UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

IZBOR INSTANCI PODATAKA

(INSTANCE SELECTION)

SEMINARSKI RAD

Predmet:

Prikupljanje i predobrada podataka za mašinsko učenje

Mentor: prof. dr Aleksandar Stanimirović

Kandidat: Milica Stojanović (1701)

Niš, 2024.

Sadržaj

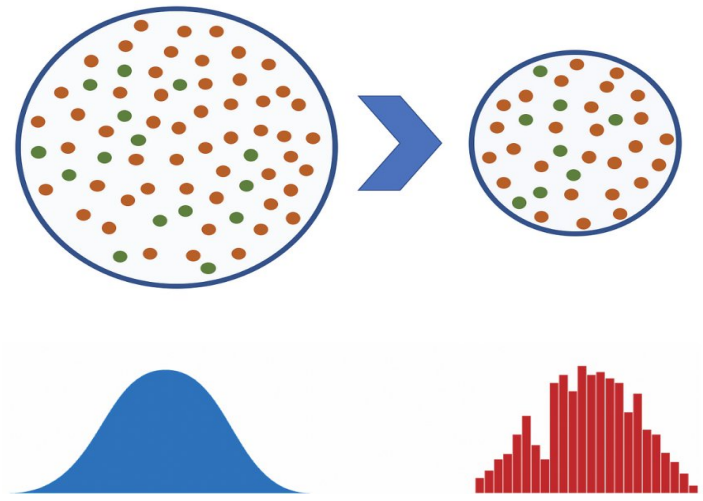
1. Uvod

U savremenim sistemima mašinskog učenja, uspeh modela u velikoj meri zavisi od kvaliteta i pripreme podataka. Prikupljanje podataka predstavlja prvi, ali ključan korak u razvoju bilo kog modela mašinskog učenja. Ovaj proces obuhvata sakupljanje informacija iz različitih izvora, kao što su baze podataka, senzori, aplikacije, društvene mreže i mnogi drugi izvori. Cilj je prikupiti podatke koji su relevantni za problem koji se rešava i koji su reprezentativni za celokupnu populaciju iz koje potiču. Kvalitet prikupljenih podataka direktno utiče na performanse modela, jer nepouzdani ili nepotpuni podaci mogu dovesti do loših rezultata i netačnih zaključaka.

Nakon što su podaci prikupljeni, prelazi se na fazu predobrade, koja je ključna za pripremu podataka za analizu. Predobrada uključuje nekoliko koraka, među kojima su čišćenje, transformacija i normalizacija podataka. Čišćenje podataka podrazumeva identifikaciju i ispravljanje grešaka, kao što su duplikati, nedostajuće vrednosti ili netačne informacije. Transformacija podataka uključuje pretvaranje podataka u format koji je prikladan za analizu, dok normalizacija osigurava da su podaci u uniformnom opsegu ili skali, čime se omogućava lakše poređenje i analiza.

Jedan od ključnih aspekata predobrade podataka je selekcija instanci. Ova metoda ima za cilj izbor reprezentativnog skupa podataka iz većeg skupa, čime se smanjuje veličina podataka i poboljšava efikasnost modela. Selekcija instanci fokusira se na identifikaciju i odabir samo onih podataka koji su najvažniji za obuku modela, dok se redundantni i irelevantni podaci eliminiraju. Ovaj proces ne samo da smanjuje potrebu za velikim računarskim resursima, već i ubrzava obuku modela, čineći je efikasnijom i manje vremenski zahtevnom.

Kroz ovaj rad, biće detaljno prikazane različite metode selekcije instanci koje pomažu u optimizaciji modela mašinskog učenja. Analiziraće se kako ove metode poboljšavaju performanse modela, omogućavajući bržu i precizniju analizu podataka. Posebna pažnja biće posvećena praktičnim primerima i analizama koje će ilustrirati kako selekcija instanci može doprineti efikasnijem i tačnijem modeliranju u različitim aplikacijama mašinskog učenja.



