uvid u prirodu varijabli, jedinice i potencijalne izazove, postavljajući temelje za daljnji rad. Budući da estimacija temperature uključuje mnoštvo čimbenika, istraživanje međuodnosa između varijabli je vrlo bitno. Korelacijska analiza daje uvid koje značajke su usko povezane i koje bi mogle međusobno utjecati. Otkrivanje tih odnosa omogućuje donošenje informiranih odluka o modeliranju estimatora temperature. Mjereni podaci često sadrže odstupanja i greške u mjerenjima. Ove netipične vrijednosti (eng. *Outliers*) mogu negativno utjecati na analizu i performanse modela. Rješavanje netipičnih vrijednosti jedan je od koraka u osiguravanju robusnosti estimacije temperature, a razmotrene su metode *Outlier Handling* iz *Feature-engine*.

Skaliranje seta podataka na zajedničku skalu je možda najvažniji korak. Skaliranje osigurava da ulazne varijable estimatora ravnomjerno doprinose procesu estimacije temperature, bez obzira na njihove izvorne jedinice ili veličine. Odnosno, određeni modeli regresije neće niti dati točne rezultate ukoliko set podataka nije skaliran. Istražen je značaj *StandardScalera* i *PowerTransformera*, a rezultati estimacije uspoređivani korištenjem oba.

3.1. Python

Za istraživanja potrebna za izradu ovog rada korišten je programski jezik Python, koji je poznat po svojoj svestranosti i mnoštvu biblioteka (eng. *Library*) potrebnih za podatkovnu znanost (eng. *Data Science*), znanstveno računalstvo, strojno učenje i analitiku. Python se stoga nameće kao glavni alat istraživača diljem svijeta. Bogat sustav biblioteka, poput *Numpy* [9], *Pandas* [10], *Matplotlib* [11], *Seaborn* [12] i drugih, uvelike pojednostavljuju manipulaciju podacima, analizu i vizualizaciju. Zatim, biblioteka *Scikit-Learn* [13,14] koja pruža obilje alata za razvoj

modela strojnog učenja. Popularnost Pythona je dovela do jakog razvoja zajednice koja radi u Pythonu te je dostupno mnoštvo resursa koji olakšavaju rad.

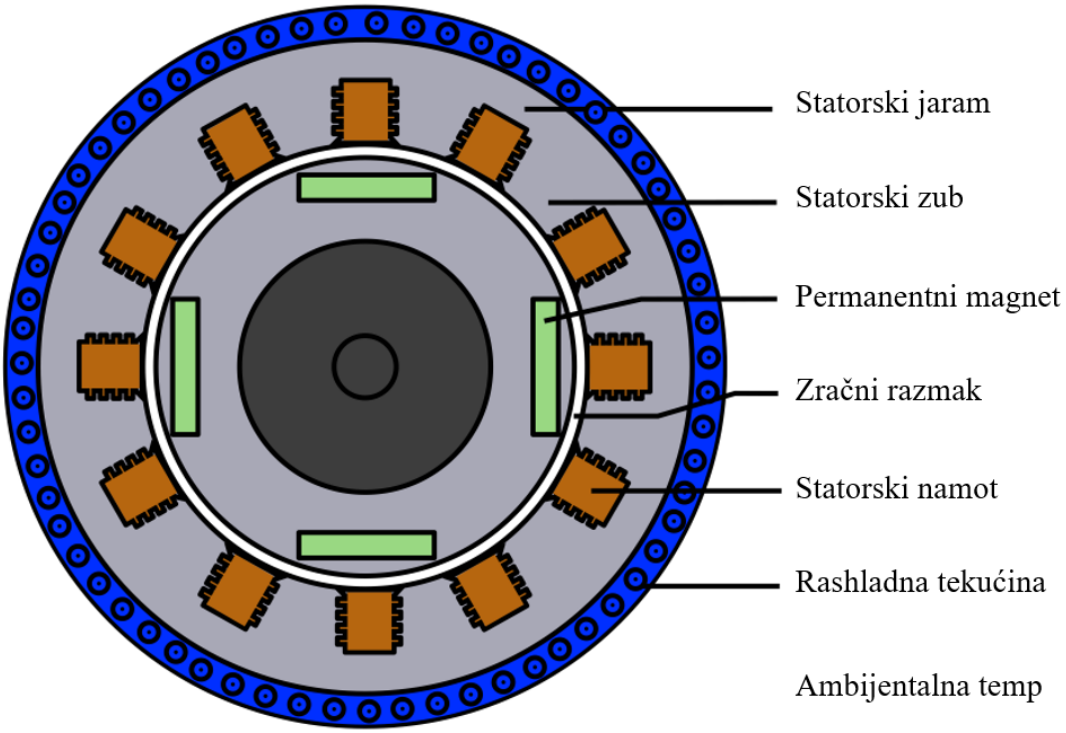
Korištena je Anaconda [15], podatkovna znanstvena platforma otvorenog koda kao ulaz u ekosustav Pythona. Anaconda olakšava upravljanje Python paketima i bibliotekama te je jako kompatibilna. Konkretno, unutar Anaconde korišten je Spyder [16], besplatno znanstveno okruženje otvorenog koda (eng. *Open source*) napisano u Pythonu za korištenje Pythona, a dizajniran je od strane znanstvenika, inženjera i analitičara podataka upravo za njih. Sadrži kombinaciju naprednog uređivanja, analize, ispravljanja pogrešaka i funkcionalnosti potrebne za rad poput ovog. Scikit-Learn ili skraćeno sklearn, je biblioteka strojnog učenja koja pruža jednostavne i učinkovite alate za analizu podataka i modeliranje. Prednosti su dosljedno sučelje za razne algoritme, što olakšava eksperimentiranje s različitim modelima, jednostavne funkcije za obradu i procjenu modela, te odličnu i dostupnu dokumentaciju.

3.2. Set podataka

Set podataka [4,8] sastoji se od nekoliko podataka senzora prikupljenih s trofaznog automobilskog PMSM-a snage 52 kW postavljenog na testnom stolu. Mjerenja su prikupljena na LEA odjelu na Sveučilištu Paderborn. Set podataka sastoji se od 185 sati multivarijantnih mjerenja uzorkovanih na 2 Hz. Motor je kontroliran zakretnim momentom, dok je njegova brzina određena brzinom motora s kontrolom opterećenja. Temperature su mjerene ugrađenim termoparovima, a temperatura rotora predstavljena je prosječnom površinskom temperaturom PM (eng. *Permanent magnet*) na četiri senzora. Set podataka ima 13 stupaca i svaki od njih predstavlja varijablu, dok svaki uzorak (eng. *Sample*) (1330816) predstavlja jednu snimku podataka senzora u određenom vremenskom koraku. Budući da su podaci uzorka 2 Hz, pojavljuje se jedan redak svakih 0,5 s [4].

Sve varijable su tipa *float*, osim *profile_id* koji je tipa *intiger*. U tablici 1 se nalaze sve varijable navedene poimence uz osnovnu statičku analizu. PMSM motor se upravlja d-q metodom [17] pa naponi *u_q* i *u_d* predstavljaju d i q komponentu napona u d-q sustavu u voltima, slično tome, *i_q* i *i_d* predstavljaju struju u amperima. Brzina motora (eng. *Motor speed*) prikazana je u okretajima u minuti, a moment u njutnmetrima. Sve ostale varijable su temperature različitih dijelova motora u Celzijevim stupnjevima. Slika 3-1 prikazuje poprečni presjek

PMSM motora na kojem su istaknuti svi ključni dijelovi motora koji se odnose i na varijable seta podataka, odnosno temperature tih dijelova. Krenuvši izvana, dolazi varijabla *ambient* što označava ambijentalnu temperaturu ili temperaturu okoline PMSM-a. Zatim, rashladni kanal u kojem se nalazi rashladna tekućina (eng. *Coolant*) što dovodi do statora. Na statoru su istaknuti statorski jaram (eng. *Stator yoke*), statorski zub (eng. *Stator tooth*) te statorski namot (eng. *Stator winding*). Između statora i rotora se nalazi zračni razmak (eng. *Air gap*) te konačno PM koji su postavljeni na obim rotora. Opisano je 12 varijabli, a preostao je *profil_id* koji označava sesiju mjerenja podataka.

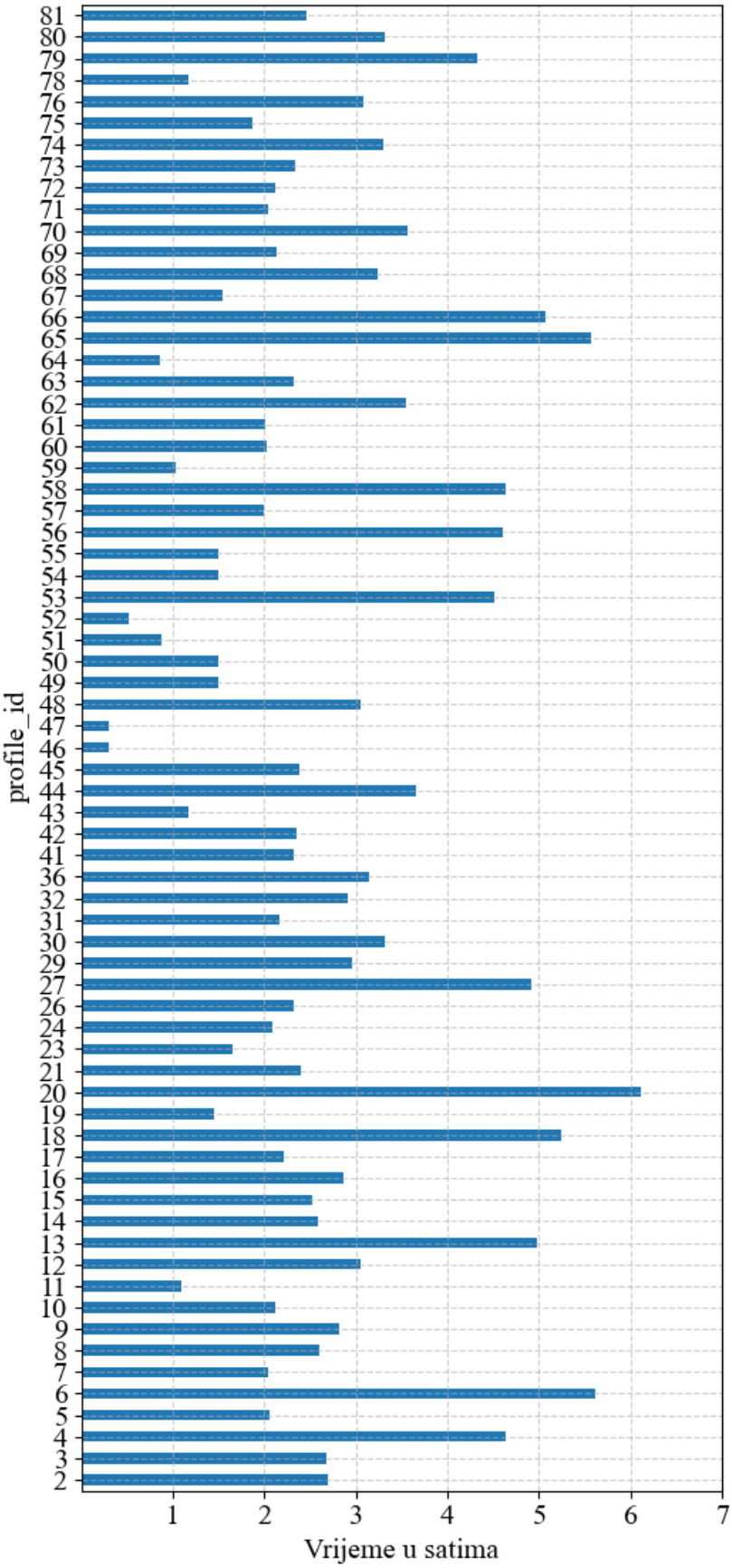


Slika 3-1 Poprečni presjek PMSM [7]

Na slici 3-2 se nalazi prikaz iz kojeg se vidi koliko je podataka prikupljeno u određenoj sesiji, odnosno preračunato je u sate mjerenja zbog lakše vizualizacije. Vidljiva je velika razlika između trajanja određenih mjerenja s čime su u setu podataka zastupljena razna stanja motora.

Tablica 1 Statistička analiza parametara

Varijabla	Jedinica	Minimum	Maksimum	Srednja vrijednost	Standardna devijacija
U_q	V	-25.3	133	54.3	44.2
Coolant	°C	10.6	102	36.2	21.8
Stator winding	°C	18.6	141	66.3	28.7
U_d	V	-132	131	-25.1	63.1
Stator tooth	°C	18.1	112	56.9	23
Motor speed	o/min	-276	6000	2200	1860
I_d	A	-278	0.05	-68.7	64.9
I_q	A	-293	302	37.4	92.2
pm	°C	20.9	114	58.5	19
Stator yoke	°C	18.1	101	48.2	20
Ambient	°C	8.78	30.7	24.6	1.93
Torque	Nm	-246	261	31.1	77.1

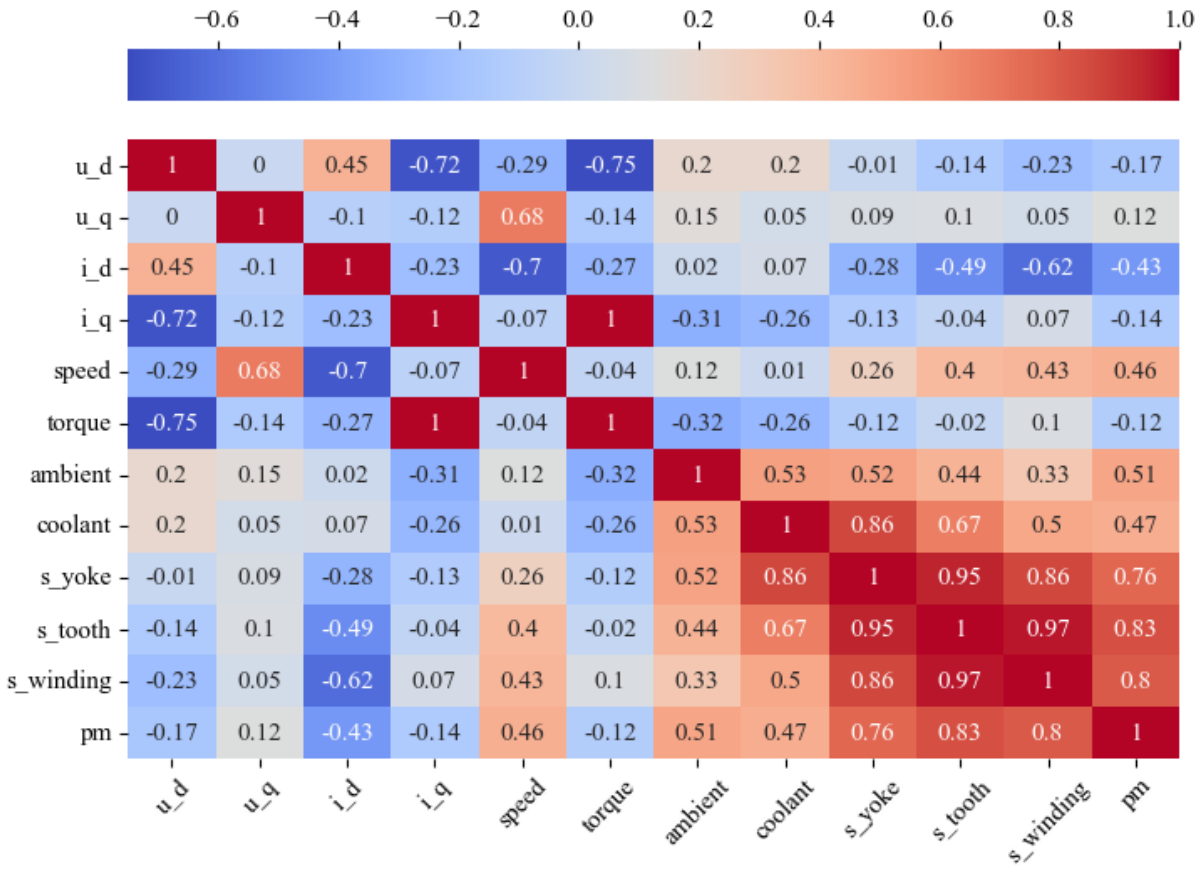


Slika 3-2 Količina podataka u pojedinom profilu

3.3. Korelacija

Sljedeći korak u analizi seta podataka je korelacija. Korelacija je statistička mjera koja izražava snagu odnosa između dvije varijable. Kreće se od -1 do +1, gdje se pozitivna korelacija javlja kada se dvije varijable kreću u istom smjeru, a negativna kada se kreću u suprotnom smjeru. Pozitivna korelacija znači da kada jedna varijabla raste, druga također ima tendenciju povećanja, a za negativnu korelaciju prilikom rasta jedne varijable, druga ima tendenciju pada. Koeficijent korelacije približno 0 sugerira slab ili nikakav linearni odnos između varijabli. Važno je koristiti korelaciju za određivanje uzročno-posljedične veze između varijabli u setu podataka [18,19].

Odličan način za jednostavno predočiti korelaciju varijabli seta podataka je korelacijski *heatmap* koji vizualno sažima korelaciju između više varijabli pomoću boja. Koristeći *Seaborn* [12] napravljen je korelacijski *heatmap* sa slike 3-3. Korelacija se brzo detektira bojom, gdje toplije boje (crvena) označavaju pozitivnu korelaciju, a hladnije (plava) negativnu korelaciju. Konkretna brojčana korelacija je istaknuta u obojanom pravokutniku. *Heatmap* je raspoređen u obliku kvadratne matrice, gdje redovi i stupci odgovaraju varijablama koje se analiziraju. Cilj ovog rada je estimacija temperature elektromotora, a analizom korelacije se može zaključiti da je ona moguća. Poseban interes je estimacija parametra pm, odnosno temperatura rotora elektromotora, koja teško izvediva zbog činjenice da se radi o samoj unutrašnjosti motora, i to o rotoru. Sve temperaturne varijable imaju relativno visoku korelaciju što je odličan indikator da bi estimacija varijable pm pomoću ostalih varijabli mogla biti uspješna. Pogotovo su visoke korelacije raznih dijelova statora u odnosu na rotor pm. Od ostalih korelacija vidljiva je najveća korelacija parametara *torque* i *i_q*, što je očekivano jer se upravljanjem PMSM u d-q upravljanju momentom upravlja strujom u q osi. Također, korelacija je izračunata korištenjem Pearsonovih i Spearmanovih korelacijskih koeficijenata koji su dali identične rezultate.



