Redukcija dimenzionalnosti podataka o mjerenjima tlaka kod oštećenja vodovodnih cijevi

Buterin, Ivan

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Redukcija dimenzionalnosti podataka o mjerenjima tlaka kod oštećenja vodovodnih cijevi

Rijeka, studeni 2024.

Ivan Buterin
0069082632

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Redukcija dimenzionalnosti podataka o mjerenjima tlaka kod oštećenja vodovodnih cijevi

Mentor: prof. dr. sc. Zoran Čarija
Komentor: v. asist. dr. sc. Ivana Lučin

Rijeka, studeni 2024.

Ivan Buterin
0069082632

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, studeni 2024.

_____
Ivan Buterin

Zahvala

Zahvaljujem mentoru prof. dr. sc. Zoranu Čariji i komentorici v. asist. dr. sc. Ivani Lučin na podršci tijekom pisanja ovoga rada i korisnim raspravama i savjetima.
Zahvaljujem svojoj obitelji, prijateljima i kolegama na podršci tijekom studiranja.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Vodovodni sustav predstavlja temeljnu infrastrukturu svakog urbanog naselja, osiguravajući dostupnost pitke vode i opskrbu kućanstava, industrije i javnih ustanova. Kvalitetna opskrba vodom jedan je od ključnih pokazatelja razvijenosti društva i odražava tehničku, ekološku i ekonomsku održivost lokalne zajednice. Vodovodne mreže diljem svijeta suočavaju se s brojnim izazovima i problemima, među kojima je posebno izražen problem oštećenja vodovodnih cijevi, gubitka vode i posljedičnih financijskih troškova.

Ovakvi problemi postali su česti u mnogim urbanim sredinama, gdje zastarjeli sustavi ne mogu uvijek podnijeti nove zahtjeve moderne urbane infrastrukture. Osim tehničkih izazova, financijski troškovi povezani s gubitkom vode i održavanjem infrastrukture postaju sve veći teret za lokalne zajednice.

Razlog istraživanja problema oštećenja vodovodnih cijevi i gubitka vode leži u hitnoći pronalaženja učinkovitih rješenja za smanjenje tih gubitaka. Gubitci vode, koji nastaju zbog curenja cijevi, ne samo da predstavljaju ozbiljan ekološki problem, već uzrokuju i značajne financijske troškove. Prema dostupnim podacima, u prosjeku se u Hrvatskoj gubi oko 50% vode iz vodovodnog sustava prije nego što ona stigne do krajnjih potrošača. Takvi gubitci vode opterećuju ne samo održavanje infrastrukture, već i same korisnike vode u vidu povećanih troškova. S obzirom na rastuću potrebu za održivim upravljanjem vodnim resursima, potrebno je razumjeti uzroke ovih problema kako bi se mogla primijeniti moderna i učinkovita rješenja.

Oštećenja vodovodnih cijevi mogu nastati iz više razloga. Najčešći su tehničke prirode, uključujući starost cijevi, lošu izvedbu instalacija, upotrebu nekvalitetnih

materijala te vanjski utjecaji poput seizmičkih aktivnosti ili promjena u tlu. Uz to, nepravilan nadzor i održavanje infrastrukture mogu dovesti do zanemarivanja manjih kvarova koji se postupno razvijaju u veća oštećenja. Osim toga, vremenski uvjeti, poput ekstremnih temperatura ili promjena u razini podzemnih voda, također mogu utjecati na strukturu cijevi i povećati njihovu podložnost oštećenjima. Gubitci vode u vodovodnim sustavima imaju dalekosežne posljedice. Osim očiglednih ekoloških problema i rasipanja resursa, ti gubitci uzrokuju i povećanje troškova u vodoopskrbi. Naime, oštećenja cijevi često zahtijevaju skupe popravke, dok troškovi prerade vode koja nikad ne dospije do korisnika povećavaju cijenu vodoopskrbe. Ti troškovi, koji se na kraju prenose na potrošače, mogu imati značajan utjecaj na gospodarstvo zajednice, naročito u manjim gradovima i općinama.

Također, dugotrajni gubitci vode mogu uzrokovati smanjenje pritiska u vodovodnoj mreži, što može imati izravne posljedice na kvalitetu života građana i funkcioniranje poslovnih subjekata. Ipak, u konačnici do najvećih problema dolazi u trenutku kada oštećenja cijevi dovedu do puknuća cijevi. Tada dolazi do kompletnog prekida vodoopskrbe, a može doći i do oštećenja okolne infrastrukture na mjestu puknuća kao što je poplavljivanje objekata. Upravo zbog ovih izazova, istraživanje problema oštećenja vodovodnih cijevi i gubitaka vode ključno je za budući razvoj vodovodnih sustava. Cilj istraživanja je predložiti konkretna rješenja za smanjenje gubitaka, poboljšanje održavanja infrastrukture i smanjenje financijskih troškova.

U modernim sustavima upravljanja i održavanja vodovodnih mreža, sve veći naglasak stavlja se na korištenje naprednih tehnologija i velikih skupova podataka za praćenje i predviđanje kvarova. Međutim, jedan od izazova s kojim se inženjeri i analitičari suočavaju jest upravljanje predimenzioniranim skupovima podataka. Nadzor vodovodnih sustava može generirati veliku količinu informacija iz različitih izvora – senzori za praćenje protoka i tlaka, povijesni podaci o kvarovima, geološki podaci o terenu te mnogi drugi. Ovakva količina podataka može biti teško obradiva, a višak nepotrebnih ili redundantnih informacija može otežati donošenje preciznih odluka.

Cilj ovog rada je ispitati učinkovitost metoda redukcija dimenzionalnosti podataka, odnosno hoće li njihovom primjenom algoritam strojnog učenja treniran na sažetom i reduciranom skupu podataka pružiti bolje performanse (npr. kraće vrijeme izvršavanja) i utjecaj na točnost predikcijskog modela u odnosu na originalni set podataka.

U sljedećem poglavlju detaljnije će se opisati karakteristike vodoopskrbnog sustava i problema kojem se pokušava pristupiti. U trećem poglavlju prikazat će se svi softverski alati korišteni u analizi i rješavanju ovog problema. Također, u nastavku rada govorit će se o strojnom učenju i specifičnom algoritmu pomoću kojeg su se vršile manipulacije nad podacima te dobivenim rezultatima. Na istim podacima provest će se odabrane metode redukcije dimenzionalnosti podataka te analizirati njihove performanse i rezultati.

Poglavlje 2

Opis problema i ulazni podaci

U vodoopskrbnom sustavu, kontinuirano očitavanje tlakova može generirati veliki broj podataka. Prikupljeni podaci mogu imati visoku dimenzionalnost koja otežava analizu i interpretaciju podataka. Navedeno može biti od velike važnosti u incidentnim situacijama detekcije primjena u mreži kao što su pojave oštećenja i osobito puknuća cijevi. Puknuća cijevi predstavljaju najozbiljniji problem u kojem je potrebna što brza detekcija lokacije puknuća kako bi se što prije saniralo oštećenje i uspostavila normalna vodoopskrba. Lokacije puknuća cijevi je nešto lakše lokalizirati jer se poremećaji najčešće mogu primjetiti na površini, dok kod oštećenja cijevi to ne mora biti slučaj. Oštećenja cijevi se mogu detektirati kao poremećaji u mjerenjima tlaka u sustavu no zbog velike količine podataka teško je identificirati vrijedne podatke na temelju kojih se mogu identificirati potencijalne lokacije oštećenja u vodoopskrbnoj mreži.

Predmetni rad nastavak je istraživanja provedenog u [1]. U navedenom radu predložena je ideja o treniranju modela strojnog učenja pomoću velikog sintentičkih podataka dobivenih simuliranjem rada sustava prilikom različitih scenarija oštećenja cijevi. U navedenom radu uočeno je to da se radi o podacima visoke dimenzionalnosti zbog kojih nije moguće daljnje povećanje broja ulaznih podataka.

U tom slučaju, moguće je smanjenje dimenzionalnost podataka pomoću metoda redukcije dimenzionalnosti, što omogućava pregledniji i sažetiji prikaz ključnih informacija o radu sustava. Veliki broj čvorova i podataka o tlakovima u svakom trenutku povećava složenost, a istovremeno mnoge informacije mogu biti redundantne. Na primjer, tlakovi u susjednim čvorovima često su povezani zbog protoka vode kroz mrežu,

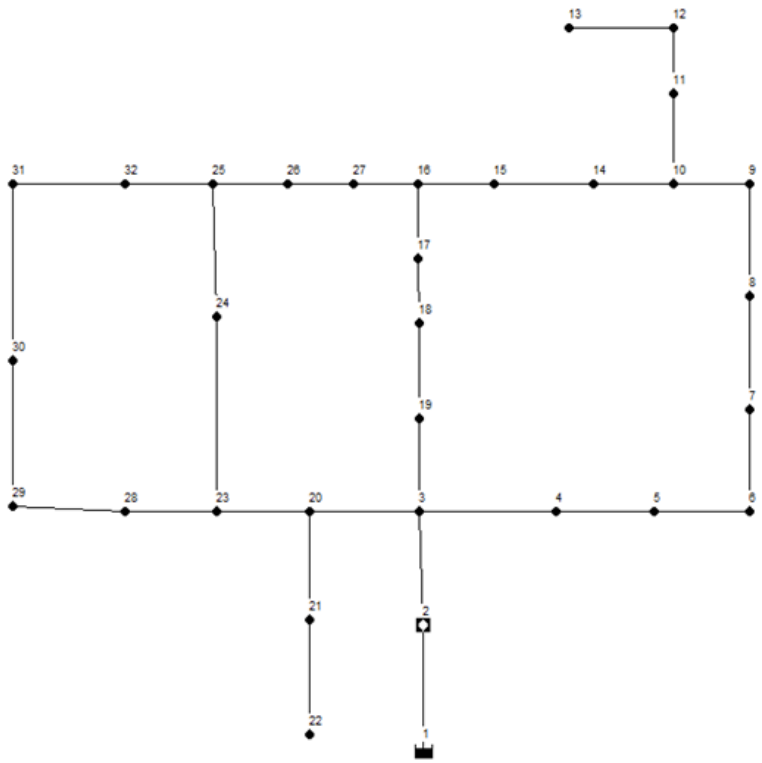
što znači da postoji korelacija između očitanja tlakova u tim čvorovima. Zbog toga su neke informacije unutar skupa podataka suvišne i moguće ih je sažeti.

Najvažniji ciljevi redukcije dimenzionalnosti podataka su smanjenje računalne složenosti (lakša računalna obrada i brža analiza podataka), identifikacija glavnih značajki podataka koje najbolje opisuju stanje mreže koristeći sažetije reprezentacije, pojednostavljena interpretacija podataka (bolje razumijevanje ključnih obrazaca i ponašanja sustava) te brže donošenje odluka (u realnom vremenu, brza analiza omogućava pravovremene reakcije na potencijalne probleme u mreži).

Metoda redukcije dimenzionalnosti podataka bit će testirana na primjeru vodovodne mreže sastavljene od jednog spremnika vode i 31 čvora. Čvorovi su ključne točke u mreži kroz koje voda teče prema potrošačima. U svakoj od točaka sustava moguće je definirati zahtjeve potrošača za vodom. Također,svakoj od tih točaka moguće je očitavati tlakove koji pružaju informacije o stanju u mreži, kao što su protok vode, potencijalne anomalije ili gubitci. Ovisno o duljini i promjeru vodova, kao i o promjenjivim potrebama za vodom u različitim dijelovima mreže, tlakovi u čvorovima mogu značajno varirati. U ovom slučaju, očitanja tlakova u određenim vremenskim intervalima vrše se pomoću senzora postavljenih na 2 čvora. Postavljanje senzora na svakom čvoru mreže iziskivalo bi velike troškove, stoga je u realnosti broj senzora daleko manji od broja čvorova. Upravo navedeno predstavlja ograničenje u tehnikama detekcije i lokalizacije oštećenja, jer na temelju ograničenih podataka je potrebno odrediti informacije o lokacijama koje je potrebno pronaći. Moguće je detektirati i dodatni problem u kojem pozicija samih senzora može značajno utjecati na dobivanje rješenja, tj. poremećaji u sustavu koji su bliže samim senzorima će biti lakše detektirani od poremećaja koji su na udaljenijim djelovima mreže.