Slika 1. Selekcija instanci

2. Identifikacija problema

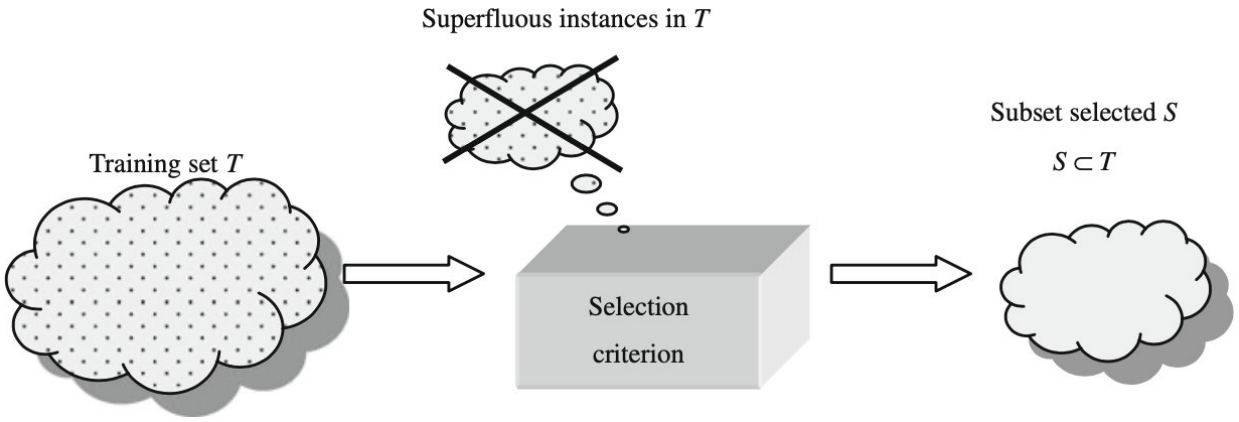
Selekcija instanci (IS) igra ključnu ulogu u smanjenju podataka u sistemima mašinskog učenja. Iako je nezavisna od selekcije karakteristika (FS), u praksi se oba procesa često primenjuju zajedno. Dok FS uklanja irelevantne i nepotrebne karakteristike, IS se fokusira na eliminaciju suvišnih instanci u velikim skupovima podataka. Ovo smanjenje broja instanci može poboljšati performanse modela, posebno u situacijama kada su podaci previše obimni za efikasnu obradu.[1]

Omogućavanje (Enabling): Datasetovi često sadrže milione podataka, što može biti previše za efikasnu obradu i obuku modela u razumnom vremenskom okviru. Veliki obim podataka može značiti duže vreme obuke i veće troškove u pogledu računarskih resursa. U ovom kontekstu, selekcija instanci igra ključnu ulogu u omogućavanju rada algoritama na velikim skupovima podataka. Smanjenjem broja instanci koje model mora da obradi, IS omogućava bržu i efikasniju obuku modela. Ovo je posebno važno u okruženjima gde je brzina obuke kritična, kao što su real-time sistemi ili komercijalne aplikacije sa ograničenim resursima.[4]

Fokusiranje (Focusing): Skupovi podataka često sadrže informacije koje nisu uvek relevantne za konkretan zadatak ili problem. Na primer, u datasetovima koji obuhvataju široke domene, kao što su zdravstveni podaci, može biti potrebno da se fokusiramo samo na specifične aspekte, kao što su podaci o određenim bolestima ili tretmanima. Selekcija instanci omogućava smanjenje skupa podataka na one koji su relevantni za konkretan zadatak, čime se poboljšava tačnost modela i smanjuje složenost obuke. Ovaj proces pomaže da se eliminišu nepotrebni ili irelevantni podaci koji mogu ometati performanse modela i otežati njegovo tumačenje.[4]