Slika 3-3 Korelacijski heatmap

3.4. Predobrada seta podataka

Kako bi regresijski modeli estimacije temperature pm dali čim bolje rezultate, potrebno je napraviti određene obrade seta podataka. Zadatak predobrade je ukloniti ekstreme i potencijalne greške u mjerenju te ujednačiti utjecaj varijabli seta podataka. Također je potrebno detektirati nulte vrijednosti i eventualne nepostojeće podatke (eng. *Missing Values*). Tokom istraživanja za ovaj rad razmotreno je nekoliko metoda uklanjanja netipičnih vrijednosti te dvije metode skaliranja podataka. Kako bi se utjecaj uklanjanja netipičnih vrijednosti i skaliranja podataka mogao usporediti, korišten je višeslojni receptor (MLP, eng. *Multilayer Perceptor*) te metrike koeficijent determinacije (R^2, eng. *Coefficient of Determination*), srednja apsolutna pogreška (MAE, eng. *Mean Absolute Error*) i prosječna kvadratna pogreška (RMSE, eng. *Root Mean Squared Error*) koje su detaljno objašnjene u poglavlju 4.

3.4.1. Netipične vrijednosti

Netipične vrijednosti su podaci unutar seta podataka koji značajno odstupaju od ostatka vrijednosti te mogu dovesti do pogrešnih zaključaka i oslabiti izvedbu modela koji će se koristiti u daljnjem radu. Netipična vrijednost može biti uzrokovana ljudskom greškom u procesu sakupljanja podataka i njihovom spremanju, greškama u mjerenjima te smetnjama. Međutim, potrebno je imati određena znanja o setu podataka i prirodi procesa čiji su podaci prikupljeni jer nisu sve netipične vrijednosti za uklanjanje, već postoji mogućnost da zaista predstavljaju legitimnu vrijednost [20,21].

Za analizu netipične vrijednosti korišten je *distplot* i *boxplot* iz seaborn biblioteke. Distplot je skraćenica za distribucijski graf te pruža uvid u distribuciju varijable u setu podataka. Koristi se za vizualizaciju oblika, središnje tendencije, širenja podataka te prisutnosti maksimuma. Na distplotu su vidljivi histogram i dijagram gustoće, što omogućuje sveobuhvatan pregled karakteristika distribucije podataka. Histogram je stupčasti dijagram koji dijeli raspon podataka u intervale (eng. *bin*) i prikazuje učestalost ili broj podatkovnih točaka koje spadaju u svaki interval. Dijagram gustoće (eng. KDE – *Kernel Density Estimation*) prikazuje uglađenu procjenu funkcije gustoće vjerojatnosti podataka. Kod promatranja distplota bitno je razmotriti oblik, središnju tendenciju, širenje i varijabilnost te netipične vrijednosti [22].

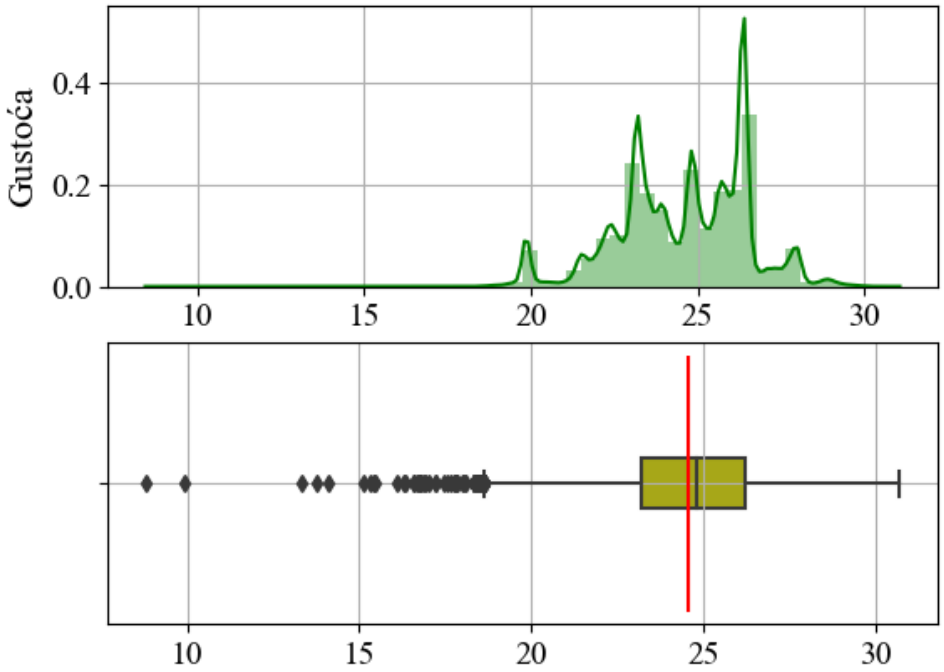
Druga vrsta grafa za analizu je *boxplot*, također poznat kao *box-and-whisker* plot koji pruža uvid u distribuciju seta podataka, a posebno u pogledu njegove središnje tendencije, širenja i prisutnosti netipičnih vrijednosti. Distribucija je prikazana na sažet i vizualan način te je moguće lako identificirati odstupanja u podacima. Kutija (eng. *box*) je središnji pravokutnik dijagrama te predstavlja interkvartilni raspon (IQR, eng. *interquartrile range*) podataka. Proteže se od prvog kvartila (Q1) do trećeg kvartila (Q3), pokrivajući srednjih 50% podataka. Linija unutar okvira predstavlja medijan ili srednju vrijednost, a označava središnju tedenciju podataka te označava drugi kvartil (Q2). Brkovi (eng. *Whiskers*) se protežu od okvira prema van do minimalnih i maksimalnih vrijednosti unutar određenog raspona. Sve točke izvan brkova se smatraju potencijalnim netipičnim vrijednostima [23].

U ovome radu korišten je *Feature-engine* koji pruža tri transformacije za rješavanje problema s netipičnim vrijednostima, a to su: *Winsorizer*, *ArbitraryOutlierCapper* i *OutlierTrimmer* [24]. Metoda *Winsorizer* rješava problem netipične vrijednosti ograničavanjem na definirani prag.

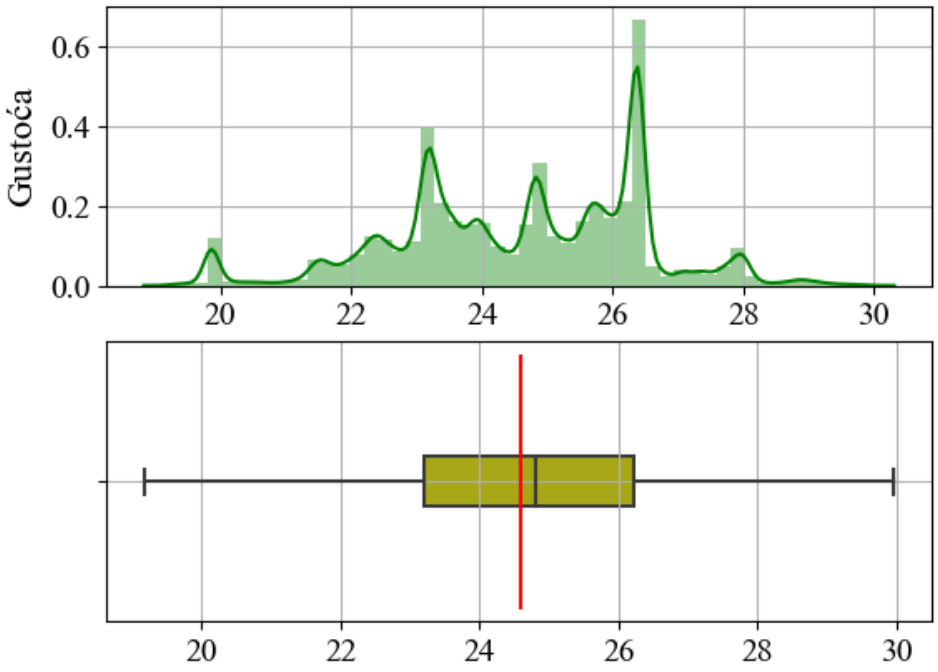
Ova metoda pomaže u održavanju integriteta distribucije podataka dok minimizira utjecaj ekstremnih vrijednosti. To je učinkovita tehnika za riješavanja problema netipičnih vrijednosti bez njihovog potpunog uklanjanja. *ArbitraryOutlierCapper* nudi fleksibilnost zamjene netipičnih vrijednosti proizvoljnim vrijednostima koje definira korisnik. Ova metoda nije korištena, a tipično je koriste stručnjaci koji su specijalizirani za vrstu podataka s kojom se rukuje te posjeduju znanja o ponašanju vrijednosti koje su detektirane kao netipične vrijednosti. Za razliku od metoda koje ograničavaju ili zamjenjuju netipične vrijednosti, *OutlierTrimmer* ih uklanja iz skupa podataka ovisno o postavkama. U kontekstu estimacije temperature, gdje ekstremne vrijednosti mogu predstavljati pogrešna očitanja ili prolazne uvjete, uklanjanje netipičnih vrijednosti može dovesti do čišćeg i točnijeg skupa podataka za treniranje modela. Odluka o odabiru metode *OutlierTrimmer* donesena je usporedbom rezultata treniranja modela regresora nakon korištenja navedenih metoda. *OutlierTrimmer* dao je minimalno bolje rezultate treniranja, međutim, kada se uzme u obzir veličina početnog seta podataka od 1330816, uklanjanje nekoliko desetaka tisuća podataka je zanemarivo, a konačni set podataka je manji (brže treniranje) i sadrži reprezentativnije podatke.

Na slici 3-4 prikazani su dosad opisani *distplot* i *boxplot* varijable *ambient*. Iz slike je odmah vidljivo da postoji nekoliko potencijalnih netipičnih vrijednosti s lijeve strane, a analizom podataka se ustvrđuje da doista postoji samo nekoliko uzoraka s ambijentalnom temperaturom nižom od 18 °C. To je automatski sumnjivo jer su se podaci senzora uzimali svakih 0.5 s te bi svakako bilo nekoliko stotina sličnih uzoraka da su to realni podaci temperature okoline. Za primjer je uzeta varijabla *ambient* jer je vrlo vidljiva razlika u odnosu na sliku 3-5 na kojoj su uklonjene netipične vrijednosti. Kod većine drugih varijabli nije bilo toliko netipičnih vrijednosti, osim kod varijabla *torque* i *i_q,* što je za očekivati jer struja u q osi dostiže velike vrijednosti prilikom pokretanja motora koje su nekoliko puta veće od nazivne vrijednosti, a kao što je spomenuto, momentom motora se upravlja upravo strujom u q osi pa se događa identična stvar. Te vrijednosti se dostižu samo nekoliko sekundi prilikom zaleta motora te ne predstavljaju realne vrijednosti prilikom dugotrajnog rada motora, ali je konačno zadržan dio tih vrijednosti što će biti objašnjeno u idućem potpoglavlju.

Prilikom analize netipičnih vrijednosti provjerena je i prisutnost nepostojećih podataka te je ustvrđeno da nema takvih podataka u ovom setu podataka.



Slika 3-4 Prisutne netipične vrijednosti



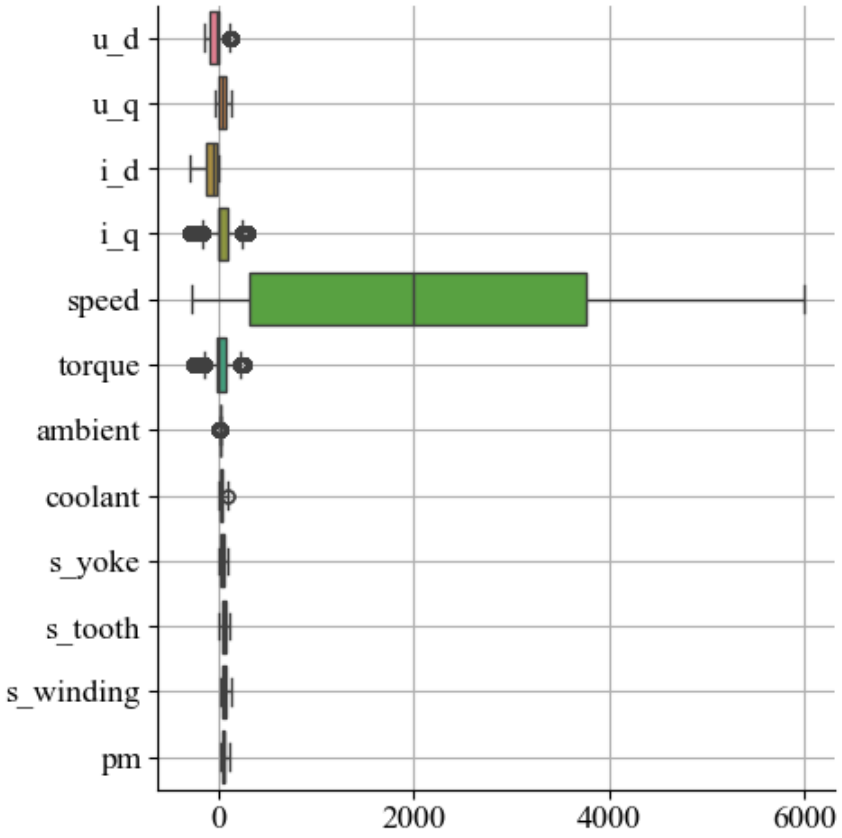
Slika 3-5 Uklonjene netipične vrijednosti

3.4.2. Skaliranje

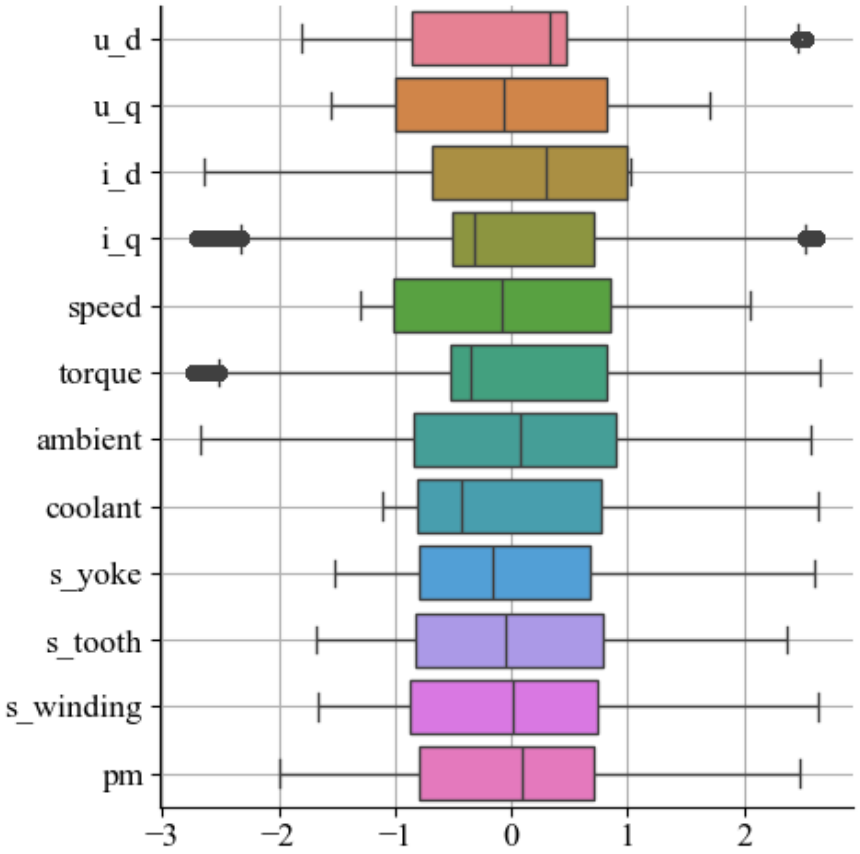
Kao što je već spomenuto, predobrada ulaznih podataka igra ključnu ulogu u postizanju točnih i pouzdanih rezultata modela estimacije. Idući korak značajno utječe na izvedbu modela strojnog učenja, odnosno neki modeli neće ni raditi ako set podataka nije skaliran. U ovom dijelu se raspravlja o upotrebi metoda *StandardScaler* [25] i *PowerTransformer* [26] za skaliranje seta podataka u kontekstu rada. Procjena temperature PMSM-a zahtijeva integraciju različitih očitanja senzora, radnih uvjeta i karakteristika motora. Ove varijable obuhvaćaju različite jedinice i skale, što dovodi do potencijalnih pristranosti prilikom treniranja. Skaliranje rješava ovaj problem pretvaranjem varijabli u usporedive ljestvice, osiguravajući da svaka varijabla ujednačeno doprinosi procesu učenja. Nadalje, skaliranje je bitno za algoritme koji se oslanjaju na metriku udaljenosti, gradijente ili regularizaciju, budući da sprječava pretjerani utjecaj varijabli s većim veličinama [27].

StandardScaler standardizira varijable tako da imaju nultu srednju vrijednost i jediničnu varijancu, učinkovito centrirajući podatke oko nule i prilagođavajući širenje na jedinstvenu ljestvicu. Korištenjem *StandardScalera* postiže se ujednačenost u skalama varijabli, što dovodi do poboljšanih stopa konvergencije, poboljšane stabilnosti modela i konačno točnijih estimacija temperature. Druga značajna korištena metoda skaliranja i transformacije podataka je *PowerTransformer*. Ova metoda ima za cilj približiti distribuciju podataka Gaussovom obliku primjenom transformacija snage. *PowerTransformer* može biti koristan za rješavanje problema asimetrije i čini podatke podložnijim pretpostavkama normalnosti [25,26].

Na slici 3-6 nalazi se *boxplot* svih varijabli iz seta podataka, te je vidljivo koliko se ističe brzina motora jer set podataka nije skaliran. Da se nastavilo sa istraživanjem bez skaliranja varijabli brzine motora, toj varijabli dao bi se preveliki značaj i rezultati estimacije bili bi drastično lošiji. Na slici 3-7 se nalazi *boxplot* nakon uklanjanja netipičnih vrijednosti i skaliranja sa StandardScaler, varijable su na zajedničkoj skali te imaju centralnu tendenciju što je bitno za daljnji rad. Još uvijek se javljaju točke kao potencijalne netipične vrijednosti kod varijabli *i_q* i *torque,* međutim, najekstremnije vrijednosti su uklonjene, a daljnjim podešavanjem *trimmera* bi se uklonile korisne vrijednosti drugih varijabli. Mišljenje autora je da je u redu ostaviti dio tih vrijednosti kako bi u modelu bio prisutan trenutak pokretanja motora.



Slika 3-6 Set podataka prije obrade



Slika 3-7 Set podataka nakon obrade

4. METODOLOGIJA RADA

Ideja ovog rada je istražiti mogu li metode strojnog učenja precizno estimirati temperaturu rotora (pm) PMSM-a. Procedura se može sumirati u sljedećim koracima inspirirana radovima [28–30]:

- inicijalno istraživanje na setu podataka koristeći metode strojnog učenja (regresore) sa zadanim parametrima kako bi se odabralo najbolje metode za nastavak istraživanja,
- s odabranim regresorima se zatim preko nasumične pretrage hiperparametara (eng. *Random Hyper-parameter search*) s peterostrukom unakrsnom validacijom (eng. *5-fold cross validation*) traže najbolji hiperparametri regresora za ostvarivanje najbolje estimacije, i konačno
- najbolji regresori s najboljim hiperparametrima se kombiniraju u ansambl koji metodom glasanja (eng. *Voting*) određuje estimacije s ciljem dobivanja još boljih modela za estimaciju temperature pm.