Spremnik predstavlja izvor vode s konstantnim tlakom, što znači da simulira idealan uvjet bez gubitka ili fluktuacija na izvoru. U praksi, ovo je idealizacija stvarnog sustava, no omogućava fokus na analizu tlakova i potrošnje vode u samoj mreži [1]. Analizirana vodovodna mreža prikazana je na slici 2.1.

Ipak, prilikom primjene metoda redukcije dimenzionalnosti na podatke o tlakovima u vodovodnoj mreži, suočavamo se s nekoliko izazova kao što su gubitak preciznosti, odnosno rizik da se pri reduciranju dimenzionalnosti izgube neke specifične informacije koje bi mogle biti ključne za precizno otkrivanje lokalnih problema, poput curenja vode ili oštećenja u pojedinim dijelovima mreže. Nadalje, tlakovi u



Slika 2.1 Promatrana vodovodna mreža

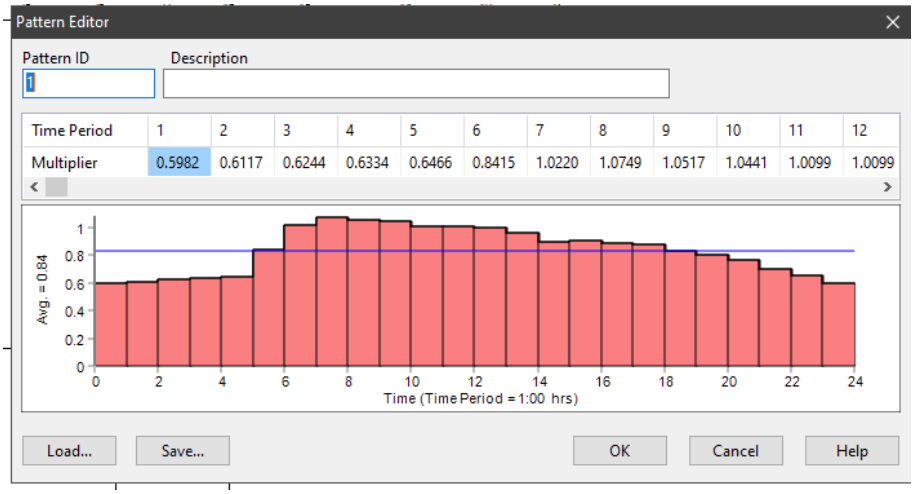
mreži mogu značajno varirati ovisno o udaljenosti čvora od spremnika, promjenama u potrošnji, vremenskim uvjetima ili eventualnim kvarovima u mreži. Takva heterogenost podataka zahtijeva pažljiv odabir metoda za redukciju dimenzionalnosti, osobito ako postoje nelinearni odnosi u podacima. S obzirom da su vodovodne mreže dinamički sustavi, a očitanja tlakova se mijenjaju ovisno o promjenama u uvjetima i protoku, vrlo je važno sačuvati dinamičke karakteristike. Kod primjene metoda za redukciju dimenzionalnosti, važno je zadržati mogućnost praćenja promjena i dinamičkih obrazaca u mreži kako bi se pravovremeno moglo reagirati na odstupanja.

Metode redukcije dimenzionalnosti koje će biti korištene i testirane su *Principal component analysis* (PCA) i ANOVA f-test. Navedene metode biti će detaljnije

opisane u narednim poglavljima.

2.1 Ulazni podaci

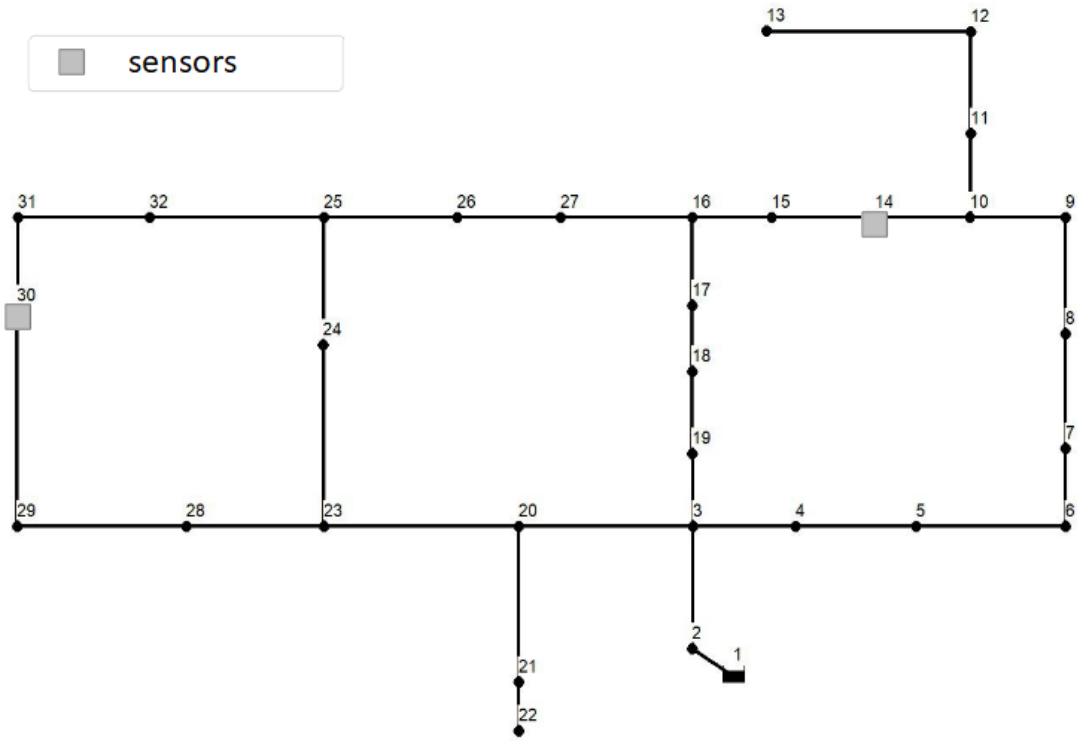
Originalni ulazni podaci spremljeni su u četiri tekstualne datoteke koje sadrže očitanja tlakova dvaju senzora u mreži. Podaci su dobiveni na temelju velikog broja simulacija provedenih u programu EPANET koji služi za modeliranje vodoopskrbnih sustava na način da se definira lokacija i veličina oštećenja te očitaju podaci sa senzora. Simulacije prikazuju promjene potrošnje u sustavu. (slika 2.2).



Slika 2.2 Promjena potrošnje u sustavu na primjeru jednog čvora

Senzori su postavljeni na čvorove 14 i 30 (slika 2.3), što znači da polovina svakog retka u input datoteci odgovara očitanjima senzora 14, dok druga polovica predstavlja očitanja sa senzora 30. Relevantna vremenska diskretizacija stanja u mreži bila je jedan sat, no s obzirom da su očitanja na senzorima prikupljana s razmakom od petnaest minuta, pojavila se određena redundantnost podataka u obliku četiri jednaka očitanja unutar jednog sata, zbog čega je podatke bilo potrebno pročistiti prije primjene modela strojnog učenja i metode redukcije podataka. Slika 2.4 prikazuje stanje jednog čvora mreže u određenom trenutku.

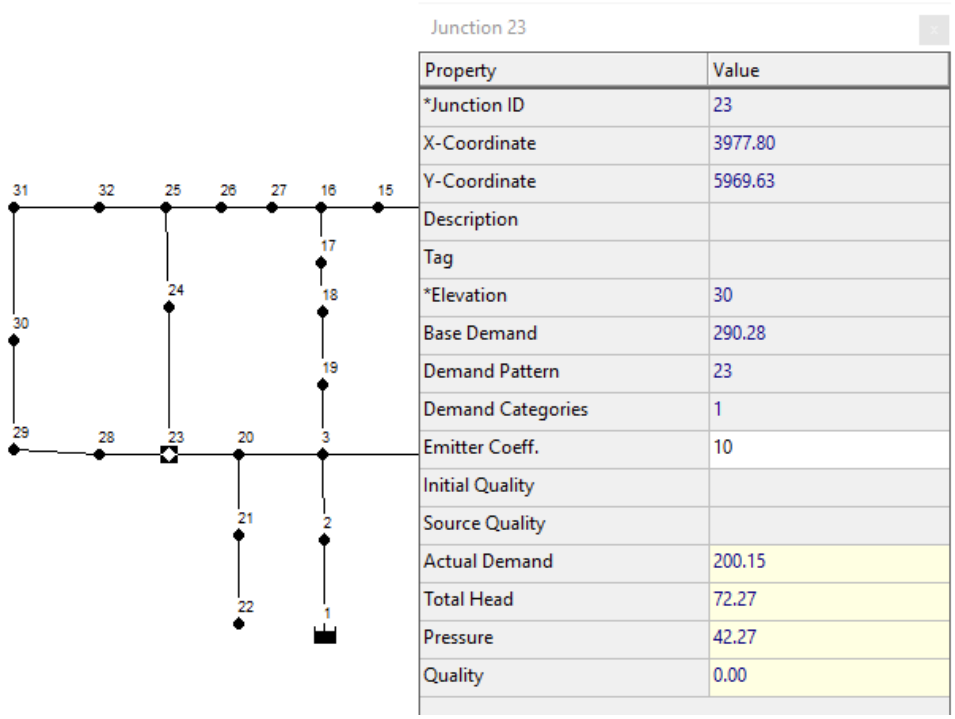
S obzirom da količina vode koja protječe kroz oštećenje u cijevi ovisi o tlaku u sustavu (što je tlak veći, protječe više vode) tijekom rada sustava teško je definirati



Slika 2.3 Lokacije senzora u vodoopskrbnom sustavu.

koliki su gubitci zbog konstantne promjene tlaka u sustavu. Iz tog razloga oštećenja u sustavu se ne modeliraju definiranjem protoka u čvoru već se koristi bezdimenzijska varijabla *emitter_coefficient* koja predstavlja veličinu otvora kroz koji voda izlazi iz sustava, tj. veličinu oštećenja.

Kako bi se dobila velika količina ulaznih podataka za strojno učenje, vrijednost navedene varijable se nasumično mijenjala u rasponu od 5 do 15. Veća vrijednost koeficijenta predstavlja veći otvor, tj. veće gubitke u sustavu te se u tom slučaju očekuju veći poremećaji u sustavu. U slučaju manje vrijednosti koeficijenta događa se suprotno. Manja vrijednost predstavlja manje oštećenje te se očekuju manji poremećaji u tlakovima koji se posljedično nešto teže mogu detektirati. Osim veličine oštećenja varirala se i lokacija oštećenja. Važno je napomenuti da se koristilo pojednostavljenje u kojem se oštećenje može pojaviti samo u jednom čvoru. U realnosti broj oštećenja u mreži može biti veći. S obzirom na to da numerički model predstav-



Slika 2.4 Stanje čvora mreže u određenom trenutku

lja procjenu rada sustava, u realnosti potrošnje u sustavu odstupaju od modeliranih vrijednosti. Iz tog razloga promatrani su različiti slučajevi gdje se za svaki čvor vodovodne mreže još dodatno izvršila provjera nasumičnim odabirom hoće li se vrijednost potrošnje u čvoru mijenjati, ukoliko se odabire mijenjanje tada se potrošnja u čvoru povećava ili smanjuje za nasumično odabrani broj. Na temelju navedenog ispitivali su se rasponi $\pm$ 2.5%, $\pm$ 5% i $\pm$ 10%.

Poglavlje 3

Korišteni alati

3.1 Programski jezik Python

Kao softverski alat za pristup problemu izabran je programski jezik Python. Python je moćan, višenamjenski programski jezik stvoren od strane nizozemskog programera Guida van Rossuma koji prvu verziju Pythona (0.9.0.) izbacuje 1991. godine. Python omogućuje brz razvoj i implementaciju te je postao jedan od vodećih programskih jezika kada se govori o razvoju softvera, analizi podataka i umjetnoj inteligenciji. U strojnom učenju često je korišten zbog jednostavne i intuitivne sintakse, široke palete knjižnica s razvijenim modulima, brojnih resursa i stalne nadogradnje alata te naposlijetku kompatibilnosti s različitim platformama i operacijskim sustavima.