Čišćenje (Cleaning): Veliki skupovi podataka često sadrže redundantne, šumne ili netačne podatke. Ove greške mogu smanjiti kvalitet podataka i dovesti do loših performansi modela. Selekcija instanci može poboljšati kvalitet podataka tako što uklanja redundantne i šumne instance. Ovo ne samo da poboljšava kvalitet ulaznih podataka, već može dovesti do boljih rezultata u obuci modela, smanjujući mogućnost prekomernog prilagođavanja (overfitting) i povećavajući generalizaciju na neviđene podatke.[4]

U praksi, skup podataka T često sadrži beskorisne informacije za zadatak klasifikacije, kao što su suvišne instance koje mogu biti šumne ili redundantne, zbog čega je potreban proces za njihovo uklanjanje iz T. Cilj metode selekcije instanci je da se iz skupa T dobije podskup $S \subset T$ tako da S ne sadrži suvišne instance i da tačnost $Acc(S)$ bude približno jednaka tačnosti $Acc(T)$, gde $Acc(X)$ predstavlja tačnost klasifikacije korišćenjem skupa X kao trening skupa (odnosno, S se koristi za označavanje odabranog podskupa). Metode selekcije instanci mogu početi sa $S=\varnothing$ (inkrementalna metoda) ili $S=T$ (dekrementalna metoda). Razlika je u tome što inkrementalne metode uključuju instance u S tokom procesa selekcije, dok dekrementalne metode uklanjaju instance iz S tokom selekcije.[5]



Slika 2. Proces selekcije instanci[5]

Optimalni rezultat IS metode je da pronađe podskup podataka koji omogućava modelu da postigne istu ili sličnu tačnost kao model treniran na celokupnom skupu podataka, uz smanjenje veličine trening skupa. Time se ne samo poboljšava efikasnost obuke, već se i smanjuje potreba za obradom suvišnih ili nerelevantnih informacija, čime se unapređuje ukupna performansa modela.

U ovom radu, biće detaljno razmatrane različite metode selekcije instanci, analizirane prema njihovim sposobnostima za smanjenje podataka i očuvanje tačnosti, kako bi se identifikovale najbolje strategije za različite scenarije mašinskog učenja.

3. Prototype vs. Training Set Selection

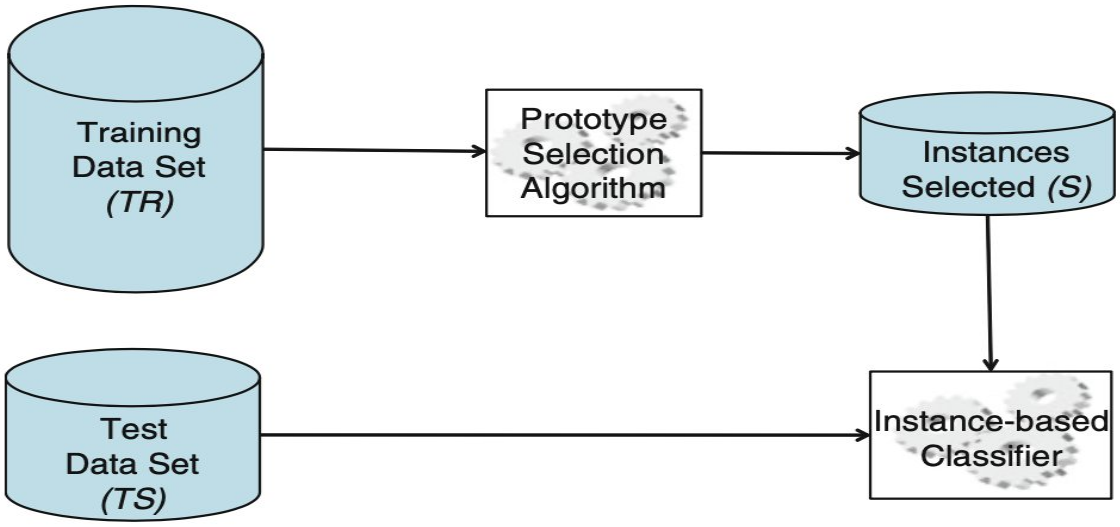
Selekcija instanci (Instance Selection - IS) se može definisati na sledeći način: neka je *Xp* instanca, gde *Xp* = (*Xp*1, *Xp*2,..., *Xpm*, *Xpc*), pri čemu *Xp* pripada klasi *c*, koja je određena komponentom *Xpc* u *m*-dimenzionalnom prostoru, a *Xpi* je vrednost *i*-te karakteristike za *p*-tu instancu. Pretpostavimo da postoji trening skup T_R koji se sastoji od *N* instanci *Xp*, kao i test skup T_S koji sadrži *t* instanci *Xp*. Neka je $S \subset T$ podskup odabranih uzoraka, koji su rezultat rada IS algoritma. Klasifikacija novih obrazaca iz test skupa T_S se zatim vrši DM (data mining) algoritmom koji operiše nad *S*. Celokupan skup podataka označava se kao *D*, koji predstavlja uniju T_R i T_S.[4]