Kako bi estimacija temperature rotora PMSM bila točna potrebno je primijeniti robusne regresijske tehnike. Ovo poglavlje se bavi konstrukcijom i evaluacijom regresijskih modela za estimaciju temperature, obuhvaćajući ključne elemente poput optimizacije hiperparametara, podjele seta podataka i validacije, metrike evaluacije te finalno metode ansambla. Rezultati svih koraka istraživanja bit će prikazani u poglavlju Rezultati.

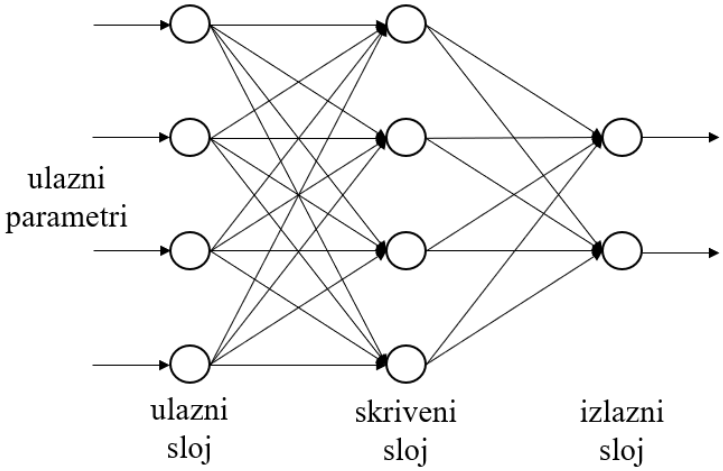
4.1. Regresijski modeli

Regresija je tehnika strojnog učenja koja se koristi za modeliranje i analizu odnosa između zavisne varijable (cilj) u ovom slučaju pm te jedne ili više nezavisnih varijabli. Primarni cilj regresijske analize je razumjeti i kvantificirati utjecaj nezavisnih varijabli na cilj te napraviti estimaciju na tom temelju [18].

Kroz rad su isprobani mnogi regresijski modeli, međutim, dio njih nije dao dobre rezultate, stoga s njima nije nastavljen rad niti su detaljno opisani. U nastavku su opisani modeli i hiperparametri modela koji su dali dobre rezultate te su korišteni i u daljnjim koracima. Osim opisanih hiperparametara, svaki model ih ima još nekoliko koji nisu obrađeni jer nisu uzeti kao relevantni za ovu primjenu ili nisu imali utjecaja na rezultate.

4.1.1. Višeslojni perceptron (MLP)

Bitan regresijski model ovog rada je višeslojni receptor (MLP, eng. *Multilayer Perceptor*) koji je korišten prilikom evaluacije početnih razmatranja ovoga rada poput uklanjanja netipičnih vrijednosti i skaliranja. Odabran je jer je svestrana i moćna arhitektura neuronske mreže poznata po svojoj sposobnosti hvatanja složenih odnosa unutar podataka. MLP je umjetna neuronska mreža (eng. ANN – *Artificial Neural Network*) koja se sastoji od troslojnih tipova naseljenih neuronima i potpuno je povezana. Prvi sloj je ulazni sloj, jedan ili više skrivenih slojeva, te izlazni sloj [31]. Opisano je i prikazano na slici 4-1.



Slika 4-1 Skica MLP modela

Arhitektura MLP-a omogućuje mu učenje zamršenih obrazaca i nelinearnih odnosa prisutnih u podacima. Svaki neuron u mreži obrađuje informacije i prosljeđuje ih sljedećem sloju, u konačnici proizvodeći izlaz koji služi kao estimirana temperatura. Opisana je pojmom *feed-forward,* što znači da se ulazi kombiniraju s početnim težinama u ponderiranom zbroju i podvrgavaju funkciji aktivacije. Težine su koeficijenti koji predstavljaju zbrojene vrijednosti neurona u prethodnim slojevima i prenose se na sljedeći sloj. Glavno svojstvo je iterativna prilagodba težina (eng. *weights*) u mreži koja se naziva *backpropagation*. U svakoj iteraciji, nakon što se ponderirani iznosi proslijede kroz sve slojeve, izračunava se pogreška. Zatim se događa *backpropagation* i težine prvog skrivenog sloja ažuriraju se u skladu s pogreškom. Pogreška se zatim prosljeđuje mrežom, prilagođavajući težine i pristranosti kako bi se pogreška svela na najmanju moguću mjeru tijekom višestrukih iteracija. Opisani proces događa se sve dok se pogreška ne promijeni više od definiranog praga u usporedbi s prethodnom iteracijom. Ovaj iterativni proces fino podešava parametre modela kako bi poboljšao njegovu izvedbu.

Snage MLP-a leže u njegovoj sposobnosti da uhvati zamršene odnose te se prilagodi složenim setovima podataka [28, 30-32].

Treniranje MLP-a zahtijeva pažljivo razmatranje dizajna arhitekture i podešavanja hiperparametara. Svi hiperparametri, njihov raspon, zadane vrijednosti (eng. *default values*) te upotreba se nalaze u tablici 2. Za početna razmatranja su korištene zadane vrijednosti iz tablice hiperparametara, a kasnije se implementiralo nasumično pretraživanje parametara iz navedenog raspona. Prateći opise MLP regresora te opise njegovih parametara sa scikit-learn stranice [13,14,33] ustvrđeno je da se ne koriste uvijek svi hiperparametri, već to ovisi o odabranom *solveru,* stoga je dodan stupac „Upotreba“ u kojem se nalazi ta informacija.

Tablica 2 MLP hiperparametri

Hiperparametar	Donja granica	Gornja granica	Zadana vrijednost	Upotreba
Number of Hidden Layers	1	5	1	
Number of neurons	10	100	100	
activation	identity, logistic, tanh, relu		relu	
solver	lbfgs, sgd, adam		adam	
alpha	0.00001	0.1	0.0001	
learning_rate	constant, invscaling, adaptive		constant	sgd
learning_rate_init	0.0001	0.01	0.001	sgd ili adam
max_iter	1000	2000	200	
shuffle	True, False		True	sgd ili adam
max_fun	10000	20000	15000	lbfgs
beta_1	0.1	0.9999	0.9	adam
beta_2	0.1	0.9999	0.999	adam
n_iter_no_change	8	15	10	sgd ili adam

Hiperparametri u tablici su na engleskom jeziku jer su upravo tako definirani kao parametri regresora te se u kodu tako pozivaju, stoga će biti dodatno pojašnjeni. *Number of Hidden Layers* se odnosi na broj skrivenih slojeva te vezano na to *Nuber of neurons* koji se odnosi na broj neurona u određenom skrivenom sloju. Zbog lakšeg razumijevanja su podijeljeni na dva različita parametra, iako se unutar MLP regresora definiraju kao *hidden_layer_size* formata *tuple* te se iz zadanog primjera (100,) može zaključiti da se radi o jednom skrivenom sloju u kojem se nalazi 100 neurona. *Activation* se odnosi na aktivacijsku funkciju skrivenih slojeva te se preko nje izračunava izlaz svakog neurona u skrivenom sloju. Moguće aktivacijske funkcije su:

- *relu* (eng. *Rectified Linear Unit*) (zadana vrijednost) - funkcija rampe,
- *tanh* – tangens hiperbolni,
- *logistic* – sigmoidna funkcija, i
- *identity* – linearna funkcija.

Hiperparametar *alpha* kontrolira L2 regularizaciju (eng. *Weight decay*) na način da smanjuje težine te se koristi za sprečavanje prekomjernog prilagođavanja (eng. *Overfitting*). *Max_iter* određuje maksimalni broj iteracija, model se iterativno podešava dok ne konvergira ili se dosegne maksimalni broj iteracija. Do sada navedeni hiperparametri su korišteni za sve modele MLP regresora, a daljnji ovise o odabiru *solver* hiperparametra po uputama sa scikit-learn [33]. *Solver* specificira optimizacijski algoritam za treniranje neuronske mreže, gdje je cilj treniranja MLP-a minimizacija funkcije gubitaka (eng. *Loss function*) što mjeri razliku između predikcija modela i ciljane (istinite) vrijednosti. Mogući odabiri *solver*-a su:

- *sgd* (eng. *Stochastic Gradient Descent*) - stohastički gradijentni pad podešava parametre modela iterativno u smjeru koji smanjuje funkciju gubitaka, gradijent je usmjeren u smjeru najvećeg povećanja funkcije gubitaka, a sukladno tome za minimizaciju se pomiče suprotno od smjera gradijenta,
- *adam* (eng. *Adaptive Moment Estimation*) (zadana vrijednost) - varijanta SGD-a, a jedna od razlika je u tome što *adam* adaptira stopu učenja (eng. *Learning rate*). *Adam* je manje osjetljiv na odabir stope učenja i podešavanje hiperparametara te često brže konvergira, i
- *lbfgs* - kvazi-Newton optimizacijska metoda uz koju se veže hiperparametar *max_fun* kojim se određuje maksimalni broj pozivanja funkcije.

Stopa učenja se po uputama podešava samo za *sgd* iz razloga što *adam* ima ugrađenu tu funkcionalnost. Moguće opcije su:

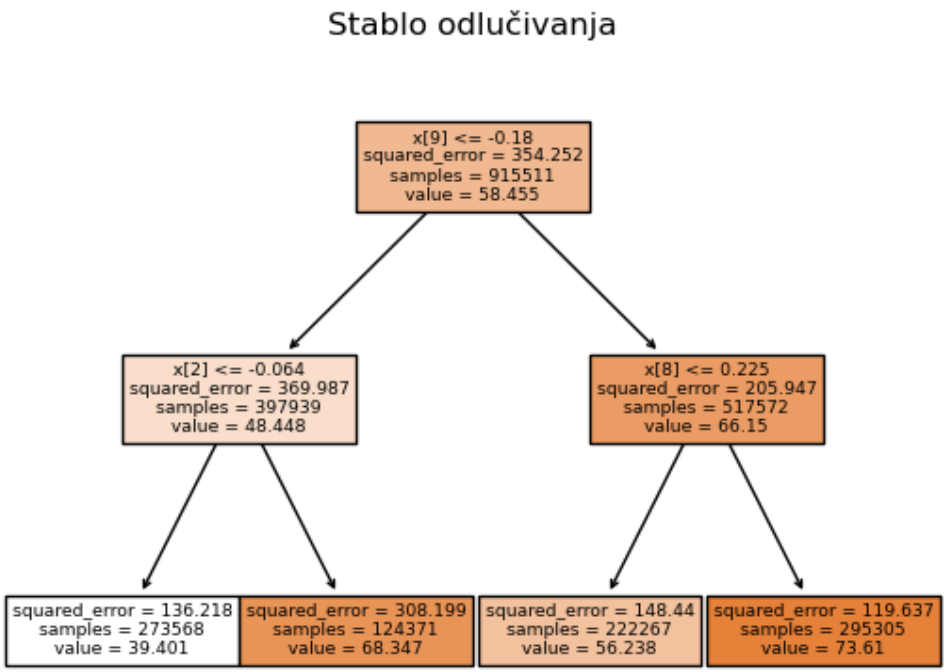
- *constant* – koristi se konstantna stopa učenja definirana hiperparametrom *learning_rate_init,*
- *invscaling* – postepeno se smanjuje stopa učenja, i
- *adaptive* – ukoliko se tokom treniranja funkcija gubitaka prestaje smanjivati, mijenja se stopa učenja.

Shuffle se određuje ukoliko će se podaci promiješati prilikom svake iteracije. Posljednji hiperparametri *beta_1* i *beta_2* se koriste samo za *adam*.

4.1.2. Stabla odlučivanja (DT)

Jedan od temeljnih alata u strojnom učenju su stabla odlučivanja ili DT (eng. *Decision Trees*), svestran i intuitivan model koji oponaša proces donošenja odluka stvaranjem stabla strukture odluka i ishoda. Svaki unutarnji čvor (eng. *Decision/Internal Node*) predstavlja odluku temeljenu na varijabli, što dovodi do grananja koje predstavljaju moguće ishode. Završni čvorovi, poznati kao listovi (eng. *Leaf/Terminal Node)*, daju konačna predviđanja. Proces konstruiranja stabla odlučivanja uključuje rekurzivno dijeljenje prostora varijabli na temelju onih najinformativnijih. Odabir varijabli i kriterija dijeljenja odvija se s ciljem maksimiziranja homogenosti ciljne varijable unutar svakog lista. Stabla odlučivanja imaju sklonost *overfitting*-a u treniranju, hvatanju šuma i lošim predikcijama na neviđenim podacima. Kako bi se to ublažilo, koristi se proces orezivanja (eng. *Pruning*) stabla. Orezivanje uključuje uklanjanje grana koje ne doprinose značajno performansama modela, što rezultira jednostavnijim i robusnijim stablom. Jedna od značajnih prednosti stabala odlučivanja je njihova interpretabilnost. Osim toga, stabla odlučivanja mogu se vizualizirati, pružajući uvid u granice odlučivanja i odnose između varijabli [32,34,35].

Upravo na slici 4-2 se nalazi stablo odlučivanja napravljeno za ovaj rad, s time da je limitirano maksimalnom dubinom kako bi podaci sa slike bili čitljivi.



Slika 4-2 Stablo odlučivanja

U tablici 3 se nalaze hiperparametri DT sa rasponom korištenim u daljnjem radu te zadanim vrijednostima.

Tablica 3 DT hiperparametri

Hiperparametar	**Donja granica**	**Gornja granica**	**Zadana vrijednost**
criterion	squared_error, friedman_mse, absolute_error, poisson		squared_error
splitter	best, random		best
max_depth	5	30	None
min_samples_split	2	10	2
min_samples_leaf	1	10	1
max_features	int, None		None

DT hiperparametrima se kontrolira struktura i ponašanje stabla odlučivanja, *criterion* determinira funkciju kojom se mjeri kvaliteta podjele na čvoru, a moguće opcije su:

- *squared_error* – određuje kvalitetu podjele na čvoru bazirano na srednjoj kvadratnoj pogrešci,
- *friedman_mse* – slično kao *squared_error*, ali inkorporira Friedmanovo vrednovanje,

- *absolute_error* – kvaliteta podjele na čvoru bazirana na srednjoj apsolutnoj pogrešci, i
- *poisson* – koristi Poissonovu devijaciju za pronalazak podjele čvorova.

Hiperparametar *splitter* određuje strategiju za podjelu na čvorovima, gdje *best* odabire najbolje podjele, a *random* nasumične podjele kako bi moguće smanjilo prekomjerno prilagođavanje. *Max_depth* limitira maksimalnu „dubinu" stabla odlučivanja, što je bitno da stablo ne postane prekompleksno i prekomjerno se prilagodi u treniranju. *Min_samples_split* je minimalni broj podataka potreban da bi se napravio unutarnji čvor, podešavanje ovog hiperparametra kontrolira veličinu listova i sprječava podjele čvorova koja bi „uhvatila" šum. *Min_samples_leaf* je minimalni broj podataka potreban da da bi se napravio lisni čvor što kontrolira veličinu listova. *Max_features* kontrolira maksimalan broj varijabli koje se analiziraju za svaku podjelu.

4.1.3. K-Nearest Neighbors (KNN)

KNN regresijski algoritam poznat je po jednostavnosti i prilagodljivosti, oslanjajući se na blizinu podatkovnih točaka za informiranu estimaciju. Za razliku od tradicionalnih parametarskih modela, KNN ne pokušava izgraditi odnos između varijabli, već se oslanja na ideju da će slične podatkovne točke u prostoru varijabli pokazivati slične ciljne vrijednosti varijable. Za danu podatkovnu točku njena ciljana vrijednost se može aproksimirati ispitivanjem ciljnih vrijednosti njegovih k-najbližih susjeda u prostoru varijabli. „K" predstavlja broj najbližih susjeda koje treba uzeti u obzir i to je hiperparametar koji se treba definirati te utječe na osjetljivost modela na lokalne varijacije. KNN regresija je konceptualno jednostavna i lako razumljiva, a nelinearana, što je odlično u stvarnim aplikacijama poput ove [32,36,37].

U tablici 4 se nalaze KNN hiperparametri zajedno sa zadanim vrijednostima te granicama koje su korištene u daljnjem istraživanju. *N_neighbors* hiperparametar određuje koliko se najbližih „susjeda" razmatra prilikom estimacije, odnosno definira veličinu susjedstva oko podatkovne točke. *Weights* kontrolira kako se ciljane vrijednosti najbližih susjeda kombiniraju u estimaciji, a može poprimiti vrijednosti:

- *uniform* (zadana vrijednost): svim susjedima su dodijeljene jedna težine, a estimacija je jednostavan prosjek ciljane vrijednosti susjeda, i

- *distance*: bliži susjedi imaju veći utjecaj te se njihovim ciljanim vrijednostima dodjeljuje težina inverzno proporcionalna udaljenosti od ciljane točke.

Algorithm specificira algoritam korišten za proračun najbližeg susjeda, a može poprimiti vrijednosti:

- *auto* (zadana vrijednost): algoritam automatski selektira najefektivniji algoritam ovisno o trening datasetu,
- *ball_tree*: koristi Ball Tree strukturu, efektivan za visokodimenzionalne podatke,
- *kd_tree*: koristi KD-Tree strukturu, efektivan za manje dimenzionalne podatke, i
- *brute*: koristi pretragu sirove snage, efektivan za male setove podataka.

Leaf_size je relevantan kada se koristi *ball_tree* ili *kd_tree* algoritam. Odlučuje u kojim točkama će algoritam prijeći na *brute* pretragu.

Tablica 4 KNN hiperparametri

Hiperparametar	Donja granica	Gornja granica	Zadana vrijednost
n_neighbors	1	50	5
weights	uniform, distance		uniform
algorithm	auto, ball_tree, kd_tree, brute		auto
leaf_size	10	50	30

4.2. Podjela seta podataka

Korak prije treniranja modela je podjela seta podataka na set za obuku/treniranje (eng. *Train*) i set za testiranje (eng. *Test*). Ovo potpoglavlje posvećeno je istraživanju važnosti pravilnog dijeljenja podataka u podsetove te kako će se to izvesti.