Neke od osnovnih karakteristika programskog jezika Python su:

- **Interpretiran jezik**: programski kod izvršava se liniju po liniju.
- **Dinamički tipovi podataka:** tipovi varijabli ne moraju se definirati prilikom inicijalizacije već se automatski prepoznaju i dodijeljuju u vremenu izvršavanja koda.
- **Višestruka paradigma:** Python podržava proceduralno, objektno orijentirano i funkcionalno programiranje.

Mogli bismo reći da programski jezik python ima više nego široku primjenu. Ovo su samo neke od grana u svijetu IT-a gdje je Python pronašao svoje mjesto:

- **Podatkovna znanost i strojno učenje**;

- **Razvoj web aplikacija**;
- **Automatizacija i skriptiranje**;
- **Razvoj igara**;
- **Ugradbeni sustavi i *Internet of Things* (IoT);**
- **Internetska sigurnost**.

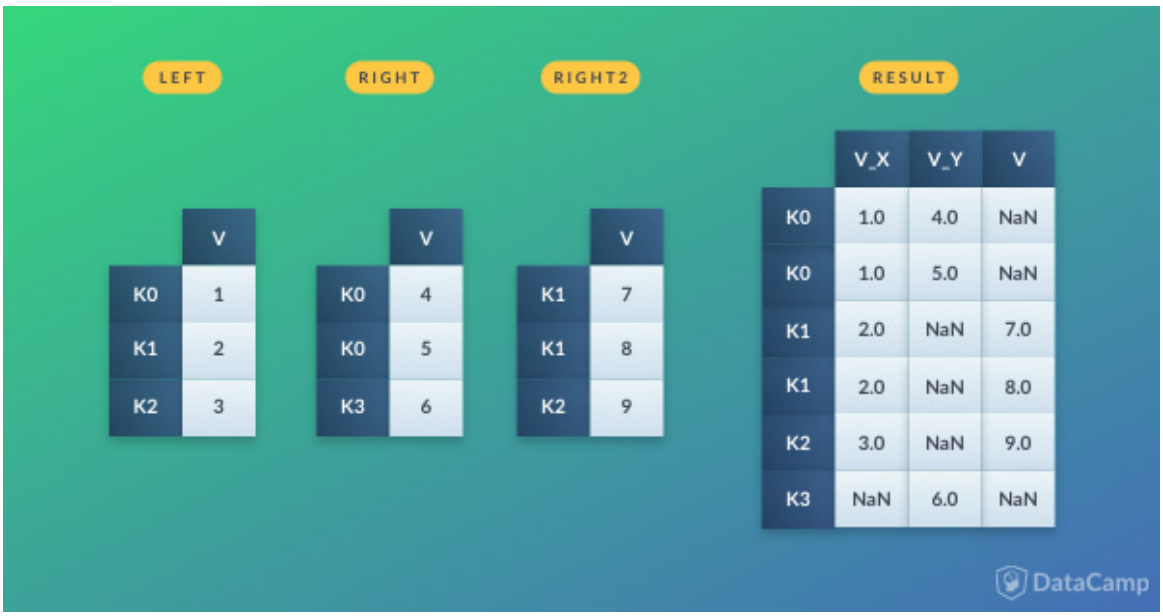
Unatoč snazi i kvalitetama, programski jezik Python ima i nekoliko ograničenja i mana. Izvršavanje koda sporije je nego kod prevodilačkih jezika kao što su C ili C++ zbog svoje interpretativne prirode. Python koristi poprilično mnogo računalne memorije, čineći ga manje idealnim za razvoj aplikacija koje mnogu opteretiti memorijsku pohranu računala. Nadalje, iako se koristi za razvoj mobilnih aplikacija, Python možda još uvijek nije najbolje rješenje za mobilnu industriju već se za tu granu većinski koristi Java.

Pythonov sustav svakodnevno je unaprijeđivan i nadograđivan od strane zajednice. Poznat je po svojoj snažnoj podršci za edukacijske svrhe te je zbog svoje dostupnosti postao sinonim za programiranje diljem svijeta. Konstantna ažuriranja pomažu da Python ostane moderan i u skladu s vremenom kako bi mogao rješavati i najkompleksnije probleme.

3.1.1 Pandas knjižnica

Pandas knjižnica služi za analizu i manipulaciju podataka. 2008. godine razvio ju je američki programer Wes McKinney u svrhu lakšeg i fleksibilnijeg rukovanja visokodimenzioniranim podacima. S Pandas knjižnicom lako se mogu učitavati razni formati podataka (CSV (engl. *Comma-separated values*) datoteke, Excel tablice, SQL baze podataka, JSON datoteke...). [2] Nudi brze i efikasne strukture podataka kao što su serije podataka i *DataFrame*, dvodimenzionalni oblik podataka nalik tablici koji omogućava kompleksne operacije nad recima i stupcima. Ulazni podaci za strojno učenje korišteni i obrađeni u ovom radu oblikovani su upravo u obliku *DataFramea*. Slika 3.1 prikazuje primjer DataFrame podataka.

Jedna od najvažnijih karakteristika Pandas-a je mogućnost rukovanja s nedostajućim podacima putem metoda poput fillna() i dropna(). Također podržava filtriranje, grupiranje, spajanje i pivotiranje podataka pomoću intuitivnih funkcija poput gro-



Slika 3.1 Primjer DataFrame podataka
[3]

upby(), merge(), i pivot_table(). Uz pomoć indeksa, omogućuje lako adresiranje redova i stupaca, dok napredne funkcije poput vektorizacije omogućuju brzu obradu velikih setova podataka. Pandas integrira s drugim knjižnicama, poput NumPy-ja za numeričke operacije i Matplotlib-a za vizualizaciju podataka.

Knjižnica je vrlo fleksibilna i omogućuje razne manipulacije, poput sortiranja podataka (sort_values()), preoblikovanja (melt(), stack()), i kreiranja novih stupaca pomoću lambda funkcija. Pandas je optimiziran za učinkovitije performanse, omogućujući rad s velikim skupovima podataka koristeći relativno malo memorije. Ukratko, Pandas je nezamjenjiv alat za analizu podataka te znanstvenike i inženjere koji žele brzo i učinkovito upravljati podacima te iz njih izvući ključne zaključke.

3.1.2 Scikit-learn knjižnica

Scikit-learn knjižnica je otvorenog koda namijenjena strojnom učenju. Inicijalno je stvorena 2007. godine kao dio *Google Summer of Code* projekta od strane Davida Cournapenaua. Od tada je postala najkorištenija knjižnica funkcija na svijetu kada

govorimo o strojnom učenju, zahvaljujući intuitivnom sučelju i izvrsnoj dokumentaciji. Knjižnica donosi širok spektar alata i modula za manipulaciju nad podacima. U ovom radu korišteni su neki od modula sckit-learn knjižnice, kao što su:

- *ensemble*: modul korišten za implementaciju funkcije RandomForestClassifier() kojom se inicijalizira algoritam nasumične šume.
- *decomposition*: ovaj modul potreban nam je za uvođenje PCA, jedne od metoda redukcije dimenzionalnosti podataka korištene u ovom radu.
- *metrics*: modul kojim uvodimo *accuracy_score*() funkciju pomoću koje računamo točnost predikcije modela nasumične šume.
- *model_selection*: modul korišten za implementaciju funkcije train_test_split kojom dijelimo ulazni skup podataka na podatke za trening i testne podatke.
- *preprocessing*: uvođenje funkcije StandardScaler() potrebne za standardizaciju podataka prije primjene metode za redukciju dimenzionalnosti.

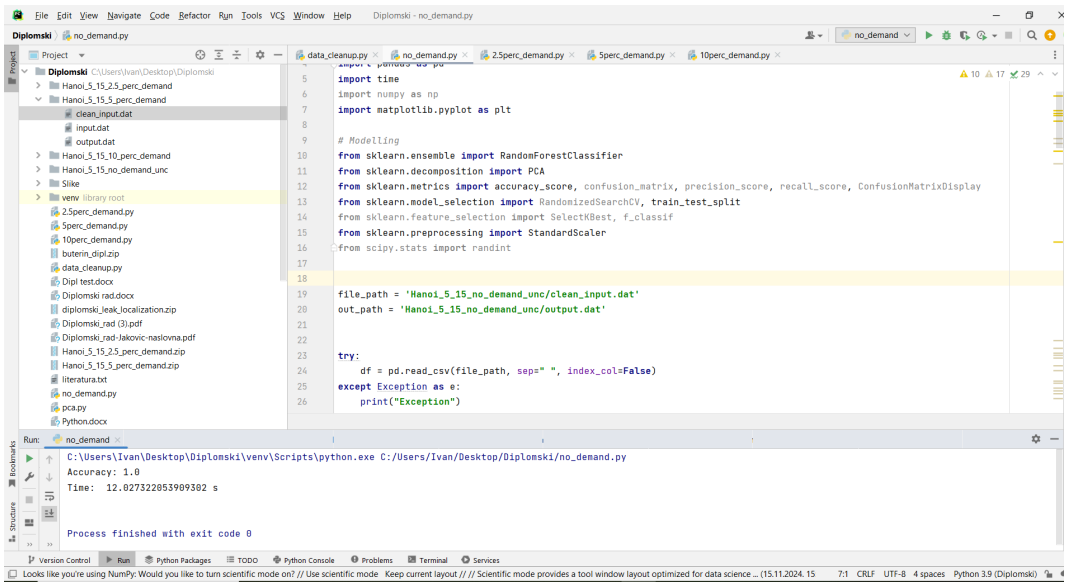
Knjižnica je optimizirana za rad s malim i srednje velikim skupovima podataka, a njezini algoritmi podržavaju različite načine učenja, uključujući nadzirano, nenadzirano i polu-nadzirano učenje. Scikit-learn je široko prihvaćena knjižnica u akademskim i industrijskim krugovima zbog svoje robusnosti, lakoće korištenja i bogate dokumentacije, što ju čini nezaobilaznim alatom za izgradnju i implementaciju modela nasumične šume i drugih algoritama strojnog učenja.

3.2 PyCharm razvojno okruženje

Kompletan programski kod pisan za rješavanje ovog problema nastao je u PyCharm razvojnom okruženju. PyCharm je moćno i bogato razvojno okruženje specijalno dizajnirano za programski jezik Python, no podržava razne programske jezike i okvire primjerice za razvoj internetskih aplikacija. (HTML, CSS, JavaScript, Django...). Razvila ga je tvrtka JetBrains, a pruža intuitivno korisničko sučelje i mnoštvo inteligentnih značajki, kao što su pametno dovršavanje koda i praćenje pogrešaka "u letu". Također, PyCharm nudi robusne alate za uklanjanje pogrešaka u programskom kodu, ugrađenu Python konzolu i alate za brzu navigaciju kroz razne projekte i programske kodove.

PyCharm podržava sustave za kontroliranje revizija kao što su Git i Mercurial, pružajući alate za grananje, spajanje te spremanje i vizualizaciju promjena direktno iz razvojnog okruženja, što ga čini pogodnim i za rad u velikim kompanijama i timovima gdje jako puno programera radi na jednom projektu. Nudi i neke dodatne napredne značajke poput Docker i SSH ekstenzije, stoga je prikladan i za profesionalne projekte visoke razine.

PyCharm je trenutno dostupan u dva izdanja: besplatno izdanje otvorenog koda, tzv. *Community edition* i plaćeno izdanje koje uključuje dodatne značajke kao što su povezivanje s bazama podataka, znanstveni alati i podrška za napredni web razvoj. Ovo razvojno okruženje poznato je po izvrsnoj dokumentaciji i dostupnim materijalima što ga čini savršenim izborom i za početnike i za iskusne razvojne programere koji žele razumljivo i učinkovito Python razvojno okruženje kako bi takav bio i programski kod. Slika 3.2 prikazuje PyCharm razvojno okruženje sa svim navedenim osnovnim značajkama.