Razlika između selekcije prototipova (Prototype Selection - PS) i selekcije trening seta (Training Set Selection - TSS) počiva u njihovim različitim ciljevima i načinima primene, iako su oba pristupa zasnovana na metodama selekcije instanci (IS). Dok PS metode služe za smanjenje broja instanci zadržavajući efikasnost u klasifikaciji na osnovu distance između instanci, TSS metode omogućavaju širu primenu selekcije instanci na sve tipove modela, poboljšavajući ukupnu preciznost i interpretabilnost modela. Iako je većina istraživanja usmerena na PS metode, TSS beleži porast popularnosti zbog svoje fleksibilnosti u različitim domenima.

U početku, mnogi predlozi za selekciju najrelevantnijih podataka iz trening seta bili su usmereni na algoritam K najbližih suseda (KNN), koji se koristi za klasifikaciju zasnovanu na instancama. Kasnije, sa uvođenjem termina učenje zasnovano na instancama (instance-based learning), poznato i kao lenjo učenje (lazy learning), pojam selekcije prototipova postao je centralna tema, obuhvatajući različite varijante kao što su redukcija prototipova, apstrakcija prototipova i generisanje prototipova.

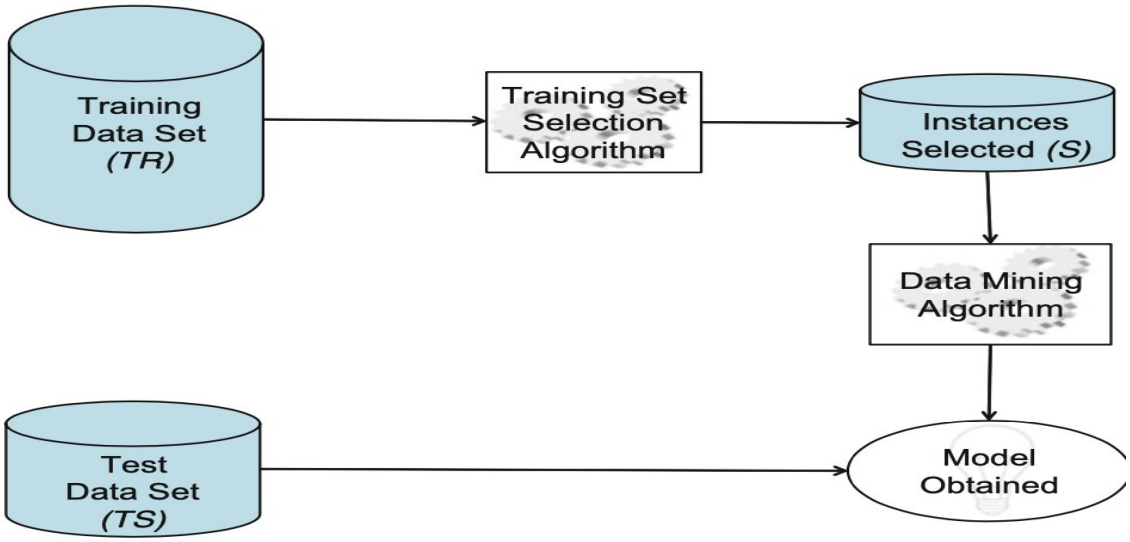
Danas, selekcija prototipova nije ograničena samo na instance-based algoritme kao što je KNN, već se primenjuje i na druge metode mašinskog učenja, kao što su odlučujuća stabla, veštačke neuronske mreže (ANNs) i mašine sa podrškom vektora (SVMs). Međutim, nije uvek jasno kada je određeni IS metod primenljiv na sve algoritme, jer zavisi od konteksta i specifičnosti modela.