4.2.1. Podjela na set za treniranje i set za testiranje

Kada se ukupni set podataka ne bi dijelio na podsetove te kada bi se model strojnog učenja učio na cijelom setu podataka, previđanje na istom setu podataka bi dalo savršene rezultate. To bi se dogodilo jer bi model samo ponovio rezultate koje je već vidio, ali na neviđene, odnosno nove podatke ne bi uspio predvidjeti ništa korisno. Ova situacija se već spomenula kod opisa modela, a naziva se prekomjerno prilagođavanje te je uobičajena praksa kod strojnog učenja uvođenje testa modela na neviđene podatke. Set podataka se na samom početku dijeli na set za treniranje i set za testiranje (eng. *Train test split*) te se nakon treniranja modela strojnog učenja

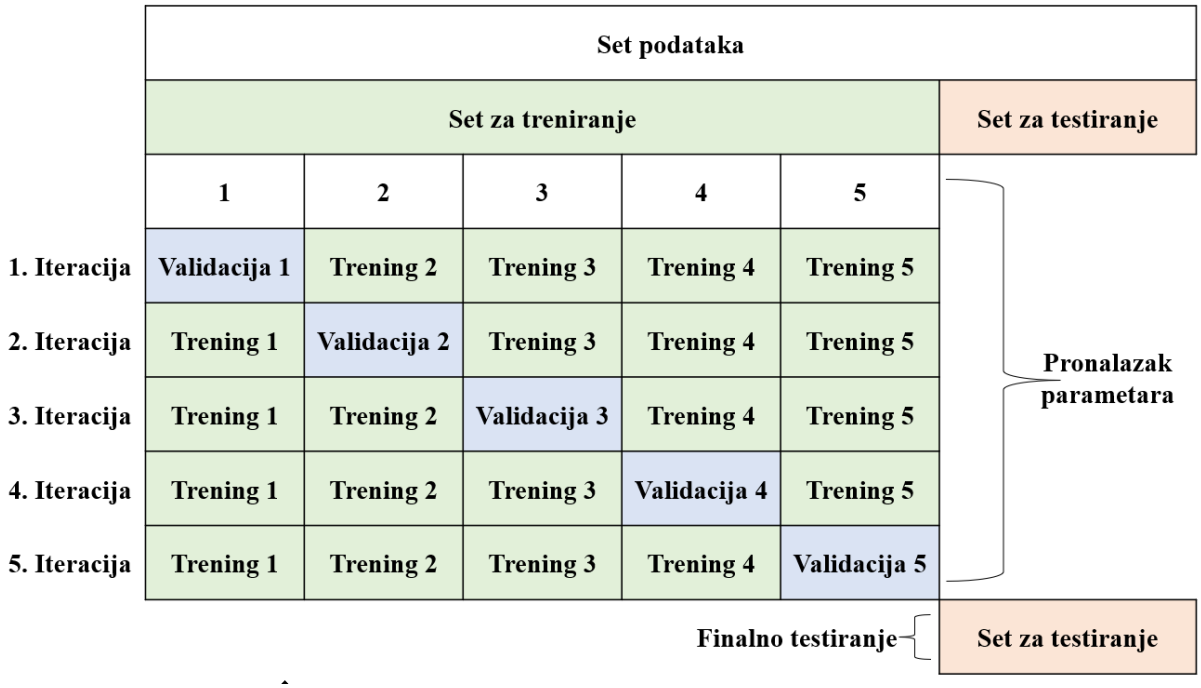
taj model evaluira na način da se provjeri kako estimira neviđene podatke. Model uči iz obrazaca, odnosa i struktura prisutnih iz seta za trening, dok je set za testiranje rezerviran za evaluaciju performansi modela i služi kao zamjena za nevidljive podatke iz stvarnog svijeta. To omogućuje da se procijeni koliko dobro model generalizira nove podatke koje do sada nije vidio. Podjela seta podataka pomaže da se uspostavi ravnoteža između pristranosti i varijance kod modela. Trening na većem dijelu podataka smanjuje pristranost (eng. *Bias*), dok estimacija na zasebnom setu za testiranje pomaže u kontroli varijance (eng. *Variance*). U praksi je uobičajena podjela 70-30 gdje je 70% seta podataka odabrano za treniranje, a 30% seta podataka za testiranje, odnosno evaluaciju modela. Na ovaj način se može realno usporediti performanse različitih modela ili varijacije jednog modela kako bi se odredilo koji je najprikladniji. Koristi se nasumična podjela seta podataka na set za treniranje i testiranje, odnosno podatkovne točke nasumično se dodjeljuju setovima za treniranje i testiranje. Ova strategija osigurava nepristran prikaz podataka u oba seta [32,38].

U ovom radu, *train-test split* služi kao temelj za procjenu izvedbe regresijskih modela. Omogućuje relevantnu procjenu koliko dobro modeli estimiraju temperaturu rotora na dosad neviđenim podacima, pružajući uvid o mogućnostima testiranog modela.

4.2.2. Unakrsna validacija

Prilikom evaluacije različitih postavki hiperparametara modela postoji rizik od pretjeranog prilagođavanja testnom setu jer se parametri mogu podešavati dok model u konačnici ne da dobre rezultate. Na taj način postoji opasnost od curenja podataka (eng. *Data Leakage)* što je pojava kada znanje o testnom setu može „procuriti“ u model, a tada metrika evaluacije više ne daje realnu ocjenu performansi modela. Da bi se riješio ovaj problem, koristi se unakrsna validacija (CV, eng. *Cross Validation*) odnosno „*k-fold CV“*, gdje k označava broj podjela unutar seta za treniranje. U ovom radu se koristi '5-fold CV', što znači da se set za treniranje dijeli na 5 jednakih dijelova, od kojih će se k-1, odnosno 4 koristiti za treniranje. Treniranje će sada trajati duže i u pet iteracija, ali će evaluacija modela biti realnija. U prvoj iteraciji će prvi dio seta za treniranje biti set za validaciju, a na ostala četiri dijela će se istrenirati model koji će se nakon treniranja validirati na odvojenom prvom dijelu. Postupak se ponavlja kroz pet iteracija gdje se izmjenjuje dio koji se koristi za validaciju, te se na kraju izračuna srednja vrijednost validacije tokom treniranja. Na taj način se dobiju metrike evaluacije za treniranje,

a konačno se model testira neviđenim setom za testiranje te se dobiju metrike evaluacije za testiranje. Opisani postupak treniranja, validacije i testiranja je vidljiv na slici 4-3[32,38].



Slika 4-3 Unakrsna validacija

Ovaj pristup zahtjeva više procesorske snage, duže traje, ali rezultati su relevantniji, a i dalje se koristi jednak dio seta podataka za učenje, što je bolje nego odvajanje dodatnog seta za validaciju.

Tokom rada su korištene dvije funkcije za unakrsnu validaciju:

- *sklearn.model_selection.cross_val_score* [39] – funkcija dizajnirana za brzu i generalnu evaluaciju performansa modela izračunom jedne metrike evaluacije, vraća jednu vrijednost koja predstavlja tu metriku, i
- *sklearn.model_selection.cross_validate* [40] – funkcija dizajnirana za prikupljanje više metrika evaluacije performansa modela, vraća rječnik (eng. *Dictionary*) s više informacija.

4.3. Metrike evaluacije

U području strojnog učenja, evaluacija izvedbe modela je vrlo važna. Odabir odgovarajuće metrike evaluacije daje uvid u to koliko dobro model obavlja svoju zadaću. Ovo je potpoglavlje posvećeno opisu tri korištene metrike evaluacije, a korišteno je više metrika zbog potrebe za višestrukim uvidom u kvalitetu estimacija. Olakšan je proces odabira najboljih parametara i modela jer prilikom finalnih razmatranja mnogi modeli imaju vrlo slične rezultate. Razmatrane su tri metrike: R^2 odnosno koeficijent determinacije, srednja apsolutna pogreška MAE i prosječna kvadratna pogreška RMSE. U finalnom dijelu rada korištena je srednja vrijednost i standardna devijacija svih metrika kako bi se usporedile performanse treniranja i testiranja.

4.3.1. Koeficijent determinacije (R^2)

R^2, također poznat kao koeficijent determinacije, kvantificira udio varijance u ciljnoj varijabli. To je vrijednost između 0 i 1, s višim vrijednostima koje pokazuju bolju prilagodbu modela podacima. U biti, R^2 mjeri koliko su dobro predviđanja modela usklađena sa stvarnim podacima. R^2 rezultat od 0 označava da model ne objašnjava nikakvu varijancu, dok rezultat od 1 znači da model savršeno predviđa varijancu u ciljanoj varijabli. Vrijednosti između 0 i 1 predstavljaju udio objašnjene varijance, gdje više vrijednosti ukazuju na bolju izvedbu modela [41,42]. To je prikazano pomoću izraza (4-1)

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}, \qquad (4\text{-}1)$$

gdje je $\hat{y}$ estimirana vrijednost prave vrijednosti *y*, a $\bar{y}$ srednja vrijednost.

4.3.2. Srednja apsolutna pogreška (MAE)

Srednja apsolutna pogreška, kao što naziv sugerira, izračunava prosječnu apsolutnu razliku između predviđanja modela i stvarnih vrijednosti. MAE pruža mjeru veličine pogrešaka, bez razmatranja njihovog smjera (precjenjivanje ili podcjenjivanje). Niže vrijednosti MAE ukazuju na bolje performanse modela. Niži MAE sugerira da su predviđanja modela u prosjeku bliža stvarnim vrijednostima [42,43]. Izračun MAE je prikazan izrazom (4-2)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|, \qquad (4\text{-}2)$$

a oznake unutar jednadžbe imaju ista značenja kao kod R^2.

4.3.3. Prosječna kvadratna pogreška (RMSE):

Prosječna kvadratna pogreška je varijanta MAE-a koja, ne samo da uzima u obzir veličinu pogrešaka, već i jače „kažnjava“ veće pogreške. Izračunava se uzimanjem kvadratnog korijena srednje vrijednosti kvadrata razlika između predviđanja i stvarnih vrijednosti. RMSE daje procjenu tipične veličine pogrešaka, naglašavajući utjecaj netipičnih vrijednosti. Kao i MAE, niži RMSE ukazuje da su predviđanja modela bliža stvarnim vrijednostima. RMSE je posebno osjetljiv na veće pogreške, što ga čini vrijednom metrikom kada su ekstremne vrijednosti kritične za razmatranje, kao što može biti slučaj u ovom radu [42,44]. Izračun RMSE prikazan je izrazom (4-3)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}, \qquad (4\text{-}3)$$

gdje je $\hat{y}$ estimirana vrijednost prave vrijednosti y, a $\bar{y}$ srednja vrijednost.

U potrazi za točnom estimacijom temperature rotora PMSM, R^2 pruža uvid u udio temperaturne varijance, dok MAE i RMSE kvantificiraju točnost i veličinu pogrešaka.

4.3.4. Dodatne metrike evaluacije

Kao što je opisano u metodologiji evaluacije u radu [28] većina radova koji se bave umjetnom inteligencijom i strojnim učenjem uglavnom pokazuju evaluaciju izračunatu na rezultatima testa, ali mišljenje je da su rezultati treninga također bitni. Ukoliko su rezultati evaluacije na setu za obuku visoki, a na setu za test slabiji, to ukazuje na potencijalno prekomjerno prilagođavanje setu za obuku. Stoga će u ovom radu biti prikazana srednja vrijednost rezultata treniranja tokom unakrsne validacije (CV) i testiranja za dodatnu analizu. Također će se izračunati i prikazati standardna devijacija rezultata treniranja i testiranja kako bi se pokazalo ukoliko je postignuta stabilna, generalizirana i robusna estimacija ciljanih vrijednost. Postupak će se odvijati u sljedećim koracima:

1. Treniranje metodom strojnog učenja na setu podataka za obuku,
2. Evaluirati metodu strojnog učenja na setu podataka za obuku,
3. Evaluirati metodu strojnog učenja na setu podataka za testiranje,
4. Izračunati srednju vrijednost rezultata evaluacije koraka 2. i 3. po izrazima iz izraza (4-4), (4-5) i (4-6),

$$\overline{R^{2\prime}} = \frac{\overline{R_{CV}^2} + R_{test}^2}{2}, \tag{4-4}$$

gdje je $\overline{R_{CV}^2}$ srednja vrijednost koeficijenta determinacije prilikom unakrsne validacije, a R_{test}^2 koeficijent determinacije seta za testiranje. Ista analogija se primjenjuje kod izraza (4-5) i (4-6)

$$\overline{MAE'} = \frac{\overline{MAE_{CV}} + MAE_{test}}{2}, \tag{4-5}$$

$$\overline{RMSE'} = \frac{\overline{RMSE_{CV}} + RMSE_{test}}{2}. \tag{4-6}$$

5. Izračunati standardnu devijaciju rezultata evaluacije koraka 2. i 3. po izrazima iz jednadžbi (4-7), (4-8) i (4-9)

$$R_{STD}^2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(R_i^2 - \overline{R^{2\prime}}\right)^2}, \tag{4-7}$$

gdje je $\overline{R^{2\prime}}$ netom opisan izrazom (4-4), a R_i^2 predstavlja koeficijent determinacije svake pojedine iteracije unakrsne validacije i testa. Ista analogija se primjenjuje kod izraza (4-8) i (4-9)

$$MAE_{STD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(MAE_i - \overline{MAE'}\right)^2}, \tag{4-8}$$

$$RMSE_{STD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(RMSE_i - \overline{RMSE}')^2}. \tag{4-9}$$

Ove dodatne metrike će se primijeniti prilikom konačnih razmatranja s najboljim ansamblom te im je dodan ' kako bi se $\overline{R^{2\prime}}$ razlikovao od srednje vrijednosti prilikom unakrsne validacije $\overline{R^2}$.

4.4. Hiperparametri

U istraživanju čim bolje konfiguracije modela strojnog učenja odabir hiperparametara ima važnu ulogu. Ovo potpoglavlje se bavi upravo njima, istražujući njihovo značenje, tehnike za njihovo podešavanje i njihov utjecaj na izvedbu modela. U strojnom učenju, hiperparametri su parametri koji se ne uče iz podataka, već se postavljaju prije procesa treninga. Oni upravljaju ponašanjem algoritma treniranja i utječu na to kako model uči iz podataka, a mogu značajno

utjecati na performanse, složenost i ponašanje modela strojnog učenja. Ispravno podešavanje hiperparametara može pomoći u uspostavljanju ravnoteže između pretjeranog prilagođavanja modela i pretjeranog pojednostavljenja modela [32,45].

Podešavanje hiperparametara je proces pronalaženja optimalne konfiguracije za korišteni model, a u ovom radu se koristilo pretraživanje mreže (eng. *Grid search*) i nasumično pretraživanje (eng. *Random search*) u kombinaciji s unakrsnom provjerom. Iako će biti obrađeno pretraživanje mreže, ono se koristilo osjetno manje i korištena je gotova funkcija iz *scikit-learn*, dok je nasumično pretraživanje primarna metoda napisana iz nule kako bi se prilagodilo potrebama ovog rada.

4.4.1. Pretraživanje mreže

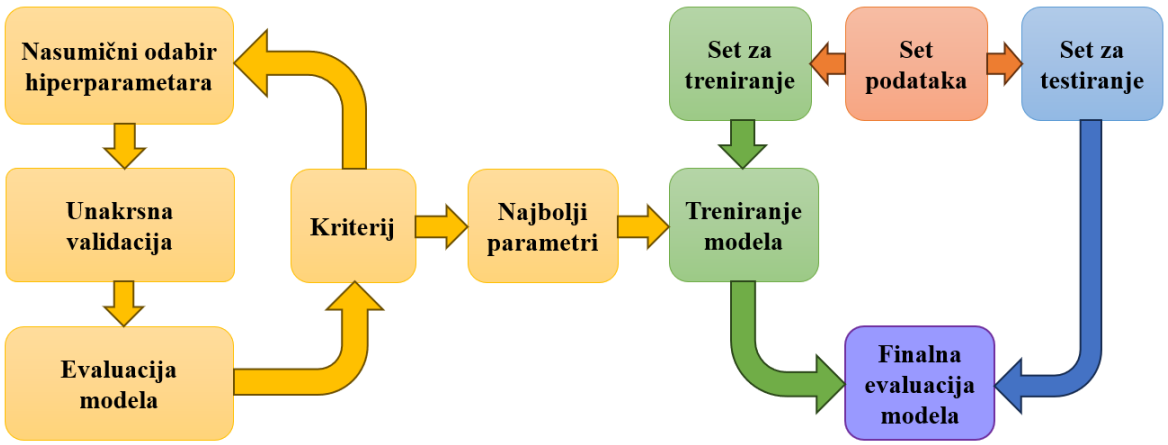
Pretraživanje mreže je sustavna i iscrpna tehnika za podešavanje hiperparametara u strojnom učenju. Uključuje definiranje skupa vrijednosti ili raspona hiperparametara i zatim evaluaciju izvedbe modela korištenjem svih mogućih kombinacija tih vrijednosti. Pretraživanje mreže vrijedan je pristup kada je potrebno sveobuhvatno istražiti unaprijed definirani set hiperparametara. Proces pretraživanja mreže počinje definiranjem mreže hiperparametara, na način da se odrede hiperparametri koji se žele podesiti te se definira raspon vrijednosti ili diskretnih izbora za svaki hiperparametar. Iterativno se trenira i evaluira model strojnog učenja za svaku kombinaciju hiperparametara u mreži te se dobije informacija o evaluacijama modela iz mreže. Prednost ove metode je što je sveobuhvatna u smislu da se istražuje cijeli unaprijed definirani prostor hiperparametara, osiguravajući da niti jedna kombinacija ne ostane neispitana. Rezultati pretraživanja mreže lako se mogu interpretirati, što omogućuje razumijevanje kako različite postavke hiperparametara utječu na performanse modela. Iako je pretraživanje mreže moćno, ono je procesorski zahtjevno i dugotrajno jer se za svaku kombinaciju hiperparametara odrađuje obuka modela te evaluacija [45].

U ovom radu pretraživanje mreže poslužilo je kao alat za ispitivanje modela s manjim brojem hiperparametara gdje je cilj bio dobiti informaciju o performansama modela za konkretne hiperparametre kako bi se ugodilo nasumično pretraživanje hiperparametara.

4.4.2. Nasumično pretraživanje

Nasumično pretraživanje je svestrana i učinkovita tehnika za podešavanje hiperparametara u strojnom učenju. Za razliku od pretraživanja mreže, koje iscrpno istražuje sve moguće kombinacije vrijednosti hiperparametara, nasumično pretraživanje nasumično uzorkuje hiperparametre iz unaprijed definiranih distribucija. Proces nasumičnog pretraživanja počinje određivanjem koje hiperparametre se želi podesiti te određivanjem opsega za svaki hiperparametar. Kada se nasumično odaberu vrijednosti hiperparametara iz prethodno definiranog opsega dolazi do jedinstvene kombinacije hiperparametara kojom se obučava model strojnog učenja. Nakon učenja se evaluiraju performanse modela. Prednost ove metode je što u odnosu na pretraživanje mreže obično zahtjeva manje iteracija za dolazak do rješenja jer se ne istražuju sve moguće kombinacije te se relativno brzo mogu identificirati obećavajuće vrijednosti hiperparametara [45].