Slika 3.2 PyCharm razvojno okruženje

Poglavlje 4

Strojno učenje

Strojno učenje naziv je za granu umjetne inteligencije sastavljenu od skupa računalnih algoritama koji koriste tehnike za prepoznavanje određenih uzoraka kako bi što efikasnije riješili problem. Aktualna je tema u istraživanju i industriji i jedno od danas najaktivnijih područja računarske znanosti, ponajviše zbog brojnih mogućnosti primjene koje se protežu od raspoznavanja uzoraka i dubinske analize podataka do robotike, računalnog vida, bioinformatike i računalne lingvistike. Nove se metode kontinuirano pojavljuju i razvijaju.

Modeli strojnog učenja "uče" informacije i odnose među njima izravno iz podataka ne oslanjajući se na teorijske jednadžbe i matematičke modele. Zadatak algoritama i modela strojnog učenja je pronaći prirodne uzorke i poveznice u podatcima te na temelju toga steći uvid i zatim odlučiti i predviđati. Prema osnovnoj podjeli, strojno učenje možemo podijeliti na nadzirano i nenadzirano učenje. Kod nadziranog učenja modeli se treniraju, odnosno grade primjenom skupa prikupljenih ulaznih podataka kojima su pridruženi odgovarajući izlazni podatci. Model treniran na tako pripremljenom setu podataka može predviđati buduće izlaze za nove ulazne podatke. Modeli nadziranog učenja mogu biti regresijski modeli (predikcija izlazne vrijednosti za ulazne podatke) ili klasifikacijski modeli (predikcija klase za ulazne podatke). Primjer takvog modela je model nasumične šume (engl.*Random forest*) korišten u ovom radu.

Kod nenadziranog učenja modeli pronalaze skrivene uzorke i inherentne strukture u ulaznim podatcima bez poznavanja izlaza. [4] Neki primjeri modela koji rade na principu nenadziranog učenja su Gaussovi mješoviti modeli i K-means clustering

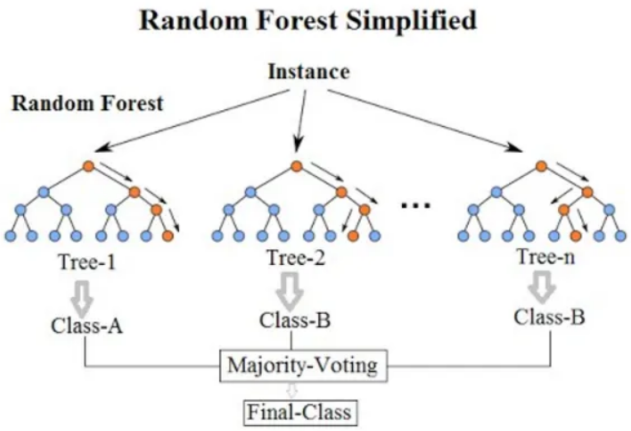
model. Navedene tehnike koriste se kod primjerice grupiranja podataka i smanjenja dimenzionalnosti podataka.

4.1 Nasumična šuma

Algoritam strojnog učenja korišten u ovom radu je algoritam nasumične šume. Prije inicijalizacije algoritma učitani skup podataka dijeli se na podatke za treniranje i testne podatke. Nasumična šuma kombinira stabla odluke kako bi izvršila određenu predikciju. Algoritam radi u nekoliko koraka. Prvi korak je kreiranje određenog broja stabala odluke (n_estimators) koji se definira u argumentu funkcije RandomForestClassifier (npr. n_estimators = 60). Svako stablo sastavljeno je od nasumično odabranog manjeg podskupa originalnog skupa podataka te razmatra nasumični podskup značajki u svakoj podjeli čime dobiva jedinstvenu strukturu i donosi neovisne odluke. Nasumičnim odabirom podskupova izbjegava se tzv. *overfitting* (nemogućnost modela da radi precizne predikcije na bilo kojem skupu podataka osim na trening skupu).

U sljedećem koraku algoritma svako pojedino stablo na bazi svog podskupa značajki radi podjelu značajki i odabire najbolju značajku. Postupak se ponavlja sve dok se ne dostigne jedan od uvjeta za prekid algoritma, kao što je primjerice maksimalna dubina (max_depth) stabla koja može biti unaprijed definirana u funkciji RandomForestClassifier zajedno s brojem stabala.

Posljednji korak je donošenje konačne predikcije koja se dobije "većinskim glasanjem" svih stabala odluke. Drugim riječima, klasa za koju je glasalo najviše stabala odluke pobjeđuje i uzima se kao konačna. Slika 4.1 prikazuje način rada algoritma nasumične šume.



Slika 4.1 Način rada algoritma nasumične šume
[5]

4.2 Primjena algoritma nasumične šume u Pythonu

Programski kod u Pythonu sastoji se od 5 datoteka: data_cleanup.py, no_demand.py, 2.5perc_demand.py, 5perc_demand.py i 10perc_demand.py. Posljednje su tri datoteke jednakih dimenzija, no razlika je u tome što su očitanja tlakova dobivena na temelju različitih postotaka varijacije bazne potrošnje vode. Datoteka data_cleanup.py iskorištena je za uklanjanje ranije spomenutih redundantnih informacija iz svih input datoteka kako bi se od višestrukih jednakih vrijednosti zadržala jedna koja predstavlja očitanje senzora u punom satu. Nakon čišćenja podataka input datoteke sadrže jednak broj redaka koji su sadržavale i prije, ali je broj stupaca smanjen na 50 (po 24 sata za senzore u čvorovima 14 i 30 te srednja vrijednost očitanja oba čvora), što je broj značajki s kojima algoritam radi.

Algoritam nasumične šume primijenjen je na svakom ulaznom skupu zasebno, u datoteci no_demand.py na manjem, a u preostalim datotekama na većim skupovima podataka. Nakon učitavanja knjižnica te njihovih modula potrebnih za manipulaciju nad podatcima i vizualizaciju rješenja, učitani su poznati input i output podatci koji su podijeljeni na podatke za trening i testne podatke. U ovom radu podjela je realizirana u omjeru 70%-30%. Zbog velikog broja dostupnih podataka koji se generiraju sintentički, 70% podataka predstavlja dovoljno velik broj ulaznih podataka

za treniranje, a 30% podataka za testiranje modela odabrano je za dodatnu validaciju rezultata predikcije modela strojnog učenja. Odabrano je da broj generiranih stabala odluke u treniranju modela na svakom ulaznom setu podataka varira u rasponu od 20 do 60. Isječak koda prikazuje implementaciju algoritma nasumične šume.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3)


rf = RandomForestClassifier(n_estimators=60)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Isječak koda 4.1 Implementacija algoritma nasumične šume

Kao što je vidljivo iz isječka programskog koda, funkcija rf.fit trenira model tijekom kojeg gradi 60 stabala odluke koja zajedno rade na klasifikaciji podataka. Nakon treniranja modela, funkcija rf.predict koristi istrenirani model na preostalom dijelu skupa odvojenom za testiranje. Model koristi većinsko glasanje svih 60 stabala kako bi se odredila konačna klasa kojoj pripada svaka instanca iz testnog skupa. Niz predviđenih vrijednosti za skup X_test pohranjuje se u varijablu y_pred.

Varijablom accuracy prikazuje se točnost modela, odnosno udio točnih predikcija u odnosu na ukupni broj predikcija. Funkcija accuracy_score uspoređuje niz predviđenih vrijednosti pohranjenih u varijablu y_pred sa stvarnim nizom vrijednostima pohranjenim u varijablu y_test. Točnost se izražava u obliku decimalnog broja u rasponu od 0 do 1 (postotak). Naravno, što je rezultat veći, tj. bliži 1, broj točnih predikcija više se podudara s ukupnim brojem svih predikcija.

4.3 Rezultati algoritma nasumične šume

Točnost modela i vrijeme izvršavanja programa može se mijenjati promjenom argumenata funkcije RandomForestClassifier, odnosno definiranjem maksimalne dubine

stabla i broja generiranih stabala odluke. Na svim ulaznim skupovima podataka mijenjali su se navedeni argumenti, a dobiveni rezultati prikazani su u tablicama 4.1-4.4. Točnosti i vrijeme potrebno za izvršavanje predstavljaju prosječne vrijednosti na temelju pet pokretanja algoritma nasumične šume. Navedeno će biti provedeno za sve ostale analize.

Tablica 4.1 Rezultati predikcijskog modela nasumične šume za scenarije bez varijacije u potrošnjama potrošača. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%)	t (s)
-	20	100	10,03
-	30	100	15,59
-	60	100	33,12
12	20	96,09	9,85
12	30	95,96	14,11
12	60	95,93	29,25

Analizom podataka iz tablica može se vidjeti da su argumenti funkcije RandomForestClassifier mijenjani na način da se najprije mijenjao broj generiranih stabala odluke uz neograničenu maksimalnu dubinu stabla, a zatim se postupak ponovio uz unaprijed definiranu maksimalnu dubinu stabla (12 za manji, a 15 za veće skupove podataka). Rezultati su dobiveni kao srednja vrijednost na temelju 5 pokretanja algoritma.

U tablici 4.1 vidimo da je točnost u svim slučajevima viša od 95%, što je i bilo za očekivati s obzirom da se radi o idealnom slučaju bez varijacije potrošnje vode, gdje se s ovako velikim brojem ulaznih podataka praktično pokriju sve moguće kombinacije koje ulaze u algoritam. Prilikom implementacije algoritma, ustanovljeno je da već s 56000 ulaznih vrijednosti algoritam postiže točnost od 100%, što je korisno saznanje s obzirom da sa četvrtinom obujma skupa postižemo istu točnost predikcije, a vrijeme izvršavanja se dodatno smanjuje.

Tablica 4.2 Rezultati predikcijskog modela nasumične šume za scenarije s ± 2.5% varijacije u potrošnjama potrošača. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%)	t (min)
-	20	67,60	1,69
-	30	68,40	2,77
-	60	69,57	5,32
15	20	69,47	1,16
15	30	61,41	1,44
15	60	61,85	3,25

Nadalje, tablica 4.1 govori nam da s neograničenom maksimalnom dubinom stabla, neovisno o povećanju broja generiranih stabala točnost algoritma iznosi maksimalnih 100%. Ipak, s definiranjem vrijednosti maksimalne dubine stabala točnost ipak počinje padati, što je i očekivano s obzirom da se na taj način prije postigne uvjet za prekid izvršavanja algoritma jer ne zalazi u toliko složene obrasce kao kada nema ograničenja dubine. Samim tim, za jednaku količinu generiranih stabala odluke, vrijeme izvršavanja algoritma kraće je uz definiranu vrijednost maksimalne dubine. Točnost u ovom slučaju minimalno opada i s povećanjem broja generiranih stabala, što nije uobičajeno, ali je moguće iz razloga što priroda značajki povećanjem broja stabala u šumi može povećati i učestalost pojave šuma u podacima.

Uvidom u tablicu 4.2 odmah možemo zaključiti da je situacija bitno drukčija. Shodno očekivanjima, vrijeme izvršavanja programa sada je značajno dulje zbog toga što ulazne datoteke s varijacijama u potrošnji sadrže 315 tisuća redaka više nego datoteka gdje varijacija u potrošnji nema. Primijećujemo da se povećanjem broja generiranih stabala odluke proporcionalno povećava vrijeme izvršavanja. Primjerice, sa 60 generiranih stabala odluke vrijeme izvršavanja otprilike 3 puta dulje nego u slučaju s 20 stabala. Nadalje, povećanjem broja stabala odluke točnost se povećava između 0.4% i 1%.

Tablica 4.3 Rezultati predikcijskog modela nasumične šume za scenarije s ± 5% varijacije u potrošnjama potrošača. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%)	t (min)
-	20	50,68	1,75
-	30	51,75	3,02
-	60	52,75	6,39
15	20	47,24	1,19
15	30	47,56	1,75
15	60	48,22	3,60

Postavljanjem ograničenja maksimalne dubine stabla točnost se smanjila između 7% i 8% u odnosu na kombinaciju parametara s jednakom količinom generiranih stabala odluke, no bez ograničenja u maksimalnoj dubini osim u kombinaciji s 20 generiranih stabala odluke, gdje se s ograničenjem u maksimalnoj dubini točnost povećala za nešto manje od 2%. Za sve kombinacije parametara vrijedi da se s maksimalnom dubinom stabla postiže brže izvršavanje.

Točnost predikcije algoritma značajno se smanjila iz nekoliko razloga. Prvi razlog je taj što su uz očitanja senzora, podaci u ovim skupovima dobiveni i na temelju varijacija potrošnje vode u čvorovima vodoopskrbnog sustava, što je otvorilo mogućnost za daleko veći broj kombinacija.

Dodatni razlog smanjenja točnosti može biti taj što veći skup podataka može povećati količinu šuma među značajkama. Također, postavljanje maksimalne dubine stabla na prenisku vrijednost može rezultirati smanjenom točnošću jer zbog ograničenja u dubini, model možda neće dobiti pristup nekim granama koje bi bile korisne u klasifikaciji.

U tablici 4.3 uočavamo da i dalje vrijeme izvršavanja raste s povećanjem broja generiranih stabala te da se smanjuje s uvođenjem granične maksimalne dubine stabla.

Tablica 4.4 Rezultati predikcijskog modela nasumične šume za scenarije s ± 10% varijacije u potrošnjama potrošača. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%)	t (min)
-	20	35,54	2,16
-	30	36,40	3,32
-	60	37,62	7,25
15	20	34,05	1,31
15	30	34,64	2
15	60	35,07	3,93

Upravo navedeno vrijedi i za točnost predikcije koja porastom broja stabala kod neograničene maksimalne dubine raste za otprilike 1%, dok je za jednaku kombinaciju generiranih stabala, ali uz uvjet maksimalne dubine niža za 3%-4%.

U tablici 4.4 možemo primijetiti da povećanje broja generiranih stabala povećava točnost predikcije i kada je ograničenje u maksimalnoj dubini definirano i kada ga nema. Nadalje, povećanjem broja generiranih stabala očekivano se povećava i vrijeme izvršavanja. Granica maksimalne dubine smanjuje točnost za 1%-2.5% u odnosu na kombinacije s jednakom količinom broja stabala koje nemaju ograničenje u dubini. Također, postavljanje granice maksimalne dubine gotovo dvostruko skraćuje vrijeme izvršavanja.

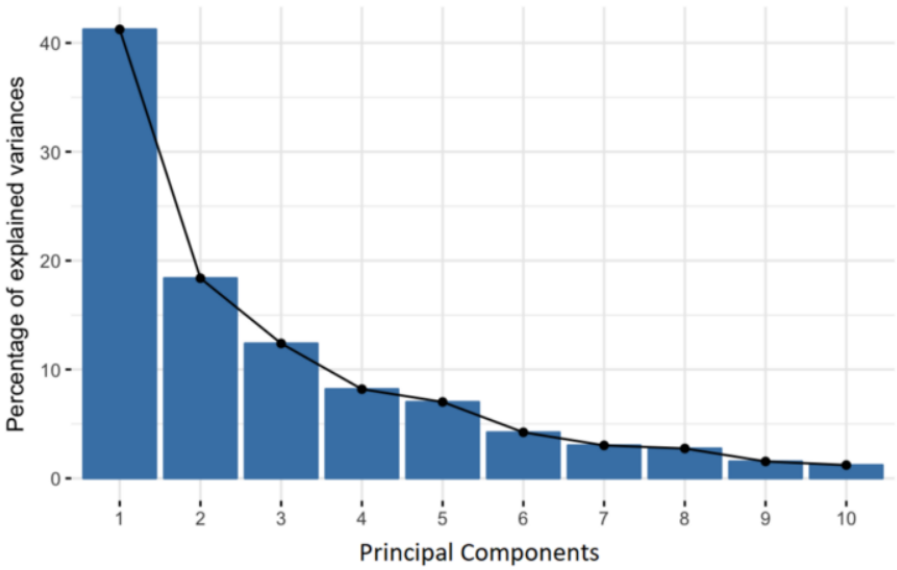
Uspoređujući rezultate prikazane u tablicama 4.2-4.4, vidimo da se točnost predikcije modela smanjuje od 15%-17% s povećanjem raspona u baznoj potrošnji. Dolazimo do zaključka da su podaci heterogeniji što je veći raspon u baznoj potrošnji vode, odnosno da se povećanjem raspona varijacija u baznoj potrošnji smanjuje točnost predikcije algoritma nasumične šume. U vremenima izvršavanja algoritma nema značajnih razlika, što ne treba iznenaditi s obzirom da su sve ulazne datoteke jednakih dimenzija.

Poglavlje 5

PCA (*Principal component analysis*)

PCA (engl. *principal component analysis*) popularna je i često korištena metoda redukcije dimenzionalnosti podataka. Koristi se u raznim područjima, no ponajviše u strojnom učenju, računalnoj znanosti i analizi podataka. Osnovni princip PCA je smanjiti predimenzionirani set podataka u manji set, ali s ciljem da se zadrži što je više moguće informacija iz originalnog seta podataka. Manji setovi podataka jednostavniji su za istraživanje i vizualizaciju te čine analizu bržom i jednostavnijom za algoritme strojnog učenja.

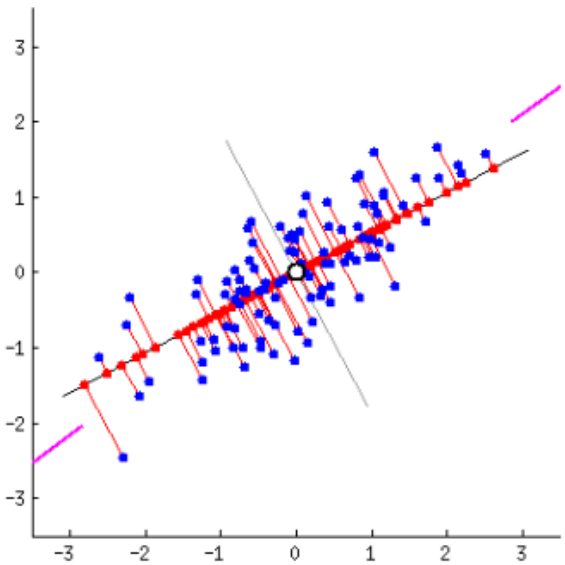
Kao što se može zaključiti iz samog naziva metode, PCA radi na temelju analize glavnih komponenti (engl. *principal components*). Glavne komponente zapravo su nove varijable konstruirane kao linearna kombinacija ili mješavina inicijalnih (originalnih) varijabli. Kombinacije su složene na takav način da su glavne komponente nekorelirane te da je većina informacija iz inicijalnog seta podataka sažeta u prvim komponentama. Npr. ako se originalni set sastoji od 5-dimenzionalnih podataka dobivamo 5 glavnih komponenti, no PCA će nastojati maksimalnu količinu informacija pridružiti prvoj komponenti, zatim će maksimalnu količinu preostalog dijela informacija pridružiti drugoj komponenti itd. Način raspodjele informacija na glavne komponente prikazan je na slici 5.1.



Slika 5.1 Raspodjela informacija na glavne komponente
[6]

5.1 Konstruiranje glavnih komponenti

Kako bi PCA metoda bila lakše shvatljiva, važno je razmotriti matematičku pozadinu i razumijeti korake u procesu redukcije podataka. S obzirom da postoji onoliko glavnih komponenti koliko je varijabli u setu podataka, konstruiranje glavnih komponenti odvija se na način da prva glavna komponenta objašnjava najveću moguću varijancu u setu. Ona, kao i sve ostale komponente zapravo su pravci koji predstavljaju nove osi koordinatnog sustava u kojem su podaci raspoređeni. Na slici 5.2 prva glavna komponenta bio bi pravac koji se podudara s ljubičastim oznakama jer maksimizira varijancu (prosjek kvadratne udaljenosti točaka od ishodišta). Druga glavna komponenta izračunava se na isti način, uz uvjet da je nekorelirana s prvom i da objašnjava sljedeću najveću varijancu. Postupak se ponavlja sve dok se ne izračuna broj glavnih komponenti jednak broju originalnih varijabli.



Slika 5.2 Prva glavna komponenta

5.2 Kompletan tijek PCA metode

Mogli bismo reći da se princip rada PCA može podijeliti u nekoliko osnovnih koraka:

- **Standardizacija:** cilj ovog koraka je standardizirati raspon inicijalnih varijabli kako bi svaka od njih jednako doprinosila analizi. Ukoliko postoje značajne razlike u rasponu inicijalnih varijabli, one varijable s većim rasponom dominirati će nad varijablama s manjim rasponom (npr. varijabla koja poprima vrijednosti između 0 i 100 dominirat će nad varijablom koja poprima vrijednosti od 0 do 1), što može dovesti do pristranih rezultata. Matematički, standardizacija se može postići oduzimanjem srednje vrijednosti te dijeljenjem sa standardnom devijacijom svake vrijednosti svake varijable (formula 5.1).

$$z = \frac{value - mean}{standard\_deviation} \tag{5.1}$$

- **Računanje matrice kovarijance:** varijable su u nekim slučajevima visoko korelirane, stoga se matrica kovarijance računa s ciljem oktrivanja takvih korelacija. Provjerava se kako varijable ulaznog seta odstupaju od srednje vrijednosti te postoji li između varijabli ulaznog seta podataka nekakva veza. Matrica kovarijance je kvadratna matrica s dimenzijama p x p, gdje je p broj dimenzija ulaznog seta podataka. Slika 5.3 prikazuje matricu kovarijance za 3-dimenzionalni set podataka.

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Slika 5.3 Matrica kovarijance za 3-dimenzionalni set podataka

Uočavamo da je matrica simetrična s obzirom na glavnu dijagonalu. Budući da za kovarijancu vrijedi svojstvo komutativnosti (Cov(a,b) = Cov(b,a)), gornje trokutasti i donje trokutasti dio matrice su jednaki. Ukoliko je kovarijanca pozitivnog predznaka, varijable koreliraju, odnosno porastom jedne varijable raste i druga. Negativan predznak kovarijance pokazuje da pad jedne varijable rezultira porastom druge.

- **Računanje svojstvenih vektora (*eigenvectors*):** svojstveni vektori i svojstvene vrijednosti računaju se iz matrice kovarijance kako bi odredili glavne komponente podataka. Vektori i vrijednosti uvijek stoje u paru, odnosno svaki svojstveni vektor ima svoju svojstvenu vrijednost. Također, njihov broj jednak je broju dimenzija podataka. Svojstveni vektori predstavljaju smjer osi koordinatnog sustava gdje je sadržano najviše varijance (informacija) među podacima. Svojstvene vrijednosti su koeficijenti pridruženi svojstvenim vektorima koji nam daju količinu varijance koju nosi pojedina glavna komponenta. [7]

  Kako bi objasnili pridruživanje svojstvenih vektora glavnim komponentama, pretpostavimo da postoji dvodimenzionalni set podataka ***x,y*** s pripadajućim

svojstvenim vektorima (v) i svojstvenim vrijednostima (λ) kao na slici 5.4. Poredamo li svojstvene vrijednosti padajućim redoslijedom, vidimo da je $\lambda 1 > \lambda 2$, što znači je da svojstveni vektor koji pripada prvoj glavnoj komponenti vektor *v1*, dok drugoj glavnoj komponenti pripada vektor *v2*.

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

Slika 5.4 Pridruživanje vektora glavnim komponentama

Nakon dobivenih glavnih komponenti, računa se postotak varijance koji nosi pojedina komponenta, a postotak dobijemo na način da svojstvenu vrijednost svake komponente podijelimo sa zbrojem svih svojstvenih vrijednosti. Iz gornje slike zaključujemo da u ovom slučaju prva glavna komponenta (PC1) nosi 96%, a druga glavna komponenta (PC2) 4% varijance podataka.

- **Kreiranje vektora značajki:** u ovom koraku odlučuje se hoće li se zadržati sve glavne komponente ili će se one s najmanjim svojstvenim vrijednostima odbaciti, dok će se od preostalih glavnih komponenti formirati matrica vektora zvana vektor značajki. Dakle, vektor značajki je matrica čiji su stupci svojstveni vektori glavnih komponenti koje smo odlučili zadržati. Primjera radi, ako smo od 5 glavnih komponenti zadržali 2 komponente, konačni set podataka imati će 2 dimenzije. Vektor značajki je prvi korak prema redukciji dimenzionalnosti podataka. Na primjeru slike 5.4, odbacimo li drugu glavnu komponentu (PC2) smanjujemo dimenziju podataka na 1, a ipak smo odbacili tek neznatan dio informacija jer je ustanovljeno da druga glavna komponenta nosi samo 4% informacija iz originalnog skupa podataka.