Za potrebe ovog rada je ručno isprogramiran postupak nasumičnog pretraživanja zbog dodatnih pogodnosti potrebnih u analizi. Koristi se postupak skiciran na slici 4-4 gdje se iterativno odabiru nasumični hiperparametri koji se treniraju unakrsnom validacijom te se nakon treniranja performanse modela evaluiraju. Postavljen je kriterij koji provjerava ukoliko je zadovoljen uvjet R^2 veći od 0,99. Ako je zadovoljen uvjet, pronađena je kombinacija najboljih hiperparametara, a ako nije ponavlja se proces odabira nasumičnih hiperparametara. Nasumično odabrani hiperparametri te evaluacija svakog modela zapisuje se u .txt datoteku za potrebe daljnje analize. Najbolji hiperparametri se kasnije koriste za treniranje modela i provjeru evaluacije modela na setu za testiranje čime se simuliraju performanse modela na neviđenim podacima.



Slika 4-4 Metodologija nasumičnog pretraživanja

4.5. Ansambl

Metode ansambla (eng. *ensemble*) moćan su pristup poboljšanju izvedbe modela kombiniranjem estimacija više modela strojnog učenja. Učenje ansambla uključuje kombinaciju više modela za stvaranje robusnijeg i točnijeg modela estimacije, s idejom da skupljanjem raznolikog znanja različitih modela, ansambl može nadmašiti pojedinačne modele. U radu su korišteni ET (eng. *Extremely randomized trees*), Bagging (eng. *Bootstrap Aggregating)* i *Voting* regresor.

4.5.1. ExtraTrees (ET)

Temelj ET regresora su već opisana stabla odlučivanja. Razlike su u tome što ET iskorištava snagu skupnog učenja kroz ansambl. Kombiniraju se estimacije iz višestrukih osnovnih modela (stabla odlučivanja) kako bi se poboljšala konačna estimacija. Agregiranjem rezultata pojedinačnih stabala smanjuje se pretjerano prilagođavanje i povećava robusnost modela. Posebnost modela je uvođenje dodatne slučajnosti u procesu izgradnje stabla. Za razliku od stabla odlučivanja, koja traže optimalnu podjelu za svaki čvor, čvorovi ET odabiru nasumične podjele iz skupa nezavisnih varijabli. Ova slučajnost dovodi do raznolikosti među stablima. U tablici 5 nalaze se hiperparametri ET, većina ih je identična stablu odlučivanja, osim *n_estimators,* što označava broj stabala odlučivanja u „šumi“ [46,47].

Tablica 5 ET hiperparametri

Hiperparametar	Donja granica	Gornja granica	Zadana vrijednost
n-estimators	10	150	100
criterion	squared_error, friedman_mse, absolute_error, poisson		squared_error
max_depth	5	50	None
min_samples_split	2	10	2
min_samples_leaf	1	10	1
max_features	int, None		None

4.5.2. Bootstrap Aggregating (Bagging)

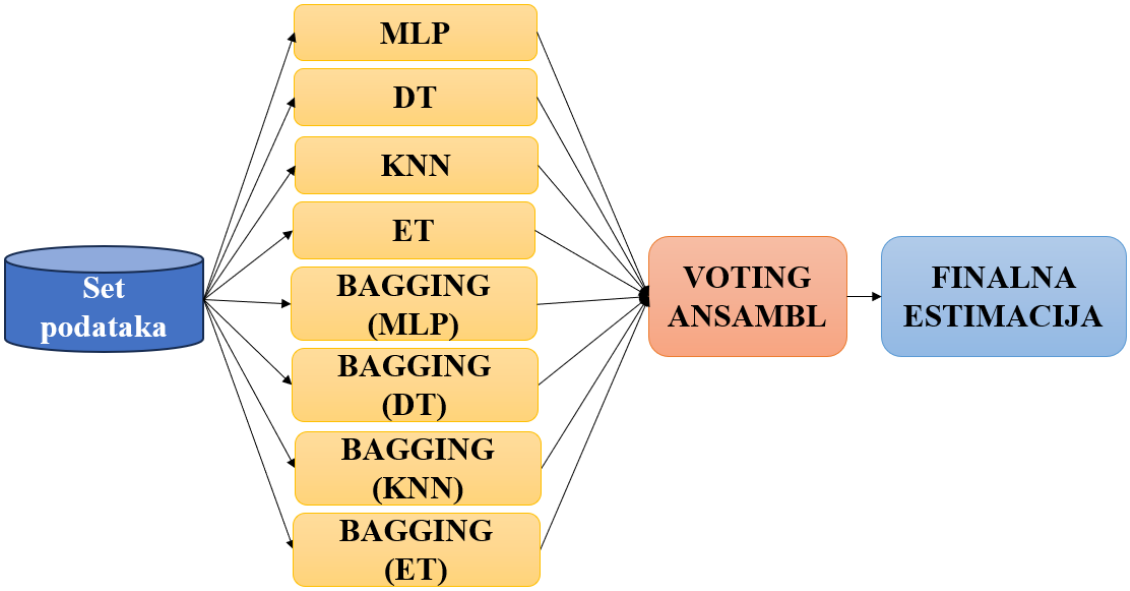
Bagging je ansambl metoda koja kombinira višestruke bazne modele kako bi poboljšala točnost i stabilnost regresijskih estimacija. Djeluje na ideji da agregiranje predviđanja iz višestrukih modela može smanjiti varijancu i prekomjerno prilagođavanje pojedinog modela odnosno baznog modela koji koristi. Bagging generira više podsetova podataka trening seta podataka te nasumično odabire podatkovne točke koje zamjenjuje te se svaki podset koristi za treniranje zasebnog osnovnog regresora. Bagging, dakle, koristi zbirku osnovnih regresora koji se neovisno treniraju na različitim boostrap uzorcima. Estimacije iz osnovnih regresora se agregiraju kako bi se dobila konačna estimacija ansambla. Hiperparametri bagging regresora se nisu podešavali nasumičnim podešavanjem jer nije bilo potrebe te su korišteni zadani hiperparametri, a jedino se mijenjao estimator [46,48,49].

Tablica 6 Bagging hiperparametri

Hiperparametar	Izbor	Zadana vrijednost
estimator	MLP, DT, KNN, ET	None (DT)
n_estimators	/	10
max_samples	/	1
min_features	/	1

4.5.3. Voting ansambl

Voting je ansambl metoda za zadatke regresije. Kombinira predviđanja iz višestrukih regresijskih modela i daje konačnu estimaciju usrednjavanjem njihovih rezultata. Nekoliko regresijskih modela, treniraju se neovisno na istom setu podataka. Tijekom estimiranja, svaki model generira vlastiti izlaz. Konačno, *voting* ansambl zatim agregira te rezultate, uzimajući njihov prosjek, što rezultira konačnom estimacijom. *Voting* ansambl je jednostavan za implementaciju i može poboljšati točnost estimacije smanjenjem utjecaja pristranosti specifičnih za model. Posebno je korisno korištenjem nekoliko različitih modela. Na slici 4-5 prikazan je način korištenja *voting* ansambla u ovom radu.



Slika 4-5 Voting ansambl

5. REZULTATI

U ovom poglavlju bit će predstavljeni rezultati estimacije temperature pm. Istraživanje za ovaj rad je krenulo analizom seta podataka, korelacijskom analizom te predobradom seta podataka, kao što je opisano u prethodnim poglavljima ovog rada. Nadalje se zalazi u rezultate s naglaskom na izvedbu različitih regresijskih modela, konkretne vrijednosti hiperparametara koji su se koristili, implementacija unakrsne validacije i konačno, *voting* ansambl. Na početku istraživanja korišteno je prijenosno računalo Asus VivoBook s 4 jezgrenim procesorom (8 logičkih jezgri) AMD Ryzen 7 3700U s Radeon Vega Mobile Gfx na 2.30 GHz te 16 GB radne memorije. U kasnijem istraživanju korišteno je stolno računalo s 4 jezgrenim procesorom (8 logičkih jezgri) Intel® Core™ i7-7700K na 4.20 GHz, grafičkom karticom FXF Radeon RX 580 te 16 GB radne memorije na 2133 MHz.

5.1. Inicijalno istraživanje

Na samom početku istraživanja odabran je MLP kao moćan model strojnog učenja kako bi se istražile mogućnosti estimacije temperature rotora PMSM. Korištenjem Spyder-a iz Anaconda navigator-a pisane su skripte u programskom jeziku Python. Učitan je set podataka pomoću *Pandas* biblioteke, a funkcijom *train_test_split* podijeljen je na set za treniranje i set za testiranje u omjeru 70-30 uz korištenje *random_state* parametra kako bi se i prilikom budućih treniranja koristila ista podjela kako bi se moglo relevantno uspoređivati modele. Učitan je MLP regresor i pomoću funkcije *fit(X_train,y_train)* počelo je treniranje.

```
dff = pd.read_csv("measures_v2.csv")
df = dff.drop('profile_id', axis=1)

y = df.pop("pm")
X_train, X_test, y_train, y_test = train_test_split(df, y,test_size=0.3,
                                                    random_state=42)

regr = MLPRegressor(verbose = True)
regr.fit(X_train,y_train)
score_test = regr.score(X_test,y_test)

y_predict = regr.predict(X_test)

r2 = r2_score(y_test, y_predict)
MAE = mean_absolute_error(y_test, y_predict)
RMSE = np.sqrt(mean_squared_error(y_test, y_predict))
hyperparameters = regr.get_params()
```

Rezultati ovog modela prikazani su u tablici 7, a korišteni hiperparametri u tablici 8.

Tablica 7 Evaluacija MLP-a sa zadanim vrijednostima

Model	R^2	MAE	RMSE
MLP	0.9489	3.0476	4.2964

Dalje se krenulo implementacijom nasumične pretrage hiperparametara da bi se poboljšala izvedba MLP modela te istraživanjem utjecaja uklanjanja netipičnih vrijednosti i skaliranje podataka. Nakon testiranja nasumične pretrage hiperparametara na neobrađenom setu podataka dobiveni su bolji modeli od onog sa zadanim vrijednostima hiperparametara pa se na njima odradilo istraživanje utjecaja uklanjanja outliera i skaliranje.

Tablica 8 MLP model za inicijalno istraživanje

Hiperparametar	Korištena vrijednost	Zadana vrijednost
Number of Hidden Layers	3	1
Number of neurons	46, 27, 27	100
activation	relu	'relu'
solver	adam	'adam'
alpha	0.003160	0.0001
learning_rate_init	0.000694	0.001
max_iter	750	200
shuffle	False	True
beta_1	0.9	0.9
beta_2	0.999	0.999
n_iter_no_change	10	10

Kao što je opisano u prošlim poglavljima, korištene su metode za uklanjanje netipičnih vrijednosti iz *Feature-engine* odnosno *Winsorizer* i *OutlierTrimmer* funkcije, a za skaliranje *StandardScaler* i *PowerTransformer*. U tablici 9 vidljivo je da *OutlierTrimmer* daje bolje rezultate od *Windsorizer* te da u kombinaciji sa *StandardScaler* daje bolje rezultate nego u kombinaciji sa *PowerTransformer*.

Tablica 9 Rezultati inicijalnog istraživanja

Netipične vrijednosti	Skaliranje	R^2	MAE	RMSE
/	/	0.967967	2.316938	3.402646
Winsorizer	/	0.967189	2.335687	3.433243
OutlierTrimmer	/	0.969382	2.219300	3.296328
OutlierTrimmer	**StandardScaler**	**0.977119**	**1.846458**	**2.849512**
OutlierTrimmer	PowerTransformer	0.969352	2.230888	3.297934

Nadalje se krenulo sa ispitivanjem drugih regresijskih modela kako bi se ustvrdilo kojima će se uz MLP nastaviti istraživanje. Koristili su se *OutlierTrimmer*, *StandardScaler*, *train_test_split* te standardne metrike evaluacije. Ispitivali su se svi modeli regresije iz tablice 10 te su u njoj navedeni i svi rezultati. Jedini modeli s kojima se nastavilo istraživanje su KNN i DT.

Tablica 10 Rezultati isprobanih modela

Model	R^2	MAE	RMSE
LinearRegression()	0.85446	5.38620	7.18666
Ridge()	0.85446	5.38612	7.18666
Lasso()	0.75476	7.15349	9.32915
Lasso(alpha=0.1)	0.83057	5.84378	7.75424
ElasticNet()	0.72581	7.72463	9.86441
LassoLars()	0.78428	6.32569	8.43525
BayesianRidge()	0.85446	5.38619	7.18666
ARDRegression()	0.85446	5.38622	7.18666
LinearSVR()	0.84783	5.27998	7.34867
SGDRegressor()	0.85403	5.38146	7.19734
HuberRegressor()	0.85130	5.29179	7.26440
KNeighborsRegressor()	0.99609	0.35123	1.17740
DecisionTreeRegressor()	0.99621	0.25328	0.89689

Za potrebe daljnjeg istraživanja napisane su funkcije za uklanjanje netipičnih vrijednosti i skaliranje, a skaliranje je odrađeno po preporukama iz [50] na način da se *StandardScaler*

prilagodi setu za treniranje, a da se po tim postavkama skalira set za trening. Upravo tako bi se skaliralo neke nove neviđene podatke koje treba skalirati prije estimacije. Tako se prilikom daljnjeg rada uvijek na početku nakon učitavanja seta podataka i podjele podataka pozovu funkcije *Outlier_trim* i *Scaler*.

```
def Outlier_Trim(df):
    trimmer = OutlierTrimmer(
    capping_method='gaussian', tail='both', fold=2.8,
variables=list(df.columns))
    data_trim = pd.DataFrame(trimmer.fit_transform(df), columns =
list(df.columns))
    df = data_trim
    return df

def Scaler (X_train, X_test):
    scaler = StandardScaler()
    scaler.fit(X_train)
    columns = list(X_train.columns)
    index = list(X_train.index)
    index2 = list(X_test.index)
    X_train = pd.DataFrame(scaler.transform(X_train),
                        columns = columns, index = index)
    X_test = pd.DataFrame(scaler.transform(X_test),
                        columns = columns, index = index2)
    return X_train, X_test
```

5.2. Nasumično pretraživanje hiperparametara s unakrsnom validacijom

S odabranim regresorima se zatim primijenila nasumična pretraga hiperparametara. Kroz primjer funkcije za izbor nasumičnih parametara MLP-a se vidi cjelovita funkcionalnost, a to je iz razloga što je ona najkompleksnija te se ovisno o izabranom *solveru* biraju drugačiji hiperparametri. Na samom početku se inicijalizira prazna lista *mlpParam* u koju će se kasnije zapisivati hiperparametri. Prvo se odabiru hiperparametri koji ne ovise o solveru, i to iz

```
def MLPRandomSearch():

    mlpParam = []
    HLS = tuple(np.random.randint(low=15, high=100,
                                  size=np.random.randint(1, 5)))
    activation = random.choice(['identity', 'logistic', 'tanh', 'relu'])
    solver = random.choice(['adam', 'sgd', 'lbfgs'])
    alpha = 10 ** np.random.uniform(low=-5, high=-1)
    max_iter = 1500
    mlpParam.append(HLS)
    mlpParam.append(activation)
    mlpParam.append(solver)
    mlpParam.append(alpha)
    mlpParam.append(max_iter)
```

ponuđene liste, ili se generiraju nasumično u definiranom rasponu. Ti parametri se zatim zapisuju u listu koja je inicijalizirana na početku.

U idućem dijelu je definirano zapisivanje odabranih hiperparametara u .txt datoteku kako bi se kasnije lako provjerili korišteni hiperparametri, te se kreće s provjerom koji je solver odabran. U ovom primjeru se nalazi samo *sgd* kako bi se objasnio princip, a cjeloviti kod se nalazi na kraju rada. Ukoliko je odabran *sgd,* biraju se potrebni hiperparametri te se oni dodaju u listu *mlpParam* i također se zapisuju u Param.txt. U konzolu se ispisuje koji je solver odabran te se kreira *Mlpregr* koji sadrži sve odabrane hiperparametre te funkcija vraća te podatke.

```
with open('Param.txt', 'a') as file:
          file.write("\n")
          file.write("Random MLP Parameters:\n")
          file.write(f"Hidden Layer Sizes: {mlpParam[0]}\n")
          file.write(f"Activation: {mlpParam[1]}\n")
          file.write(f"Solver: {mlpParam[2]}\n")
          file.write(f"Alpha: {mlpParam[3]}\n")
          file.write(f"Max Iterations: {mlpParam[4]}\n")
if solver == 'sgd':
    learning_rate = random.choice(['constant','adaptive'])
    learning_rate_init = 10 ** np.random.uniform(low=-4, high=-2)
    n_iter_no_change = 12
    shuffle = bool(np.random.choice([0, 1]))
    mlpParam.append(learning_rate)
    mlpParam.append(learning_rate_init)
    mlpParam.append(n_iter_no_change)
    mlpParam.append(shuffle)
    with open('Param.txt', 'a') as file:
        file.write(f"learning_rate: {mlpParam[5]}\n")
        file.write(f"learning_rate_init: {mlpParam[6]}\n")
        file.write(f"n_iter_no_change: {mlpParam[7]}\n")
        file.write(f"shuffle: {mlpParam[8]}\n")
    print("MLP with sgd solver chosen")
    Mlpregr = MLPRegressor(hidden_layer_sizes=mlpParam[0],
                           activation=mlpParam[1],
                           solver=mlpParam[2],
                           alpha=mlpParam[3],
                           max_iter=mlpParam[4],
                           learning_rate = mlpParam[5],
                           learning_rate_init=mlpParam[6],
                           n_iter_no_change=mlpParam[7],
                           shuffle=mlpParam[8],
                           verbose=True, random_state=42)
```

U nastavku koda je definiran uvjet (R^2>0.999) te *while* petlja koja unakrsnom validacijom trenira model s izabranim hiperparametrima te na kraju provjerava ukoliko je pronađen model koji zadovoljava uvjet nakon čega se zaustavlja program i smatra se da je pronađeno konačno rješenje. Ukoliko uvjet nije zadovoljen, kreće se s pronalaskom novih hiperparametara te treniranjem i evaluacijom novog modela. Tokom pretrage hiperparametara primijenjena je

unakrsna validacija funkcijom *cross_val_score* te evaluacijska metrika R^2 odnosno srednja vrijednost $\overline{R^2}$ i standardna devijacija između iteracija CV. U idućim tablicama nalaze se najbolji modeli pojedinih regresijskih modela. U tablici 11 nalaze se MLP modeli zajedno s hiperparametrima i evaluacijom.

Tablica 11 Najbolji MLP modeli

Hiperparametar	Model 1	Model 2	Model 3	Model 4
Number of Hidden Layers	4	3	4	3
Number of neurons	(63,42,57,22)	(62, 31, 27)	(33, 66, 24, 19)	(38, 41, 17)
activation	tanh	tanh	tanh	logistic
solver	adam	adam	adam	sgd
alpha	0.000783	0.002404	0.000253	0.000033
learning_rate	/	/	/	constant
learning_rate_init	0.000210	0.000446	0.000102	0.000895
max_iter	1500	1500	1500	1000
shuffle	TRUE	TRUE	FALSE	FALSE
n_iter_no_change	12	12	12	10
$\overline{R^2}$	0.991697	0.986629	0.988188	0.980571
R^2 **STD**	0.000242	0.000262	0.000413	0.001607

Sličnom funkcijom te identičnim postupkom se biraju hiperparametri DT modela te se u tablici 12 nalaze najbolji DT modeli.