5.3 Implementacija PCA u Pythonu

Nakon što su pokazane glavne karakteristike i objašnjeni koraci PCA metode redukcije dimenzionalnosti podataka, metoda je primijenjena u programskom jeziku Python. Model nasumične šume koji je u prethodnom poglavlju treniran na originalnim ulaznim podacima, ovdje će nakon primjene metode redukcije biti treniran na manjem ulaznom setu podataka te će se prikazati i komentirati dobiveni rezultati.

Implementacija PCA metode u Pythonu prilično je intuitivna i jednostavna te se sastoji od svega nekoliko dodatnih linija programskog koda. Za dodavanje metode u programski kod potrebno je uključiti knjižnicu funkcija scikit-learn s pripadajućim modulom decomposition. Implementacija metode prikazana je u isječku programskog koda.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

pca = PCA(n_components=0.999)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

start_time = time.time()
rf = RandomForestClassifier(max_depth=None, n_estimators=60)
rf.fit(X_train_pca, y_train)

y_pred = rf.predict(X_test_pca)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
total_time = time.time() - start_time
print("Time: ", total_time, "s")
```

Isječak koda 5.1 Implementacija PCA metode

Ovaj programski kod u odnosu na kod iz prethodnog poglavlja gdje se model trenirao na originalnim podacima sadrži nekoliko dodataka čije objašnjenje slijedi u nastavku:

- scaler = StandardScaler(): izvršava se prvi korak u implementaciji PCA metode, a to je standardizacija podataka.
- X_train_scaled = scaler.fit_transform(X_train): standardizacija se primjenjuje na trening skupu i "uči" srednje vrijednosti i standardne devijacije trening skupa.
- X_test_scaled = scaler.transform(X_test): testni podaci se standardiziraju isto na isti način kao trening podaci.
- pca = PCA(n_components=0.999): kreiranje instance PCA modela. Argument metode možemo zadati na 2 načina. Prvi način je definiranje točnog broja glavnih kompomenti koje zadržavamo (npr. n_components = 10 zadržava točno 10 glavnih komponenti), a drugi način je zadavanje postotka ukupne varijabilnosti u podacima. Drugim riječima, postavljanjem varijable n_components = 0.999 PCA odabire najmanji broj glavnih komponenti koje zajedno nose 99,9% varijance ulaznih podataka. Odabir načina ovisi o prirodi podataka koji se obrađuju, no u ovom radu će se koristiti drugi način, odnočno postotak varijance koji želimo zadržati.
- X_train_pca = pca.fit_transform(X_train_scaled): PCA se primjenjuje na trening skupu i na temelju varijabilnosti računaju se glavne komponente. PCA nakon izračuna glavnih komponenti transformira trening skup u novi prostor manjih dimenzija.
- X_test_pca = pca.transform(X_test_scaled): model koji je naučen na trening podacima PCA primjenjuje na testne podatke kako bi se dobio novi testni skup jednakih reduciranih dimenzija kao trening skup.

5.4 Rezultati PCA metode

Nakon implementacije PCA metode redukcije dimenzionalnosti u programskom kodu, slijedi prikaz rezultata i utjecaja metode na točnost modela strojnog učenja i vrijeme izvršavanja programskog koda. Testiranje metode obavljeno je na svim ulaznim skupovima podataka u jednakim uvjetima, odnosno na jednakim kombinacijama argumenata funkcije RandomForestClassifier kako bi se postigla vjerodostojnost rezultata i realno prikazao utjecaj metode redukcije dimenzionalnosti. Dodatno, željeni postotak opisane varijance podataka fiksiran je na 99,9% (n_components = 0.999), što znači da je cilj reducirati set podataka, no tako da još uvijek zadržava 99,9% varijance. Dobiveni rezultati za svaku datoteku prikazani su u tablicama 5.1-5.4.

Tablica 5.1 Rezultati predikcijskog modela nasumične šume za scenarije bez varijacije u potrošnjama potrošača nakon provedene PCA analize. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%) (n_components = 0.999)	t (s)
-	20	100	5,33
-	30	100	8,81
-	60	100	16,21
12	20	85,51	3,93
12	30	84,72	5,30
12	60	86,79	11,09

U tablici 5.1 uočavamo da za neograničenu dubinu stabla točnost predikcije i dalje iznosi 100% neovisno o broju generiranih stabala odluke. Kada se postavi ograničenje u maksimalnoj dubini stabla, točnost ipak opada za otprilike 10% odnosu na rezultate dobivene na originalnim ulaznim podacima u tablici 4.1.

Tablica 5.2 Rezultati predikcijskog modela nasumične šume za scenarije s ± 2.5% varijacije u potrošnjama potrošača nakon provedene PCA analize. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%) (n_components = 0.999)	t (min)
-	20	53,67	1,15
-	30	54,18	1,41
-	60	54,58	3,74
15	20	51,75	0,77
15	30	52,26	1,27
15	60	52,34	2,57

Uočavamo da se vrijeme izvršavanja nakon primjene PCA metode skratilo za sve kombinacije parametara modela nasumične šume (broj generiranih stabala odluke i maksimalne dubine). Nameće se pitanje koliko je metoda zapravo smanjila dimenzije podataka. Naredbom print u programskom kodu lako saznajemo dimenzije reduciranog trening skupa podataka u obliku (x,y) gdje je x broj redaka, a y broj stupaca seta podataka.

Saznajemo da su trenutne dimenzije trening skupa (147000, 2), što bi značilo da se sastoji od 147000 redaka (od ukupno 210000, 70% odvojenih za trening) i samo 3 stupca od početnih 50. S obzirom da smo argumentom n_components definirali koliki postotak varijance originalnih podataka želimo zadržati, zaključujemo da 3 glavne komponente sadrže 99,9% varijance originalnog seta podataka i da su samo 3 stupca očitanja tlakova dovoljna da točnost predikcije modela i dalje bude iznimno visoka. Budući da je većina originalnog seta podataka odbačena, ne iznenađuje nas ni prosječno više nego dvostruko kraće izvršavanje treninga modela.

Iz tablica 5.2, 5.3 i 5.4 doznajemo da je vrijeme izvođenja ostalo značajno kraće nego u slučaju treniranja modela s originalnim podacima. Upisom (X_train_pca.shape)

Tablica 5.3 Rezultati predikcijskog modela nasumične šume za scenarije s ± 5% varijacije u potrošnjama potrošača nakon provedene PCA analize. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%) (n_components = 0.999)	t (min)
-	20	41,98	1,21
-	30	42,89	1,76
-	60	43,53	3,76
15	20	42,22	0,82
15	30	42,69	1,28
15	60	42,73	2,55

u programski kod bilo koje od triju datoteka doznajemo da se broj stupaca u skupu smanjio na 4, odnosno 4 glavne komponente sadrže 99,9% varijance. Točnost je u načelu niža u usporedbi s rezultatima dobivenima u tablicama 4.2 i 4.3, za skup s varijacijom potrošnje od ± 2.5% to je otprilike 14% u prosjeku za nedefiniranu maksimalnu dubinu stabla, a 9% za postavljanje ograničenja u dubini.

Kod skupa s varijacijom potrošnje od ± 5% primijećujemo pad od otprilike 9% kod neograničene dubine, a kod definiranog ograničenja oko 5%. Možemo izvesti zaključak da definiranjem određenog maksimalnog broja generiranih stabala odluke ubrzavamo vrijeme treniranja modela, a smanjujemo razliku u točnosti u odnosu na model treniran na originalnim podacima.

Tablica 5.4 Rezultati predikcijskog modela nasumične šume za scenarije s ± 10% varijacije u potrošnjama potrošača nakon provedene PCA analize. U tablici t predstavlja vrijeme izvršavanja.

Maksimalna dubina	Broj stabala	Točnost (%) (n_components = 0.999)	t (min)
-	20	37,31	1,12
-	30	38,08	1,73
-	60	39,21	3,73
15	20	37,21	0,84
15	30	37,68	1,23
15	60	38,41	2,44

Ipak, primjena PCA metode na skupu s varijacijom potrošnje od ± 10% pridonijela je laganom porastu točnosti predikcije. Točnost je na primjeru kombinacije sa 60 generiranih stabala i definiranom granicom maksimalne dubine stabla porasla za 3,34% u odnosu na točnost modela nereduciranog skupa podataka. Dalje možemo zaključiti da točnost predikcije blago opada s definiranjem ograničenja maksimalne dubine stabla, ali i blago raste s porastom broja generiranih stabala. Jednako vrijedi i za vrijeme izvršavanja.

Poglavlje 6

Odabir značajki (*Feature selection*)

Odabir značajki proces je koji prema određenom kriteriju iz skupa značajki odabire podskup kako bi optimalno reducirao količinu podataka. Najvažnije uloge odabira značajki su naravno redukcija dimenzionalnosti podataka, ubrzanje algoritma strojnog učenja, eventualno povećanje točnosti predikcije algoritma te povećanje razumljivosti rezultata.

Prema osnovnoj podjeli algoritama odabira značajki, razlikujemo 3 glavne klase metoda: metode omotača (engl. *wrapper methods*), ugrađene metode (engl. *embedded methods*) i filter metode (engl. *filter methods*). Svaka glavna klasa dalje se dijeli na konkretne metode redukcije dimenzionalnosti, a u narednim potpoglavljima slijedi kratko objašnjenje svake od njih.

6.1 Metode omotača

Metode omotača vrše evaluaciju podskupa značajki trenirajući i testirajući model na raznim kombinacijama. Ne razmatraju prediktivnu snagu jedne značajke, već podskupa istih. Neke od metoda omotača su:

- Odabir prema naprijed (engl. *forward selection*): iterativni postupak započinje s praznim skupom značajki gdje gdje se postepeno dodaju značajke koje u svakoj iteraciji najviše unaprijeđuju model. Postupak se ponavlja sve dok dodavanje nove značajke više ne pridonosi poboljšanju performansama modela.

- Eliminacija prema natrag (engl. *backward elimination*): iz punog skupa značajki iterativno se uklanja značajka s najnižom važnosti. Slično odabiru prema naprijed, u ovom slučaju postupak se ponavlja dok uklanjanje značajke više ne pridonosi poboljšanju modela.
- Iscrpna selekcija (engl. *exhaustive selection*): kreira svaki mogući podskup značajki te za svaki podskup gradi zaseban algoritam za učenje. Odabire se podskup čiji algoritam pokaže najbolje rezultate.
- Rekurzivna eliminacija (engl. *recursive feature elimination (RFE)*): odabire značajke na temelju njihove važnosti za model rekurzivno promatrajući sve manji i manji podskup značajki. U svakom koraku iteracije eliminira značajke s najmanjom važnosti te se postupak ponavlja sve dok ne ostane željeni broj značajki. [8]

6.2 Ugrađene metode

Ugrađene metode ugrađuju odabir značajki direktno u proces treniranja modela. Drugim riječima, treniranje modela i odabir značajki vrše se paralelno. [9]. Tijekom treninga, algoritmi značajkama dodijeljuju određenu težinu temeljenu na njohovom doprinosu performansama modela. Značajke s minimalnim doprinosom mogu biti ignorirane i odbačene. Popularne i često korištene ugrađene metode su:

- LASSO regresija (engl. *Least Absolute Shrinkage and Selection Operator*): metoda linearne regresije koja težine nebitnih značajki snižava prema nuli i na taj ih način efektno uklanja iz modela. [10] Prednost LASSO regresije je ta što automatski odabire podskup značajki tijekom treninga modela, no mana može biti velika korelacija među značajkama.
- Ridge regresija: metoda linearne regresije slična LASSO regresiji, smanjuje utjecaj značajki na model, no za razliku od LASSO regresije ne snižava težine značajki na nulu te ih tako ne uklanja eksplicitno iz modela.

6.3 Filter metode

Filter metode skupina su metoda odabira značajki za redukciju dimenzionalnosti koje odabiru značajke bazirane na određenim kriterijima prije građenja modela, stoga su neovisne o bilo kojem algoritmu strojnog učenja [11]. Značajke se biraju na temelju njihovih rezultata statističkih testova koji provjeravaju njihovu korelaciju s ciljnom varijablom (u ovom radu s lokacijom čvora na kojem se nalazi oštećenje) [12]. Baziraju se na jednostavnim izračunima pa su zato lako primjenjive na visokodimenzioniranim podacima. Za razliku od metoda omotača, proučavaju svaku značajku pojedinačno, ignorirajući potencijalne interakcije među značajkama. Obično funkcioniraju u nekoliko jednostavnih koraka:

- Evaluacija značajki: svaka značajka evaluirana je koristeći određenu statističku metriku.
- Rangiranje: značajke su rangirane prema njihovim rezultatima
- Odabir: bira se najboljih k značajki koje zadovoljavaju odabrani kriterij

Slijedi opis nekih od najvažnijih i najčešće korištenih filter metoda:

- Prag varijance (engl. *Variance treshold*): metoda koja izračunava varijancu svake značajke i odbacuje značajke čija je varijanca ispod određenog predefiniranog praga. Značajke s niskom varijancom smatraju se nedovoljno korisnima da pridonesu performansama modela. Prednosti ove metode su jednostavna implementacija i interpretacija te računalna efikasnost (zahtijeva vrlo malo računalnih resursa čak i za velike setove podataka). S druge strane, ova metoda ima nekoliko mana. Primjerice, visoka varijanca neke značajke ne garantira da je ta značajka relevantna za zadatak koji se obrađuje. Također, metoda evaluira značajke zasebno te na taj način potencijalno propušta značajke koje bi mogle biti relevatne samo u kombinaciji s nekom drugom značajkom.
- Metoda zajedničkih informacija (engl. *Mutual information*): mjeri statističku zavisnost između značajki i ciljne varijable pomoću entropije. Izračunava koliko informacija pojedina značajka sadrži kako bi mogla predvidjeti ciljnu varijablu. Prednost metode zajedničkih informacija je što se ponaša podjednako dobro u radu i s kontinuiranim i diskretnim vrijednostima, a kao manu bismo mogli istaknuti računalnu složenost za velike količine podataka.

6.3.1 ANOVA f-test

Filter metoda odabira značajki u ovom radu korištena za redukciju dimenzionalnosti ulaznih podataka je ANOVA f-test metoda. Akronim ANOVA nastao je iz sintagme "analiza varijance" (engl. *Analysis Of Variance*). To je statistički test koji određuje postoje li statistički značajne razlike među srednjim vrijednostima dviju ili više grupa, odnosno procjenjuje je li varijanca između grupa značajno veća od varijance unutar jedne grupe. Taj se omjer naziva f-statistika. Veća vrijednost f-statistike govori da određena značajka značajno doprinosi razlici između grupa.

Način rada ANOVA f-testa u odabiru značajki

Način rada ANOVA f-testa u odabiru značajki i redukciji dimenzija podataka možemo podijeliti u nekoliko osnovnih koraka:

- za svaku značajku, podaci se dijele u grupe s obzirom na klase ciljne varijable
- računa se varijanca između grupa i varijanca unutar jedne grupe
- izračun f-statistike: kao što je navedeno, omjer vrijednosti iz prethodnog koraka predstavlja f-statistiku kako bi se odredila važnost pojedine značajke
- rangiranje značajki na temelju vrijednosti njihove f-statistike
- zadržavanje k najboljih značajki

Ova je metoda kao i PCA implementirana u Pythonu i primijenjena na svim ulaznim setovima podataka s jednakim kombinacijama argumenata funkcije modela nasumične šume kako bi se mogao vidjeti utjecaj metode na točnost predikcije modela te kako bismo mogli izvršiti usporedbe dviju metoda redukcije dimenzionalnosti podataka.

Implementacija ANOVA f-testa u Pythonu

Kao i u slučaju PCA metode redukcije, implementacija ANOVA f-testa u programskom jeziku Python intuitivna je i jednostavna. Potrebno je iz modula *feature_selection* koji se nalazi unutar scikit-learn knjižnice uvesti metodu SelectKbest za odabir k najboljih značajki te metodu fclassif za računanje f-statistike. Isječak

programskog koda prikazuje implementaciju ANOVA f-testa.

```
import pandas as pd
import time
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection train_test_split
from sklearn.feature_selection import SelectKBest, f_classif

file_path = 'Hanoi_5_15_10_perc_demand/clean_input.dat'
out_path = 'Hanoi_5_15_10_perc_demand/output.dat'

try:
    df = pd.read_csv(file_path, sep=" ")
except Exception as e:
    print("Exception")

X = df
y = pd.read_csv(out_path).values.ravel()

X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.3)
selector = SelectKBest(score_func=f_classif, k=10)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

start_time = time.time()

rf = RandomForestClassifier(max_depth=15, n_estimators=60)
rf.fit(X_train_selected, y_train)
y_pred = rf.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
total_time = time.time() - start_time
print("Time: ", total_time, "s", "\n", total_time/60, "min")
```

Isječak koda 6.1 Implementacija ANOVA f-testa

Programski kod iznimno je sličan kodu za implementaciju PCA metode, uz nekoliko sitnih razlika:

- selector = SelectKBest(score_func=f_classif, k=10): ovom naredbom inicijalizira se varijabla selector te definira metoda čiji argumenti govore da će elevantna funkcija za važnost značajki biti f-statistika i da će se nakon izračuna varijance zadržati 10 značajki s najvišom vrijednosti f-statistike. Te će značajke predstavljati reducirani ulazni skup podataka na kojem će se trenirati Random forest model.
- X_train_selected = selector.fit_transform (X_train, y_train): metoda se primijenjuje na trening skupu.
- X_test_selected = selector.transform(X_test): testni skup poprima dimenzije trening skupa.

6.3.2 Rezultati ANOVA f-testa

Nakon učitavanja ulaznih i izlaznih datoteka, postupak kojim je provedeno treniranje modela na originalnim ulaznim skupovima i na reduciranim skupovima pomoću PCA analize, ponovljen je uz primjenu metode ANOVA f-testa kako bi se u jednakim uvjetima (broj generiranih stabala odluke i maksimalna dubina stabla u algoritmu nasumične šume), odnosno na identičnim podacima mogle usporediti različite metode redukcije dimenzionalnosti i izvući pojedini zaključci. Broj najboljih značajki (značajke koje su izračunom varijance imale najveće vrijednosti f-statistike) postavljen je na 10. U nastavku su pomoću tablica prikazani rezultati primjene ANOVA f-testa

Točnost i vrijeme izvršavanja programa ponovno su prosječne vrijednosti dobivene na temelju 5 pokretanja programskog koda. Analizom podataka iz manjeg skupa gdje nema varijacija u potrošnji vode iz tablice 6.1, uočavamo da je primjenom ANOVA f-testa jednako kao u slučaju treniranja nasumične šume na originalnim podacima i reduciranom setu podataka pomoću PCA metode redukcije algoritam pokazao točnost od 100% u slučajevima kada nema ograničenja u maksimalnoj dubini stabla. Vrijeme izvršavanja za neograničenu maksimalnu dubinu te 20 i 30 generiranih stabala odluke otprilike je 4-5 sekundi dulje, a kada se generira 60 stabala odluke

Tablica 6.1 Rezultati predikcijskog modela nasumične šume za scenarije bez varijacije u potrošnjam potrošača nakon provedene ANOVA-e.

Maksimalna dubina	Broj stabala	Točnost % (k = 10)	Vrijeme izvršavanja (s)
-	20	100	9,18
-	30	100	13,26
-	60	100	27,90
12	20	72,80	7,01
12	30	73,15	10,18
12	60	76,62	20,25

otprilike 11 sekundi dulje nego kod primjene PCA metode redukcije, no ipak nešto kraće nego u slučaju kada se program izvršavao s originalnim setom podataka.

Vidimo da postavljanjem maksimalne dubine stabla te generiranjem 20 i 30 stabala odluke točnost opada za oko 27%, dok je sa 60 stabala odluke pala za nešto više od 23% u odnosu na slučaj gdje maksimalna dubina nije definirana. U odnosu na PCA metodu točnost je pala za otprilike 11-13%, a u usporedbi s originalnim skupom podataka za oko 22%. Što se vremena izvršavanja tiče, s 20 stabala odluke program se prosječno izvršava oko 3 sekunde dulje, s 30 stabala odluke oko 5 sekundi dulje, a sa 60 stabala odluke oko 9 sekundi dulje nego kod PCA metode redukcije. U odnosu na originalan skup podataka, kod primjene ANOVA f-testa program se izvršava neznatno brže s 20 i 30 stabala odluke, dok je sa 60 stabala odluke razlika oko 9 sekundi.

Zaključujemo da je na manjem skupu podataka u objema mjerenim metrikama, točnosti predikcije modela strojnog učenja i vremenu izvršavanja programa, PCA pokazala bolje rezultate nego primjena ANOVA f-testa. Pogledajmo što se događalo s preostala tri ulazna seta podataka većih dimenzija (tablica 6.2, tablica 6.3 i tablica

6.4).

Tablica 6.2 Rezultati predikcijskog modela nasumične šume za scenarije s ± 2.5% varijacije u potrošnjama potrošača nakon provedene ANOVA-e.

Maksimalna dubina	Broj stabala	Točnost % (k = 10)	Vrijeme izvršavanja
-	20	37,69	47,88 s
-	30	38,38	1,16 min
-	60	38,68	2,68 min
15	20	34,64	24,58 s
15	30	35,08	34,23 s
15	60	35,53	1,21 min

U tablici 6.2 uočavamo da kod primjene ANOVA f-testa na podatke s varijacijom u potrošnjama potrošača od 2.5% točnost predikcije kod neograničene maksimalne dubine stabla ne ovisi toliko o broju generiranih stabala odluke, s obzirom da je sa 60 stabala odluke točnost tek oko 1% veća nego u slučaju sa 20 stabala. Nadalje, vidimo da se vrijeme izvršavanja sukladno očekivanjima povećava s porastom broja stabala odluke. U odnosu na originalni set podataka gdje se točnost kretala između 67% i 70% vidimo značajan pad točnosti predikcije za više od 30%, ali i gotovo dvostruko kraće vrijeme izvršavanja, s obzirom da je treniranje nasumične šume sa 60 stabala na originalnom setu podataka trajalo 5,32 minuta, a u ovom slučaju treniranje je završeno nakon 2,68 minuta.

Definiranjem granice maksimalne dubine stabala točnost predikcije dodatno je pala za otprilike 3% koju god količinu stabala odluke odabrali, što je razlika u odnosu na originalni set podataka gdje je u slučajevima s 30 i 60 stabala odluke točnost pala za otprilike 8%, dok je s 20 stabala odluke točnost čak i porasla za 2%. Vrijeme izvršavanja je postavljanjem maksimalne dubine stabla otprilike dvostruko kraće.

Tablica 6.3 Rezultati predikcijskog modela nasumične šume za scenarije s ± 5% varijacije u potrošnjama potrošača nakon provedene ANOVA-e.

Maksimalna dubina	Broj stabala	Točnost % (k = 10)	Vrijeme izvršavanja
-	20	25,28	49,53 s
-	30	25,42	1,35 min
-	60	26,19	3,01 min
15	20	24,55	24,17 s
15	30	25,03	35,14 s
15	60	25,38	1,24 min

U usporedbi s PCA metodom primijenjenom na ovom setu podataka, točnost predikcije s ANOVA f-testom otprilike je 17% niža za sve kombinacije broja stabala i maksimalne dubine stabla. Kod obje metode s porastom broja stabala možemo primijetiti povećanje točnosti za otprilike 0.5%, dok je kod PCA metode definiranjem granice maksimalne dubine stabla točnost snizila za otprilike 2%. Što se vremena izvršavanja tiče, ANOVA-f-test na ovom je setu povećao brzinu izvršavanja u odnosu na PCA metodu. Zaključujemo da je treniranje nasumične šume na ovom setu nakon primjene ANOVA f-testa brže nego nakon primjene PCA metode, no uz značajan pad točnosti predikcije.

U tablici 6.3 prikazani su rezultati primjene ANOVA f-testa na set podataka s varijacijama u potrošnjama potrošača od 5%. I na ovom setu podataka metoda redukcije značajno je skratila vrijeme izvršavanja u odnosu na originalni set. Sa 60 stabala odluke vrijeme se smanjilo sa 6,39 minuta na 3,01 minuta. Primijećujemo da se točnost predikcije u odnosu na originalni set podataka dvostruko smanjila. Porast broja stabala i kod originalnog seta podataka i nakon primjene ANOVA f-testa rezultirao je porastom točnosti predikcije između 0,5% i 1%. Ograničena maksimalna dubina stabla smanjila je točnost predikcije za oko 0.7%. Vrijeme izvršavanja gotovo

Tablica 6.4 Rezultati predikcijskog modela nasumične šume za scenarije s ± 10% varijacije u potrošnjama potrošača nakon provedene ANOVA-e.

Maksimalna dubina	Broj stabala	Točnost % (k = 10)	Vrijeme izvršavanja
-	20	16,55	56,92 s
-	30	16,98	1,39 min
-	60	17,40	3,15 min
15	20	16,82	27,24 s
15	30	17,11	40,69 s
15	60	17,57	1,40 min

je dvostruko kraće nakon definiranja maksimalne dubine stabla.

U usporedbi s točnosti predikcije modela nakon primjene PCA metode na ovom setu podataka, dobiven je gotovo jednak omjer točnosti kao kod prethodnog seta podataka. U ovom je slučaju također primijećen pad točnosti između 16% i 17% u svim kombinacijama broja generiranih stabala odluke i maksimalne dubine stabla. Porast broja stabala i s PCA metodom i primjenom ANOVA f-testa povećao je točnost za manje od 1%. ANOVA f-test ponovno je povećao brzinu izvršavanja u odnosu na PCA metodu.

Tablica 6.4 prikazuje rezultate primjene ANOVA f-testa na set podataka s varijacijama u potrošnji potrošača od 10%. Analizom rezultata odmah možemo ustvrditi da je u ovom slučaju točnost predikcije iznimno niska i da za neograničenu maksimalnu dubinu stabla uz 60 stabala odluke dostiže tek 17,40%. To je također dvostruko niži postotak točnosti od predikcije modela koji je treniran na originalnom skupu podataka, kao što je bio slučaj s prethodnim setom s obzirom da je predikcijski model nasumične šume na ovom originalnom setu imao točnost u rasponu od 35% do 37.5%. Na ovom je setu podataka PCA čak i podigla točnost modela u