Tablica 12 Najbolji DT modeli

Hiperparametar	Model 1	Model 2	Model 3	Model 4
criterion	squared_error	poisson	friedman_mse	squared_error
splitter	best	best	best	best
max_depth	None	44	24	29
min_samples_split	2	2	8	2
min_samples_leaf	1	11	12	18
max_features	None	None	None	None
$\overline{R^2}$	0.997599	0.995526	0.995345	0.994195
R^2 **STD**	0.000184	0.000149	0.000184	0.000109

U tablici 13 su prikazani najbolji rezultati KNN modela.

Tablica 13 Najbolji KNN modeli

Hiperparametar	Model 1	Model 2	Model 3	Model 4
n_neighbors	5	5	2	7
weights	distance	uniform	distance	distance
algorithm	auto	auto	auto	kd_tree
leaf_size	30	30	30	41
$\overline{R^2}$	0.9959	0.9958	0.996976	0.980561
R^2 **STD**	0.000071	0.000071	0.000061	0.000060

Konačno, prije korištenja *voting* ansambla testirani su i ET modeli te su njihovi rezultati prikazani u tablici 14.

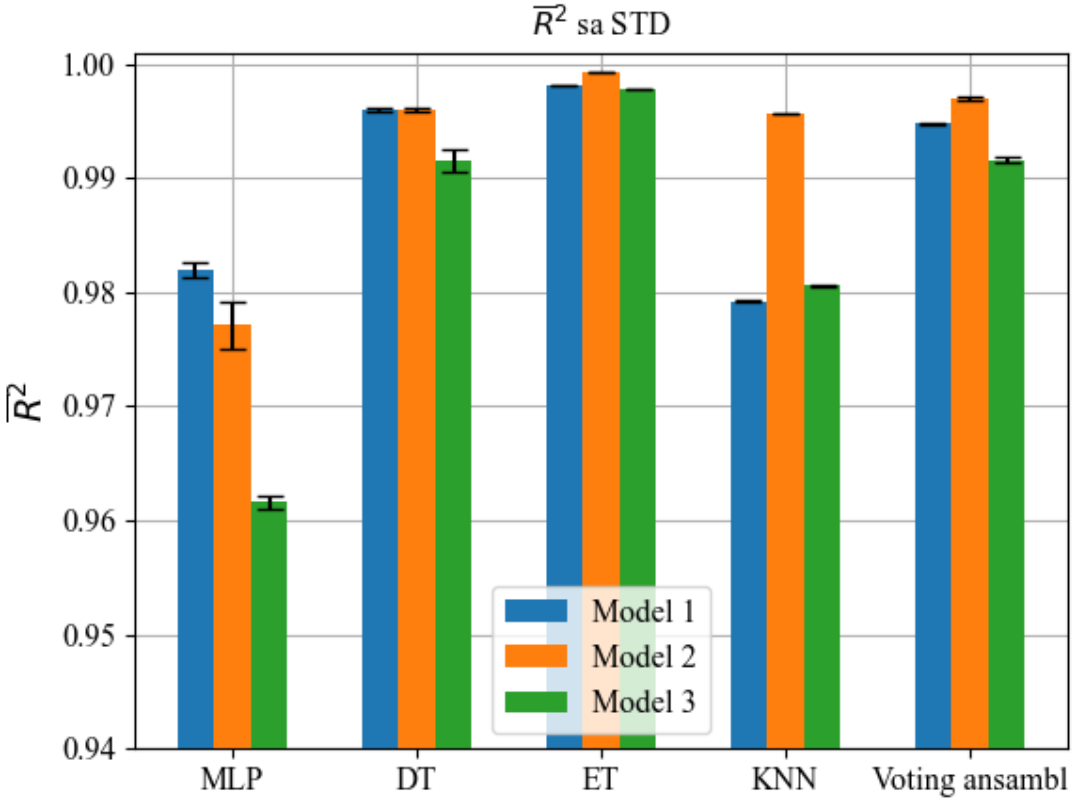
Tablica 14 Najbolji ET modeli

Hiperparametar	Model 1	Model 2	Model 3	Model 4
n_estimators	100	30	91	84
criterion	squared_error	poisson	poisson	friedman_mse
max_depth	None	None	None	25
min_samples_split	2	2	3	7
min_samples_leaf	1	4	8	8
$\overline{R^2}$	0.999693	0.999042	0.998107	0.997831
R^2 **STD**	0.000021	0.000015	0.000038	0.000042

5.3. Voting ansambl

Nakon što su izrađene funkcije za nasumično pretraživanje za sve spomenute modele, napisana je skripta koja unutar *while* petlje poziva funkcije za nasumično pretraživanje te zatim radi unakrsnu validaciju prilikom koje se računa srednja vrijednost i standardna devijacija R^2. Skripta je nadograđena dodavanjem *voting* ansambla u koji su dodana četiri modela te je unakrsnom validacijom treniran *voting* ansambl koji je zatim i evaluiran. U tablici 15, te na slici 5-1 su prikazani rezultati modela dobiveni ovom metodom. Stupac na grafu predstavlja $\overline{R^2}$ (srednju vrijednost), a brkovi označavaju standardnu devijaciju (STD).

```
while voting_mean < 0.999:
    mlp = MLPRandomSearch()
    dt = DTRandomSearch()
    et = ETRandomSearch()
    knn = KNNRandomSearch()

    mlp_scores = cross_val_score(mlp, X_train, y_train, cv=5,
    scoring='r2')
    dt_scores = cross_val_score(dt, X_train, y_train, cv=5,
    scoring='r2')
    et_scores = cross_val_score(et, X_train, y_train, cv=5,
    scoring='r2')
    knn_scores = cross_val_score(knn, X_train, y_train, cv=5,
    scoring='r2')

    mlp_mean, mlp_std = mlp_scores.mean(), mlp_scores.std()
    dt_mean, dt_std = dt_scores.mean(), dt_scores.std()
    et_mean, et_std = et_scores.mean(), et_scores.std()
    knn_mean, knn_std = knn_scores.mean(), knn_scores.std()
    voting_regressor = VotingRegressor(estimators=[
        ('mlp', mlp),
        ('dt', dt),
        ('et', et),
        ('knn', knn)
        ])

    voting_scores = cross_val_score(voting_regressor, X_train, y_train,
                                    cv=5, scoring='r2')
    voting_mean, voting_std = voting_scores.mean(), voting_scores.std()
```

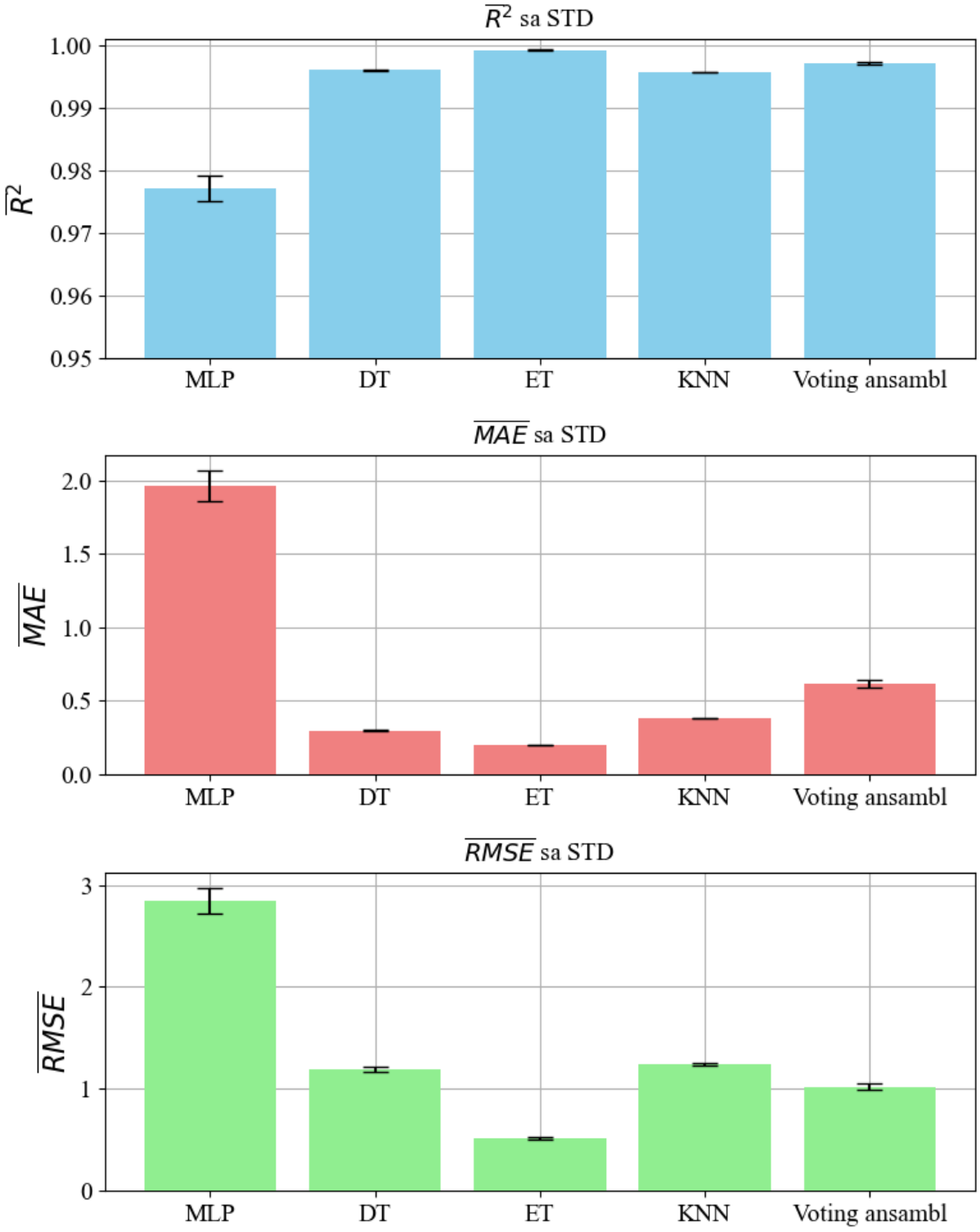


Slika 5-1 Voting ansambl - R^2

Tablica 15 Najbolji ansambl modeli

Model strojnog učenja	Hiperparametri
MLP	hidden_layer_size = (66, 36, 69, 58), (61,58), (24, 39) activation = relu, tanh, logistic solver = adam, adam, adam alpha = 0.000366, 0.065470743, 0.000025 learning_rate_init = 0.007412, 0.006732162, 0.006329 shuffle = True, True, False
DT	criterion = squared_error, squared_error, squared_error splitter = best, best, random max_depth = None, 30, 25 min_samples_split = 9, 9, 7 min_samples_leaf = 9, 9, 6
KNN	n_neighbors = 46, 6, 41 weights = uniform, distance, uniform algorithm = auto, kd_tree, ball_tree leaf_size = 44, 40, 30
ET	n_estimators = 91, 30, 84 criterion = poisson, poisson, friedman_mse max_depth = None, 30, 25 min_samples_split = 3, 3, 7 min_samples_leaf = 8, 3, 8

Najbolji *voting* ansambl je ponovno obrađen korištenjem funkcije *cross_validate* kako bi se dobile sve potrebne metrike evaluacije i njihove standardne devijacije koje su zatim prikazane na slici 5-2.



Slika 5-2 Najbolji nasumični voting ansambl

5.4. Finalni rezultati

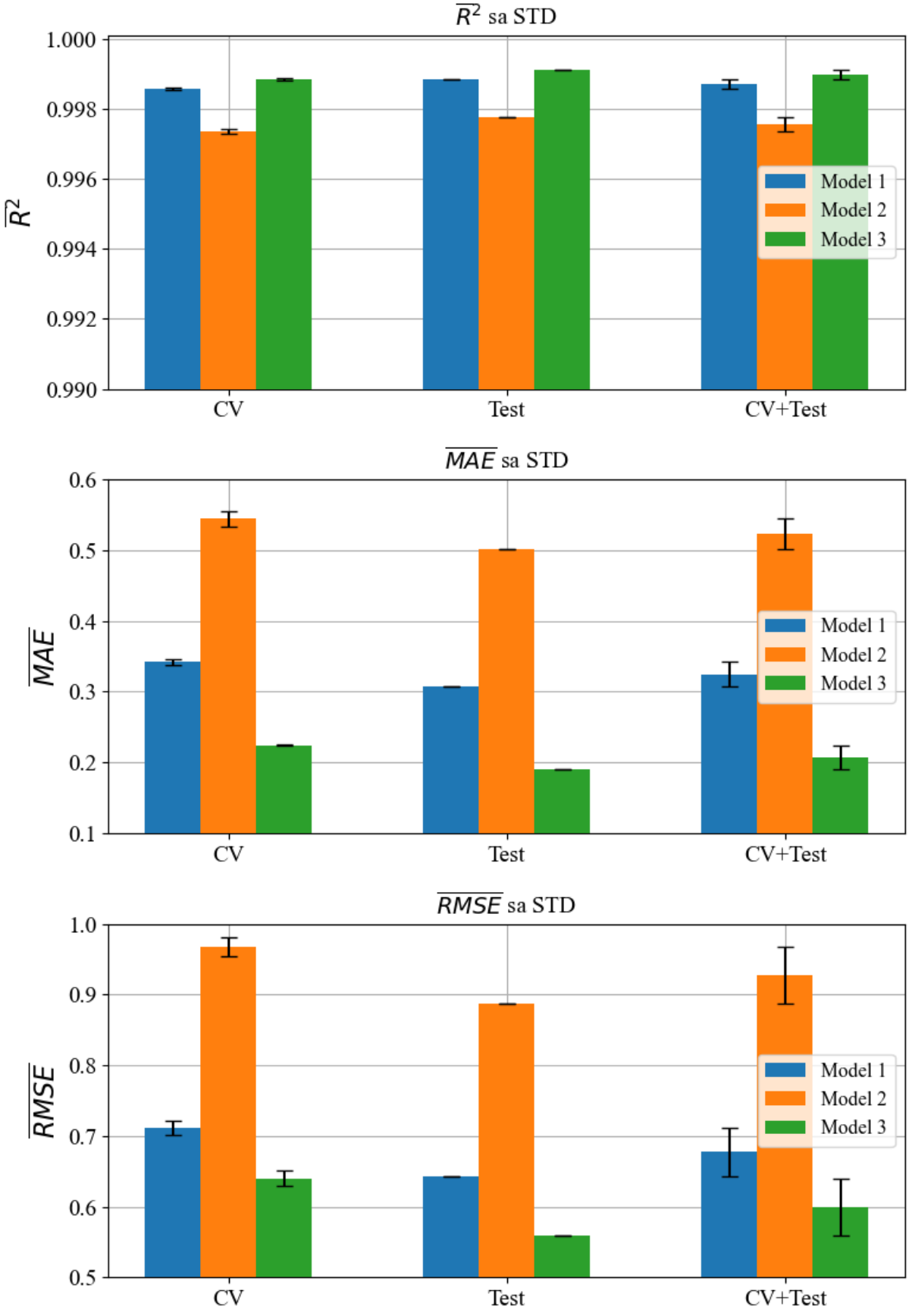
Finalni korak ovog rada je bio implementacija najboljih modela strojnog učenja u *voting* ansambl s dodatkom *bagging* modela kako bi se dodatno poboljšala estimacija. Ideja je da pored četiri osnovna modela se kreiraju još četiri *bagging* modela koji će kao *estimator* koristiti osnovne modele. Modeli strojnog učenja s hiperparametrima su prikazani u tablici 16. Računate su $\overline{R^2}, \overline{MAE}, \overline{RMSE}$ te standardne devijacije tokom unakrsne validacije, zatim R^2, MAE i $RMSE$ na setu za treniranje te konačno $\overline{R^{2\prime}}, \overline{MAE'}, \overline{RMSE'}$ što su evaluacijske metrike koje kombiniraju rezultate treniranja (CV) i testa.

```
br1 = BaggingRegressor(estimator = Mlpregr, random_state=42)
br2 = BaggingRegressor(estimator = DTregr, random_state=42)
br3 = BaggingRegressor(estimator = ETregr, random_state=42)
br4 = BaggingRegressor(estimator = KNNregr, random_state=42)

scoring = ['r2', 'neg_mean_absolute_error',
'neg_root_mean_squared_error']

voting_regressor = VotingRegressor(estimators=[
    ('mlp', mlp),
    ('dt', dt),
    ('et', et),
    ('knn', knn),
    ('br1', br1)
    ('br2', br2),
    ('br3', br3),
    ('br4', br4)])

results_voting = cross_validate(voting_regressor, X_train, y_train,
cv=5, scoring=scoring, n_jobs=-1)

voting_r2_mean = results_voting['test_r2'].mean()
voting_r2_std = results_voting['test_r2'].std()
voting_mae_mean = -results_voting['test_neg_mean_absolute_error'].mean()
voting_mae_std = results_voting['test_neg_mean_absolute_error'].std()
voting_rmse_mean = -
results_voting['test_neg_root_mean_squared_error'].mean()
voting_rmse_std =
results_voting['test_neg_root_mean_squared_error'].std()

voting_regressor.fit(X_train, y_train)
y_pred = voting_regressor.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

mean_r2_all = np.mean([voting_r2_mean, r2])
std_r2_all = np.std([voting_r2_mean, r2])
mean_mae_all = np.mean([voting_mae_mean, mae])
std_mae_all = np.std([voting_mae_mean, mae])
mean_rmse_all = np.mean([voting_rmse_mean, rmse])
std_rmse_all = np.std([voting_rmse_mean, rmse])
```

Tablica 16 Finalni voting ansambl

Model strojnog učenja	Hiperparametri
MLP	hidden_layer_size = (63, 42, 57, 22), (61,58) activation = tanh, tanh solver = adam, adam alpha = 0.0007827, 0.0654707 learning_rate_init = 0.0002099, 0.00673216 shuffle = True, True
DT	criterion = squared_error, squared_error splitter = best, best max_depth = None, 30 min_samples_split = 2, 9 min_samples_leaf = 1, 9
KNN	n_neighbors = 5, 6 weights = distance, distance algorithm = auto, kd_tree leaf_size = 40, 40
ET	n_estimators = 30 criterion = poisson max_depth = 30 min_samples_split = 3 min_samples_leaf = 3
Bagging1	estimator = MLP
Bagging2	estimator = DT
Bagging3	estimator = ET
Bagging4	estimator = KNN

Na slici 5-3 nalaze se prikazani rezultati gdje su model 1 i model 2 iz tablice 16, a model 3 koristi parametre modela 1 s time da nije korišten MLP i Bagging1, odnosno koristi 6 modela. U tablici 17 su prikazane dodatne metrike evaluacije opisane u istoimenom potpoglavlju u dijelu rada gdje su opisane evaluacijske metrike.