odnosu na originalni skup za oko 1.5%-2%. Primjetna je razlika u brzini izvršavanja, gdje se primjerice u kombinaciji sa 60 stabala odluke i postavljenom maksimalnom dubinom stabla vrijeme izvršavanja smanjilo s 3,93 minute kod originalnog seta na 1,40 minuta kod seta reduciranog primjenom ANOVA f-testa. Porast broja stabala neznatno povećava točnost predikcije.

Usporedba s PCA metodom provedenom na ovom setu podataka govori nam da se program izvršava brže nakon redukcije ANOVA f-testom, no predikcijski model ponovno pokazuje značajno nižu točnost kada se trenira sa setom podataka reduciranim ANOVA f-test metodom nego što je to slučaj kod treniranja sa setom nad kojim je provedena PCA metoda. Također je utvrđeno da porastom broja stabala točnost neznatno raste, no zanimljiv je podatak da je i nakon postavljanja ograničenja maksimalne dubine stabla točnost porasla za oko 0.2%, što nije bio slučaj kod prethodnih setova podataka gdje je definiranjem ograničenja maksimalne dubine stabla točnost polako padala.

Na manjem setu podataka PCA metoda pokazala se kao bolji odabir od ANOVA f-test metode zbog kraćeg vremena treniranja modela nasumične šume uz vrlo visok postotak točnosti. Na opširnijim setovima podataka primjena ANOVA f-test metode dodatno je povećala brzinu treniranja modela u odnosu na PCA, no uz cijenu značajnog smanjenja točnosti predikcije.

U tablici 6.5 na primjeru skupa podataka s varijacijom potrošnje od $\pm$ 2.5% na jednom su mjestu prikazane usporedbe točnosti predikcije i vremena izvršavanja algoritma nasumične šume treniranog na originalnim podacima, a zatim na reduciranim podacima nakon provedbe PCA i ANOVA f-test metoda redukcije dimenzionalnosti.

Tablica 6.5 Usporedba metrika

Maksimalna dubina, broj stabala	Originalni podaci	PCA	ANOVA
-, 60	69,57 % 5,32 min	54,58 % 3,74 min	38,68 % 2,68 min
15, 60	61,85 % 3,25 min	52,34 % 2,57 min	35,53 % 1,21 min

Zaključujemo da bi obje testirane metode uz optimalnu kombinaciju parametara, npr. maksimalne dubine stabla i broja stabala odluke u modelu strojnog učenja te prikladnim odabirom broja najboljih značajki kod ANOVA-e i broja komponenti (postotka varijance) kod PCA, a možda i kombinacijom navedenih metoda mogle značajno poboljšati performanse algoritama strojnog učenja.

Poglavlje 7

Zaključak

Redukcija dimenzionalnosti podataka i optimizacija algoritama strojnog učenja predstavljaju ključne izazove u modernim sustavima analize podataka, osobito u scenarijima gdje je dostupna velika količina ulaznih informacija. Ovaj rad demonstrirao je kako se metode poput Analize glavnih komponenata (PCA) i ANOVA f-testa mogu koristiti za smanjenje dimenzionalnosti podataka, što omogućuje učinkovitije modeliranje i analizu.

Implementacijom PCA metode, podaci su reducirani na manji broj značajki, čuvajući pritom gotovo svu varijancu iz originalnog skupa. To je rezultiralo znatno bržim treniranjem predikcijskog modela nasumične šume, što je bilo osobito korisno kod velikih skupova podataka s povećanim brojem varijacija. Na primjeru scenarija s 525 tisuća redaka podataka, PCA metoda je omogućila smanjenje dimenzionalnosti na samo 2-4 značajke, a ponegdje i uz očuvanje točnosti predikcije u granicama prihvatljivog odstupanja.

S druge strane, primjena ANOVA f-testa kao filter metode pokazala je kako je moguće izravno odabrati značajke koje su statistički najrelevantnije za predikciju ciljne varijable. Ova metoda omogućila je dodatno ubrzanje obrade podataka te su se na taj način jasno pokazale prednosti u specifičnim situacijama gdje je interpretacija ključna, no pokazalo se da je mana ANOVA f-test metode značajan pad točnosti.

Usporedba metoda na različitim skupovima podataka pokazala je da PCA općenito daje bolje rezultate u smislu odnosa točnosti predikcije i vremena izvršavanja. Posebno se to očituje kod većih skupova podataka s varijacijama u potrošnji vode, gdje je PCA održao stabilnije performanse.. Za scenarije s minimalnim varijacijama,

točnost je dosezala 100% čak i s reduciranom dimenzionalnošću, dok je kod složenijih scenarija zabilježen određen postotak pada točnosti predikcije, ali značajna ušteda u vremenu obrade.

U realnim sustavima upravljanja, kao što su vodovodne mreže, smanjenje dimenzionalnosti omogućava bržu identifikaciju ključnih obrazaca, čime se olakšava detekcija kvarova i donošenje odluka u stvarnom vremenu. Time se mogu smanjiti operativni troškovi i poboljšati održavanje infrastrukture.

Iako su implementirane metode dale dobre rezultate, postoji prostor za daljnju optimizaciju, poput istraživanja drugih tehnika redukcije dimenzionalnosti ili hibridnih pristupa koji kombiniraju nekoliko metoda redukcije dimenzionalnosti s naprednim algoritmima strojnog učenja. Također, budući radovi mogli bi se fokusirati na evaluaciju metoda na stvarnim podacima iz vodovodnih sustava, što bi dodatno validiralo dobivene rezultate.

Zaključno, ovaj rad potvrđuje da metode redukcije dimenzionalnosti uz optimalan odabir određenih parametara mogu unaprijediti performanse algoritama strojnog učenja, otvarajući put za njihovu širu primjenu u inženjerskim sustavima i analizi podataka velikih dimenzija.

Bibliografija

[1] Lučin, I.; Lučin, B; Čarija, Z.; Sikirica, A.: *Data-Driven Leak Localization in Urban Water Distribution Networks Using Big Data for Random Forest Classifier*, s Interneta: https://www.mdpi.com/2227-7390/9/6/672, 2021.

[2] *Pandas introduction*, s Interneta: https://www.w3schools.com/python/pandas/pandas_intro.asp, 2024.

[3] Willems, K.: *Pandas Tutorial: DataFrames in Python*, s Interneta: https://www.datacamp.com/tutorial/pandas-tutorial-dataframe-python?dc_referrer=https%3A%2F%2Fwww.google.com%2F, 2024.

[4] Bolf, N.: *Strojno učenje*, s Interneta: https://hrcak.srce.hr/file/382926, 2021.

[5] Koehrsen, W.: *Random forest simplified*, s Interneta: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d, 2017.

[6] Jaadi, Z.: *Principal component analysis*, s Interneta: https://builtin.com/data-science/step-step-explanation-principal-component-analysis, 2024.

[7] *What is principal component analysis*, s Interneta: https://www.ibm.com/topics/principal-component-analysis, 2023.

[8] *Feature Selection Techniques in Machine Learning*, s Interneta: https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/, 2024.

[9] Tzini, E.: *Feature Selection: Embedded Methods*, s Interneta: https://medium.com/analytics-vidhya/feature-selection-embedded-methods-a7940036973f, 2020.

[10] *Filter vs Wrapper vs Embedded Methods For Feature Selection*, s Interneta: https://medium.com/@learnwithwhiteboard_digest/filter-vs-wrapper-vs-embedded-methods-for-feature-selection-8cc21e2174f7, 2024.

[11] *Filter Methods*, s Interneta: https://www.codecademy.com/article/fe-filter-methods, 2024.

[12] Kaushik, S.: *Introduction to Feature Selection methods with an example*, s Interneta: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/#3, 2024.

Sažetak

Ovaj diplomski rad analizirao je problematiku visokodimenzionalnih podataka o mjerenjima tlaka kod oštećenja vodovodnih cijevi. Problem je definiran i opisan na primjeru pojednostavljene vodovodne mreže gdje su predstavljeni obrađeni ulazni podaci. Opisani su softverski alati korišteni u rješavanju problema. Dan je kratak opis strojnog učenja i modela nasumične šume te su istražene i opisane dvije metode redukcije dimenzionalnosti podataka - Principal Component Analysis (PCA) i ANOVA f-test koje su primjenjene na dostupne podatke. Glavni cilj rada bio je utvrditi kako metode redukcije dimenzionalnosti utječu na točnost predikcijskog modela nasumične šume i vrijeme treniranja modela u odnosu na originalni set podataka. Nakon provedbe navedenih metoda, opisani su i uspoređeni njihovi rezultati te je na temelju tih rezultata izveden zaključak i dan prijedlog nekih unaprijeđenja u pristupu ovom problemu.

Ključne riječi **— strojno učenje, nasumična šuma, PCA, ANOVA**

Abstract

This thesis analyzed the issue of high-dimensional data on pressure measurements in the case of water pipes leakage. The problem was defined and described using the example of a simplified water supply network, with processed input data presented. The software tools used to address the problem were outlined. A brief description of machine learning and the Random Forest model was provided, along with an exploration and explanation of two data dimensionality reduction methods – Principal Component Analysis (PCA) and ANOVA f-test which were applied on the dataset. The main goal of the thesis was to determine how dimensionality reduction methods affect the accuracy of the Random Forest predictive model and the model's training time compared to the original dataset. After applying these methods, their results were described and compared, and conclusions were drawn based on these results, along with proposals for potential improvements in approaching this problem.

Keywords **— machine learning, random forest, PCA, ANOVA**