Slika 5-3 Finalni voting ansambli

Tablica 17 Dodatna evaluacija

Evaluacija	Model 1	Model 2	Model 3
$\overline{R^{2\prime}}$	0.998702961	0.997570064	0.998982646
R^2_{STD}	0.000135001	0.000209388	0.000138632
$\overline{MAE'}$	0.324420793	0.522797995	0.207639963
MAE_{STD}	0.017441727	0.021840724	0.016747872
$\overline{RMSE'}$	0.677176526	0.927283059	0.599132748
$RMSE_{STD}$	0.035003131	0.039576685	0.040703942

6. DISKUSIJA

U ovom poglavlju će se komentirati rezultati izloženi u prošlom poglavlju te dodatno pojašnjavati proces istraživanja u radu. Nakon početne statističke analize, obrade netipičnih vrijednosti i skaliranja seta podataka s MLP modelom, ustvrđeno je da su najbolji rezultati dobiveni primjenom *OutlierTrimmer* i *StandardScaler.* Izrađene su funkcije za automatiziranje tog procesa te se krenulo s implementacijom drugih modela regresije, i to iz početka linearne regresije. Međutim, niti jedan linearni model nije dao niti približno dobre rezultate te se niti s jednim nije nastavilo istraživanje. Modeli koji su dali dobre rezultate su KNN i DT.

Implementirano je nasumično pretraživanje hiperparametara čime je značajno poboljšana izvedba MLP modela, ali je sam proces bio izrazito dug, pogotovo kod modela s više skrivenih slojeva i većim brojem neurona. Također, zbog velikog broja hiperparametara i mnogo kombinacija, velik broj modela je nakon dugog treniranja dao loše rezultate. Najuspješniji modeli su najčešće koristili *adam* kao *solver*, zatim *sgd*, a vrlo rijetko *lbfgs*. Struktura MLP-a odnosno broj skrivenih slojeva je najviše utjecao na vrijeme treniranja, a 3-5 skrivenih slojeva s manje od 100 neurona po sloju su dali dobre rezultate u zadovoljavajućem vremenu treniranja. Aktivacijske funkcije su u početnim razmatranjima davale podjednake rezultate, osim *identity,* koji je i izbačen iz nasumičnog izbora. Veći *alpha* je donekle ubrzavao proces treniranja, ali konačno najbolji modeli su redom imali manji i vrlo sličan *alpha* hiperparametar. S obzirom da je *adam* redovito davao najbolje rezultate, dodatno se istražio utjecaj hiperparametara specifičnih za *adam* te je ustvrđeno da *beta_1* i *beta_2* ne utječu znatno na rezultate te u konačnim razmatranjima nisu niti nasumično birani, već su se koristile zadane vrijednosti. Veća inicijalna stopa učenja je ubrzavala treniranje, ali je često dovela do lošijih modela. Broj iteracija bez promjene se uglavnom koristio nešto veći od zadane vrijednosti da dozvoli da se modeli bolje istreniraju, ali je zato i sam proces treniranja duže trajao. Najbolji model ima $\overline{R^2}$ = 0.991697 uz std u četvrtoj decimali.

Izbor nasumičnih parametara za DT je bilo puno lakše implementirati zbog manjeg broja hiperparametara te je model sa zadanim vrijednostima u samom početku dao odlične rezultate. Nasumičnim izborom hiperparametara su se uglavnom dobivali lošiji rezultati od modela sa zadanim parametrima. Kod određenih kombinacija *min_samples_split* i *min_samples_leaf* s manjim *max_depth* je bilo problema prilikom treninga pa je implementiran uvjet da

min_samples_split bude manji od *min_samples_leaf*. Najbolji model ima $\overline{R^2}$ = 0.997599 uz std u četvrtoj decimali. Izbor nasumičnih hiperparametara za KNN je također bio jednostavniji za primjenu zbog manjeg broja hiperparametara, a model sa zadanim vrijednostima je dao odličan rezultat. Implementacijom nasumične pretrage hiperparametara pronađen je najbolji model s rezultatom $\overline{R^2}$ = 0.9959, uz std u petoj decimali. Prije implementacije *voting* ansambla isproban je ET koji je, očekivano zbog dobrog rezultata DT, dao još bolje rezultate te potvrdio pretpostavke da će metode ansambla poboljšati estimaciju temperature pm. Implementacijom nasumične pretrage hiperparametara nije pronađen bolji model od onog sa zadanim vrijednostima, a broj od 100 stabala s dubinom *None* je prilikom paralelnog rada (*n_jobs*) unakrsne validacije rubno imao problema sa dostupnom memorijom zbog svoje kompleksnosti.

Implementacijom *voting* ansambla koristila se unakrsna validacija, ali u početku *cross_val_score* funkcijom koja je koristila samo $\overline{R^2}$. Parametri modela su odabrani nasumičnim odabirom, a postignuti su dobri rezultati. MLP je još u prvoj fazi dao najlošije rezultate što se ponovilo i unutar ansambla, te je samo kod modela 1 postignut veći $\overline{R^2}$ MLP modela, u odnosu na neki drugi, odnosno u tom slučaju KNN. ET je dao najbolje rezultate, uz jako dobre rezultate DT. Najveća std je primijećena kod MLP modela. Ukupni rezultat najboljeg ansambla je odličan te je postignut $\overline{R^2}$ = 0.997 uz std u četvrtoj decimali. Najbolji *voting* ansambl je naknadno ponovno obrađen kako bi se dobio uvid u $\overline{MAE}$ i $\overline{RMSE}$ metrike iz čega je vidljiva velika razlika ostalih modela unutar ansambla te samog ansambla u odnosu na MLP model. Također, najveća std je prisutna kod MLP, što je utjecalo i na std ansambla. Ciljana vrijednost $\overline{MAE}$ i $\overline{RMSE}$ je čim manja odnosno da je čim bliže nuli, te su određeni modeli unutar ansambla imali odlične rezultate. Najbolje rezultate $\overline{MAE}$ imaju ET i DT, dok su što se tiče $\overline{RMSE}$ najbolji ET pa vrlo ujednačeni DT i KNN te ukupni *voting* ansambl.

Implementaciji konačnog *voting* ansambla s 8 modela je pristupljeno s najboljim parametrima svih modela te dodatkom *bagging* modela kako bi se postigli još bolji rezultati, a time su kreirani model 1 i model 2. S obzirom na prijašnja opažanja, uključen je i model 3 koji je sličan modelu 1, ali su izbačeni modeli MLP i *Bagging* (MLP) kako bi se dodatno obradio utjecaj MLP-a na ansambl koji je u prijašnjem koraku detektiran kao najslabiji model. Očekivano su postignuti još bolji rezultati u odnosu na prijašnje ansamble, te je najbolji model upravo model 3 kojem je izbačen MLP i *Bagging* (MLP). Dosegnut je $\overline{R^2}$ = 0.9988 uz std u četvrtoj decimali prilikom unakrsne validacije. U ovom finalnom istraživanju su implementirane sve opisane

metrike te se je dodatno evaluiralo modele na setu za test i konačno se računala srednja vrijednost rezultata unakrsne validacije i testa ($\overline{R^{2\prime}}, \overline{MAE'}, \overline{RMSE'}$). Model 3 je najbolji po svim metrikama, a model 2 je najlošiji po svim metrikama. Primarni razlog je nešto manje kompleksan MLP model koji je korišten i prilikom prethodnog razmatranja. Vrijeme potrebno za treniranje ovih modela je nekoliko sati za model 3, dok su model 1 i model 2 izvodili duže od 12 sati, odnosno model 1 skoro 24 sata, što je opet primarno do MLP-a. Analizom rezultata vidljivo je odstupanje rezultata prilikom unakrsne validacije i testiranja na setu za test te se može reći da je došlo do blagog pretjeranog prilagođavanja setu za test. Std kod $\overline{R^{2\prime}}$ se javlja u četvrtoj decimali, dok se kod $\overline{MAE'}$ *i* $\overline{RMSE'}$ javlja u drugoj decimali. To je zasigurno slučajnost jer su se najbolji modeli birali temeljem unakrsne validacije na setu za treniranje, set za testiranje se koristio samo prilikom evaluacije izvedbe skaliranja, uklanjanja neočekivanih vrijednosti i testiranja modela sa zadanim vrijednostima. Set za testiranje je više nego dvostruko manji od seta za treniranje te je moguće da slučajnom podjelom podataka upravo podaci iz manjeg seta bolje odgovaraju modelima. Testiranjem na novim i uistinu neviđenim podacima moglo bi se precizno detektirati o čemu je riječ, a ovako se može samo nagađati.

7. ZAKLJUČAK

U ovom radu je obrađen javno dostupan set podataka te su na njemu napravljene estimacije temperature, odnosno varijable pm, koristeći različite algoritme strojnog učenja. Kroz rad je ustanovljeno koji algoritmi imaju najveću točnost u estimaciji (MLP, DT, KNN) te se zatim nastavilo s implementacijom nasumičnog odabira hiperparametara, unakrsne validacije i tehnika ansambla. Upravo to su ključna pitanja ovoga rada te se sistematičnom obradom može dati odgovore na ta pitanja. Implementacijom nasumičnog odabira hiperparametara, odnosno paralelno upotrebom unakrsne validacije ili preciznije peterostruke unakrsne validacije dobiveni su još bolji modeli algoritama strojnog učenja. Odnosno, dobiveni su modeli evaluirani s $\overline{R^2} > 0.99$, što je više nego odličan rezultat, s obzirom da je maksimalan rezultat da R^2 teži k 1. Međutim, i dalje je bilo prostora za napredak te se implementacijom modela ET, koji je sam po sebi tehnika ansambla temeljena na stablima odlučivanja (DT), postigao mogući vrhunac ovog rada i $\overline{R^2} > 0.999$. Implementacijom *voting* ansambla s četiri modela (MLP, DT, KNN, ET) s nasumičnim hiperparametrima postignuti su odlični rezultati $\overline{R^2} = 0.997$, $\overline{MAE} = 0.614$, $\overline{RMSE} = 1.023$, ali se u toj fazi rada sakupilo dovoljno evaluiranih modela da je bilo moguće potrebno primijeniti najbolje modele s najboljim kombinacijama hiperparametara. Cilj je svakako bio dobiti još bolje rezultate, zbog čega je dodatno implementiran *Bagging* što je također ansambl tehnika, te se korištenjem *voting* ansambla s početnih četiri uz dodana četiri *Bagging* modela, koji koriste već spomenuta četiri osnovna modela, napravio *voting* ansambl s osam modela. Rezultati su bili očekivano dobri, ali se dodatnom analizom potvrdila sumnja da je MLP, iako je korišten od samog početka, upravo najslabija karika s kojom se nisu mogli doseći najbolji rezultati. *Voting* ansambl bez MLP-a je u konačnici dao najbolje rezultate te su analizirani rezultati unakrsne validacije, evaluacije na testu i srednje vrijednost rezultata unakrsne validacije i evaluacije na testu. Time su dobiveni rezultati $\overline{R^{2\prime}} = 0.99898$, $\overline{MAE'} = 0.207639$, $\overline{RMSE'} = 0.599132$. Svakako nije došlo do problema prekomjernog prilagođavanja setu za treniranje, već su donekle bolji rezultati na setu za test, što može upućivati na prekomjerno prilagođavanje tom setu. Međutim, sva analiza je temeljena na izboru hiperparametara uz unakrsnu validaciju, a prilikom tog procesa se koristio isključivo set za treniranje te je prilagođenost setu za testiranje temeljena slučajnom podjelom podataka iz ukupnog seta podataka.

Postavljene teze vezane za mogućnost točne estimacije temperature rotora PMSM su dokazane, te je dokazano da su itekako poboljšani modeli estimacija na kojima se primijenio postupak nasumičnog odabira hiperparametara te unakrsne validacije. Konačna primjena ansambl tehnika je donijela najbolje rezultate, iako je vidljiva dominacija DT, odnosno ansambla ET koji je dao najbolje rezultate.

8. LITERATURA

[1] Z. Wang, T. W. Ching, S. Huang, H. Wang, i T. Xu, „Challenges Faced by Electric Vehicle Motors and Their Solutions“, *IEEE Access*, sv. 9, str. 5228–5249, 2021, doi: 10.1109/ACCESS.2020.3045716.

[2] O. Wallscheid, „Thermal Monitoring of Electric Motors: State-of-the-Art Review and Future Challenges“, *IEEE Open Journal of Industry Applications*, sv. 2, str. 204–223, lip. 2021, doi: 10.1109/ojia.2021.3091870.

[3] O. Wallscheid, T. Huber, W. Peters, i J. Böcker, „Real-Time Capable Methods to Determine the Magnet Temperature of Permanent Magnet Synchronous Motors - A Review“.

[4] W. Kirchgassner, O. Wallscheid, i J. Bocker, „Estimating Electric Motor Temperatures with Deep Residual Machine Learning“, *IEEE Trans Power Electron*, sv. 36, izd. 7, str. 7480–7488, srp. 2021, doi: 10.1109/TPEL.2020.3045596.

[5] E. G. Gedlu, O. Wallscheid, i J. Böcker, „Permanent Magnet Synchronous Machine Temperature Estimation using Low-Order Lumped-Parameter Thermal Network with Extended Iron Loss Model Permanent Magnet Synchronous Machine Temperature Estimation using Low-Order Lumped-Parameter Thermal Network with Extended Iron Loss Model* * An extension of a full paper submitted to the International Conference on Power Electronics, Machines and Drives (PEMD) 2020“, doi: 10.36227/techrxiv.11671401.

[6] O. Wallscheid i J. Böcker, „Global identification of a low-order lumped-parameter thermal network for permanent magnet synchronous motors“, *IEEE Transactions on Energy Conversion*, sv. 31, izd. 1, str. 354–365, ožu. 2016, doi: 10.1109/TEC.2015.2473673.

[7] W. Kirchgässner, O. Wallscheid, i J. Böcker, „Thermal neural networks: Lumped-parameter thermal modeling with state-space machine learning“, *Eng Appl Artif Intell*, sv. 117, sij. 2023, doi: 10.1016/j.engappai.2022.105537.

[8] „Electric Motor Temperature | Kaggle“. Pristupljeno: 26. kolovoz 2023. [Na internetu]. Dostupno na: https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature

[9] C. R. Harris *i ostali*, „Array programming with NumPy“, *Nature 2020 585:7825*, sv. 585, izd. 7825, str. 357–362, ruj. 2020, doi: 10.1038/s41586-020-2649-2.

[10] „pandas-dev/pandas: Pandas“, doi: 10.5281/ZENODO.10045529.

[11] J. D. Hunter, „Matplotlib: A 2D graphics environment“, *Comput Sci Eng*, sv. 9, izd. 3, str. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[12] M. Waskom, „seaborn: statistical data visualization“, *J Open Source Softw*, sv. 6, izd. 60, str. 3021, tra. 2021, doi: 10.21105/joss.03021.

[13] L. Buitinck *i ostali*, „API design for machine learning software: experiences from the scikit-learn project“, ruj. 2013, [Na internetu]. Dostupno na: http://arxiv.org/abs/1309.0238

[14] F. Pedregosa *i ostali*, „Scikit-learn: Machine Learning in Python“, 2011. [Na internetu]. Dostupno na: http://scikit-learn.sourceforge.net.

[15] „Anaconda | The World’s Most Popular Data Science Platform“. Pristupljeno: 28. listopad 2023. [Na internetu]. Dostupno na: https://www.anaconda.com/

[16] „Home — Spyder IDE“. Pristupljeno: 28. listopad 2023. [Na internetu]. Dostupno na: https://www.spyder-ide.org/

[17] J. Liu i W. Chen, „Generalized DQ model of the permanent magnet synchronous motor based on extended park transformation“, *1st International Future Energy Electronics Conference, IFEEC 2013*, str. 885–890, 2013, doi: 10.1109/IFEEC.2013.6687627.

[18] „Statistics - Sampling, Variables, Design | Britannica“. Pristupljeno: 29. listopad 2023. [Na internetu]. Dostupno na: https://www.britannica.com/science/statistics/Experimental-design

[19] „Correlation Concepts, Matrix & Heatmap using Seaborn - Data Analytics“. Pristupljeno: 27. kolovoz 2023. [Na internetu]. Dostupno na: https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/

[20] J. W. Osborne i A. Overbay, „The power of outliers (and why researchers should ALWAYS check for them)“, *Practical Assessment, Research, and Evaluation*, sv. 9, izd. 6, 2004, doi: 10.7275/qf69-7k43.

[21] S. K. Kwak i J. H. Kim, „Statistical data preparation: Management of missing values and outliers“, *Korean Journal of Anesthesiology*, sv. 70, izd. 4. Korean Society of Anesthesiologists, str. 407–411, 01. kolovoz 2017. doi: 10.4097/kjae.2017.70.4.407.

[22] „Seaborn Distplot: A Comprehensive Guide | DigitalOcean“. Pristupljeno: 29. listopad 2023. [Na internetu]. Dostupno na: https://www.digitalocean.com/community/tutorials/seaborn-distplot

[23] „Understanding Boxplots: How to Read and Interpret a Boxplot | Built In“. Pristupljeno: 29. listopad 2023. [Na internetu]. Dostupno na: https://builtin.com/data-science/boxplot

[24] „Outlier Handling — 1.6.1“. Pristupljeno: 29. kolovoz 2023. [Na internetu]. Dostupno na: https://feature-engine.trainindata.com/en/latest/api_doc/outliers/index.html

[25] „sklearn.preprocessing.StandardScaler — scikit-learn 1.3.0 documentation“. Pristupljeno: 30. kolovoz 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[26] „sklearn.preprocessing.PowerTransformer — scikit-learn 1.3.0 documentation“. Pristupljeno: 30. kolovoz 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html

[27] „6.3. Preprocessing data — scikit-learn 1.3.0 documentation“. Pristupljeno: 30. kolovoz 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/preprocessing.html

[28] N. Anđelić, I. Lorencin, M. Glučina, i Z. Car, „Mean Phase Voltages and Duty Cycles Estimation of a Three-Phase Inverter in a Drive System Using Machine Learning Algorithms“, *Electronics (Switzerland)*, sv. 11, izd. 16, kol. 2022, doi: 10.3390/electronics11162623.

[29] M. Glučina, N. Anđelić, I. Lorencin, i S. Baressi Šegota, „Drive System Inverter Modeling Using Symbolic Regression“, *Electronics (Switzerland)*, sv. 12, izd. 3, velj. 2023, doi: 10.3390/electronics12030638.

[30] I. Lorencin, N. Andelić, V. Mrzljak, i Z. Car, „Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation“, *Energies (Basel)*, sv. 12, izd. 22, stu. 2019, doi: 10.3390/en12224352.

[31] Z. Car, S. Baressi Šegota, N. Anđelić, I. Lorencin, i V. Mrzljak, „Modeling the Spread of COVID-19 Infection Using a Multilayer Perceptron“, *Comput Math Methods Med*, sv. 2020, 2020, doi: 10.1155/2020/5714714.

[32] G. Bonaccorso, *Machine learning algorithms : reference guide for popular algorithms for data science and machine learning*. Pack Publishing, Birmingham, 2017.

[33] „sklearn.neural_network.MLPRegressor — scikit-learn 1.3.2 documentation“. Pristupljeno: 26. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html?highlight=mlp%20regressor#sklearn.neural_network.MLPRegressor

[34] „1.10. Decision Trees — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/tree.html#decision-trees

[35] „sklearn.tree.DecisionTreeRegressor — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor

[36] „sklearn.neighbors.KNeighborsRegressor — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor

[37] „1.6. Nearest Neighbors — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/neighbors.html#classification

[38] „3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/cross_validation.html

[39] „sklearn.model_selection.cross_val_score — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html#sklearn.model_selection.cross_val_score

[40] „sklearn.model_selection.cross_validate — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html#sklearn.model_selection.cross_validate

[41] L. Plonsky i H. Ghanbar, „Multiple Regression in L2 Research: A Methodological Synthesis and Guide to Interpreting R2 Values“, *Modern Language Journal*, sv. 102, izd. 4, str. 713–731, pros. 2018, doi: 10.1111/modl.12509.

[42] „3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/model_evaluation.html

[43] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, i C. H. Lee, „On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression“, *IEEE Signal Process Lett*, sv. 27, str. 1485–1489, 2020, doi: 10.1109/LSP.2020.3016837.

[44] T. Chai i R. R. Draxler, „Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature“, *Geosci Model Dev*, sv. 7, izd. 3, str. 1247–1250, lip. 2014, doi: 10.5194/gmd-7-1247-2014.

[45] „3.2. Tuning the hyper-parameters of an estimator — scikit-learn 1.3.2 documentation“. Pristupljeno: 30. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/grid_search.html

[46] „1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking — scikit-learn 1.3.2 documentation“. Pristupljeno: 31. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles

[47] „sklearn.ensemble.ExtraTreesRegressor — scikit-learn 1.3.2 documentation“. Pristupljeno: 31. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html#sklearn.ensemble.ExtraTreesRegressor

[48] L. Breiman, „Bagging Predictors“. U: Machine Learning. Kluwer Academic Publishers, Boston, 1996.

[49] „sklearn.ensemble.BaggingRegressor — scikit-learn 1.3.2 documentation“. Pristupljeno: 31. listopad 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegressor

[50] „10. Common pitfalls and recommended practices — scikit-learn 1.3.2 documentation“. Pristupljeno: 01. studeni 2023. [Na internetu]. Dostupno na: https://scikit-learn.org/stable/common_pitfalls.html

9. POPIS OZNAKA

PMSM – *permanent magnet synchronous motor* (sinkroni motor s permanentnim magnetima)

LPTN – *lumped paramether thermal networks* (toplinske mreže s grupiranim parametrima)

NN – *neural network* (neuronska mreža)

PM – *permanent magnet* (permanentni magnet)

MLP – *multilayered perceptron* (višeslojni perceptron)

R2 – koeficijent determinancije

MAE – *mean absolute error* (srednja apsolutna pogreška)

RMSE – *root mean square error* (prosječna kvadratna pogreška)

KDE – *kernel density estimation*

IQR – *interquartile range* (interkvartilni raspon)

ANN – *artificial neural network* (umjetna neuronska mreža)

Relu – *rectified linear unit*

Sgd – *stohastic gradient descent*

Adam – *adaptive moment estimation*

DT – *decision tree* (stablo odluke)

KNN – *K-Nearest Neighbours* (K-najbliži susjedi)

CV – *cross validation* (unakrsna validacija)

ET – *extremely randomised trees*

Bagging – *bootstrap agreggating*

STD – *standard deviation* (standardna devijacija)

10. SAŽETAK I KLJUČNE RIJEČI

U ovom radu opisan je značaj točne estimacije temperature rotora PMSM-a za automobilsku primjenu zbog utjecaja na učinkovitost, performanse i trajnost stroja. Odrađen je pregled literature koja je estimirala temperaturu PMSM te je predloženo korištenje umjetne inteligencije, odnosno naprednih algoritama strojnog učenja. U radu je korišten programski jezik Python te biblioteke Numpy, Pandas, Matplotlib, Scikit-Learn. Analiziran je javno dostupan set podataka koji sadrži mjerene varijable PMSM te je napravljena predobrada, korelacijska analiza, skaliranje seta podataka i uklanjanje netipičnih vrijednosti.

Početna razmatranja odrađena su korištenjem MLP, a zatim su testirani razni algoritmi strojnog učenja sa zadanim postavkama. Testirani su na obrađenom setu podataka koristeći podjelu seta podataka na set za treniranje i set za testiranje uz evaluacijske metrike R^2, MAE i RMSE te su MLP, DT i KNN postigli dobre rezultate. Implementirana je peterostruka unakrsna validacija i nasumično pretraživanje hiperametara kako bi se poboljšala izvedba izdvojenih algoritama.

Zbog dodatnog poboljšanja implementirane su ansambl tehnike poput ET, *Bagging* i *voting* ansambla, te su postignuti odlični rezultati. Verzije *voting* ansambla koristile su četiri do osam modela te je model evaluiran srednjom vrijednosti metrika prilikom unakrsne validacije na setu za treniranje i testu na setu za test te su postignuti izvrsni rezultati.

Ključne riječi: *umjetna inteligencija, strojno učenje, neuronske mreže, napredni algoritmi strojnog učenja, PMSM, Python, Pandas, Scikit-Learn, MLP, DT, KNN, ET, unakrsna validacija, nasumično pretraživanje hiperparametara, bagging, voting ansambl*

11. SUMMARY AND KEYWORDS

In this paper, the significance of accurate rotor temperature estimation for PMSM in automotive applications is described, due to its impact on machine efficiency, performance, and durability. A literature review of PMSM temperature estimation was conducted, proposing the use of artificial intelligence, specifically advanced machine learning algorithms. The Python programming language and libraries such as Numpy, Pandas, Matplotlib, and Scikit-Learn are utilized in the research. A publicly available dataset containing measured PMSM variables is analysed, involving data preprocessing, correlation analysis, dataset scaling, and the handling of outliers.

Initial investigations were performed using MLP. Subsequently, various machine learning algorithms with default settings were tested on the pre-processed dataset, using a split into training and testing sets. The evaluation metrics R^2, MAE, and RMSE were employed, where MLP, DT, and KNN yield favourable results. To enhance the performance of selected algorithms, five-fold cross-validation and random hyperparameter tuning are implemented.

Additional improvements are achieved by employing ensemble techniques such as ET, Bagging, and a Voting ensemble. Excellent results are obtained, with the Voting ensemble employing four to eight models and being evaluated based on the mean values of metrics during cross-validation and testing. This approach results in outstanding performance.

Keywords: *artificial intelligence, machine learning, neural networks, advanced machine learning algorithms, PMSM, Python, Pandas, Scikit-Learn, MLP, DT, KNN, ET, cross-validation, random hyperparameter search, bagging, voting ensemble*.

POPIS SLIKA

POPIS TABLICA

DODATAK A – NASUMIČNO PRETRAŽIVANJE HIPERPARAMETARA I UNAKRSNA VALIDACIJA ZA VOTING ANSAMBL

```
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 15 11:27:56 2023

@author: tstim
"""

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from feature_engine.outliers import OutlierTrimmer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
import random as random
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import VotingRegressor, ExtraTreesRegressor
from sklearn.neighbors import KNeighborsRegressor


def Outlier_Trim(df):
    trimmer = OutlierTrimmer(
    capping_method='gaussian', tail='both', fold=2.8,
    variables=list(df.columns))
    data_trim = pd.DataFrame(trimmer.fit_transform(df),
                             columns = list(df.columns))
    df = data_trim
    return df

def Scaler (X_train, X_test):
    scaler = StandardScaler()
    scaler.fit(X_train)
    columns = list(X_train.columns)
    index = list(X_train.index)
    index2 = list(X_test.index)
    X_train = pd.DataFrame(scaler.transform(X_train),columns = columns,
                           index = index)
    X_test = pd.DataFrame(scaler.transform(X_test), columns = columns,
                          index = index2)
    return X_train, X_test

def MLPRandomSearch():
    mlpParam = []
    HLS = tuple(np.random.randint(low=15, high=100,
                                  size=np.random.randint(1, 5)))
    activation = random.choice(['identity', 'logistic', 'tanh', 'relu'])
    solver = random.choice(['adam', 'sgd', 'lbfgs'])
    alpha = 10 ** np.random.uniform(low=-5, high=-1)
    max_iter = 1500
    mlpParam.append(HLS)
    mlpParam.append(activation)
    mlpParam.append(solver)
    mlpParam.append(alpha)
```

```
    mlpParam.append(max_iter)
    with open('Param.txt', 'a') as file:
         file.write("\n")
         file.write("Random MLP Parameters:\n")
         file.write(f"Hidden Layer Sizes: {mlpParam[0]}\n")
         file.write(f"Activation: {mlpParam[1]}\n")
         file.write(f"Solver: {mlpParam[2]}\n")
         file.write(f"Alpha: {mlpParam[3]}\n")
         file.write(f"Max Iterations: {mlpParam[4]}\n")
    if solver == 'sgd':
        learning_rate = random.choice(['constant','invscaling',
'adaptive'])
        learning_rate_init = 10 ** np.random.uniform(low=-4, high=-2)
        n_iter_no_change = 12
        shuffle = bool(np.random.choice([0, 1]))
        mlpParam.append(learning_rate)
        mlpParam.append(learning_rate_init)
        mlpParam.append(n_iter_no_change)
        mlpParam.append(shuffle)
        with open('Param1.txt', 'a') as file:
            file.write(f"learning_rate: {mlpParam[5]}\n")
            file.write(f"learning_rate_init: {mlpParam[6]}\n")
            file.write(f"n_iter_no_change: {mlpParam[7]}\n")
            file.write(f"shuffle: {mlpParam[8]}\n")
        print("MLP with sgd solver chosen")
        Mlpregr = MLPRegressor(hidden_layer_sizes=mlpParam[0],
                               activation=mlpParam[1],
                               solver=mlpParam[2],
                               alpha=mlpParam[3],
                               max_iter=mlpParam[4],
                               learning_rate = mlpParam[5],
                               learning_rate_init=mlpParam[6],
                               n_iter_no_change=mlpParam[7],
                               shuffle=mlpParam[8],
                               verbose=True, random_state=42)
    elif solver == 'adam':
        learning_rate_init = 10 ** np.random.uniform(low=-4, high=-2)
        n_iter_no_change = 12
        shuffle = bool(np.random.choice([0, 1]))
        beta_1 = 0.9
        beta_2 = 0.999
        mlpParam.append(learning_rate_init)
        mlpParam.append(n_iter_no_change)
        mlpParam.append(shuffle)
        mlpParam.append(beta_1)
        mlpParam.append(beta_2)
        with open('Param.txt', 'a') as file:
            file.write(f"learning_rate_init: {mlpParam[5]}\n")
            file.write(f"n_iter_no_change: {mlpParam[6]}\n")
            file.write(f"shuffle: {mlpParam[7]}\n")
            file.write(f"beta_1: {mlpParam[8]}\n")
            file.write(f"beta_2: {mlpParam[9]}\n")
        print("MLP with adam solver chosen")
        Mlpregr = MLPRegressor(hidden_layer_sizes=mlpParam[0],
                               activation=mlpParam[1],
                               solver=mlpParam[2],
                               alpha=mlpParam[3],
                               max_iter=mlpParam[4],
                               learning_rate_init=mlpParam[5],
                               n_iter_no_change=mlpParam[6],
                               shuffle=mlpParam[7],
```

```
                            beta_1=mlpParam[8],
                            beta_2=mlpParam[9],
                            verbose=True, random_state=42)
    elif solver == 'lbfgs':
        max_fun = random.randint(10000, 20000)
        mlpParam.append(max_fun)
        with open('Param.txt', 'a') as file:
            file.write(f"max_fun: {mlpParam[5]}\n")
        print("MLP with lbfgs solver chosen")
        Mlpregr = MLPRegressor(hidden_layer_sizes=mlpParam[0],
                               activation=mlpParam[1],
                               solver=mlpParam[2],
                               alpha=mlpParam[3],
                               max_iter=mlpParam[4],
                               max_fun = mlpParam[5],
                               verbose=True, random_state=42)
    return Mlpregr

def DTRandomSearch():
    dtParam = []
    criterion = random.choice(['squared_error', 'friedman_mse',
                               'absolute_error','poisson'])
    splitter = random.choice(['best', 'random'])
    max_depth = random.randint(5, 30)
    min_samples_split = random.randint(2, 10)
    min_samples_leaf = random.randint(1, min_samples_split - 1)
    max_features = None
    dtParam.append(criterion)
    dtParam.append(splitter)
    dtParam.append(max_depth)
    dtParam.append(min_samples_split)
    dtParam.append(min_samples_leaf)
    dtParam.append(max_features)
    with open('Param.txt', 'a') as file:
        file.write("\n")
        file.write("Random DT Parameters:\n")
        file.write(f"Criterion: {dtParam[0]}\n")
        file.write(f"Splitter: {dtParam[1]}\n")
        file.write(f"Max Depth: {dtParam[2]}\n")
        file.write(f"Min Samples Split: {dtParam[3]}\n")
        file.write(f"Min Samples Leaf: {dtParam[4]}\n")
        file.write(f"Max Features: {dtParam[5]}\n")
    DTregr = DecisionTreeRegressor(criterion=dtParam[0],
                               splitter=dtParam[1],
                               max_depth=dtParam[2],
                               min_samples_split=dtParam[3],
                               min_samples_leaf=dtParam[4],
                               max_features=dtParam[5],
                               random_state=42)
    return DTregr

def ETRandomSearch():
    etParam = []
    n_estimators = random.randint(10,150)
    criterion = random.choice(['squared_error', 'absolute_error',
                               'friedman_mse', 'poisson'])
    max_depth = random.randint(5, 50)
    min_samples_split = random.randint(2, 10)
    min_samples_leaf = random.randint(1, min_samples_split - 1)
    max_features = None
    etParam.append(n_estimators)
```

```
    etParam.append(criterion)
    etParam.append(max_depth)
    etParam.append(min_samples_split)
    etParam.append(min_samples_leaf)
    etParam.append(max_features)
    with open('Param.txt', 'a') as file:
        file.write("\n")
        file.write("Random ET Parameters:\n")
        file.write(f"N Estimators: {etParam[0]}\n")
        file.write(f"Criterion: {etParam[1]}\n")
        file.write(f"Max Depth: {etParam[2]}\n")
        file.write(f"Min Samples Split: {etParam[3]}\n")
        file.write(f"Min Samples Leaf: {etParam[4]}\n")
        file.write(f"Max Features: {etParam[5]}\n")
    ETregr = ExtraTreesRegressor(n_estimators=etParam[0],
                                 criterion=etParam[1],
                                max_depth=etParam[2],
                                min_samples_split=etParam[3],
                                min_samples_leaf=etParam[4],
                                max_features=etParam[5],
                                random_state=42)
    return ETregr

def KNNRandomSearch():
    knnParam = []
    n_neighbors = random.randint(1, 50)
    weights = random.choice(['uniform', 'distance'])
    algorithm = random.choice(['auto', 'ball_tree', 'kd_tree', 'brute'])
    leaf_size = random.randint(10, 50)
    knnParam.append(n_neighbors)
    knnParam.append(weights)
    knnParam.append(algorithm)
    knnParam.append(leaf_size)
    with open('Param.txt', 'a') as file:
        file.write("\n")
        file.write("Random KNN Parameters:\n")
        file.write(f"N Neighbors: {knnParam[0]}\n")
        file.write(f"Weights: {knnParam[1]}\n")
        file.write(f"Algorithm: {knnParam[2]}\n")
        file.write(f"Leaf Size: {knnParam[3]}\n")
    KNNregr = KNeighborsRegressor(n_neighbors=knnParam[0],
                                  weights=knnParam[1],
                                  algorithm=knnParam[2],
                                  leaf_size=knnParam[3])
    return KNNregr


dff = pd.read_csv("measures_v2.csv")
df = dff.drop('profile_id', axis=1)
df_desc = df.describe().T

df = Outlier_Trim(df)
df_trim_desc = df.describe().T

y = df.pop("pm")

X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.3,
                                                    random_state=42)

X_train, X_test = Scaler(X_train, X_test)
```

```
voting_mean = 0

while voting_mean < 0.999:
    mlp = MLPRandomSearch()
    dt = DTRandomSearch()
    et = ETRandomSearch()
    knn = KNNRandomSearch()

    # CV za svaki model
    mlp_scores = cross_val_score(mlp, X_train, y_train, cv=5, scoring='r2')
    dt_scores = cross_val_score(dt, X_train, y_train, cv=5, scoring='r2')
    et_scores = cross_val_score(et, X_train, y_train, cv=5, scoring='r2')
    knn_scores = cross_val_score(knn, X_train, y_train, cv=5, scoring='r2')

    mlp_mean, mlp_std = mlp_scores.mean(), mlp_scores.std()
    dt_mean, dt_std = dt_scores.mean(), dt_scores.std()
    et_mean, et_std = et_scores.mean(), et_scores.std()
    knn_mean, knn_std = knn_scores.mean(), knn_scores.std()

    print("MLP CV R2 Mean:", mlp_mean)
    print("MLP CV R2 Std:", mlp_std)
    print("DT CV R2 Mean:", dt_mean)
    print("DT CV R2 Std:", dt_std)
    print("ET CV R2 Mean:", et_mean)
    print("ET CV R2 Std:", et_std)
    print("KNN CV R2 Mean:", knn_mean)
    print("KNN CV R2 Std:", knn_std)

    voting_regressor = VotingRegressor(estimators=[
        ('mlp', mlp),
        ('dt', dt),
        ('et', et),
        ('knn', knn)
        ])

    voting_scores = cross_val_score(voting_regressor, X_train, y_train,
                                    cv=5, scoring='r2')

    # Mean i std voting
    voting_mean, voting_std = voting_scores.mean(), voting_scores.std()

    print("Voting Regressor CV R2 Mean:", voting_mean)
    print("Voting Regressor CV R2 Std:", voting_std)

    with open('Voting.txt', 'a') as file:
        file.write("\n")
        file.write("Cross-Validation Results:\n")
        file.write(f"MLP CV R2 Mean: {mlp_mean}\n")
        file.write(f"MLP CV R2 Std: {mlp_std}\n")
        file.write(f"DT CV R2 Mean: {dt_mean}\n")
        file.write(f"DT CV R2 Std: {dt_std}\n")
        file.write(f"ET CV R2 Mean: {et_mean}\n")
        file.write(f"ET CV R2 Std: {et_std}\n")
        file.write(f"KNN CV R2 Mean: {knn_mean}\n")
        file.write(f"KNN CV R2 Std: {knn_std}\n")
        file.write(f"Voting Regressor CV R2 Mean: {voting_mean}\n")
        file.write(f"Voting Regressor CV R2 Std: {voting_std}\n")
```