Glavni zadatak (PS) metoda jeste odabir podskupa instanci koje nude optimalnu tačnost klasifikacije i smanjenje obima podataka, uz zadržavanje odgovarajuće metrike sličnosti ili udaljenosti između instanci. Ove metode su sve popularnije u redukciji podataka, jer omogućavaju bolju interpretaciju i ubrzanje procesa treniranja modela.[4]



Slika 3.1. PS proces [4]

Selekcija trening skupa (Training Set Selection - TSS) se odnosi na odabir podskupa instanci unutar trening skupa, a primarno je usmerena ka poboljšanju performansi modela, bilo kroz povećanje tačnosti, smanjenje složenosti ili poboljšanje interpretabilnosti. Dok je PS prvenstveno fokusiran na instance-based klasifikatore, TSS obuhvata metode kao što su neuronske mreže, SVM, regresija, vremenske serije i drugi modeli. Na taj način, TSS može unaprediti modele u različitim paradigmama učenja, uključujući regresiju, prognoziranje vremenskih serija, neuravnotežene podatke i učenje sa više instanci.[4]



Slika 3.2. TSS proces [4]

4. Prototype Selection

Prototype Selection (PS) metode igraju ključnu ulogu u procesima smanjenja podataka, optimizaciji performansi modela i poboljšanju efikasnosti prilikom klasifikacije. U ovom delu, razmatraju se različite vrste selekcije, smerove pretrage, evaluaciju pretrage i kriterijume za poređenje metoda. Razumevanje ovih aspekata omogućava efikasnije smanjenje obima podataka dok se očuvava ili poboljšava tačnost modela, pružajući osnovu za optimalno upravljanje instancama tokom procesa selekcije.

4.1. Vrste selekcije

Vrste selekcije se uglavnom razlikuju po načinu na koji algoritmi pretražuju i biraju instance koje će biti zadržane. Na osnovu toga, možemo razlikovati tri glavna pristupa:

1. kondenzaciju
2. ediciju
3. hibridne metode.

Kondenzacione metode su fokusirane na zadržavanje tačaka koje se nalaze blizu granica odluka, poznatih kao granične tačke. Osnovna ideja kondenzacije je da unutrašnje tačke, koje se nalaze unutar definisanih klasa, ne utiču značajno na odluke klasifikatora. Granične tačke su te koje formiraju granice između klasa, i njihovo uklanjanje bi moglo imati negativne posledice na tačnost modela. Zbog toga se unutrašnje tačke uklanjaju, dok se granične tačke zadržavaju. Na taj način, kondenzacione metode postižu visoku stopu redukcije podataka, jer u većini skupova podataka ima manje graničnih tačaka nego unutrašnjih. Međutim, jedan od glavnih izazova kondenzacije je da, iako model može zadržati tačnost na skupu za obuku, može doći do smanjenja tačnosti na skupu za testiranje. Razlog za to je potencijalno previše agresivno uklanjanje unutrašnjih tačaka, što može rezultirati modelom koji je previše "oprezan" i ne generalizuje dobro na nove, nepoznate podatke.[4]

Edicione metode imaju za cilj da uklone granične tačke koje deluju kao šum ili se ne slažu sa susednim instancama. Ove metode su usmerene na poboljšanje kvaliteta skupa podataka tako što uklanjaju tačke koje narušavaju konzistentnost granica između klasa. Kada se uklone te šumovite tačke, granice između klasa postaju glađe i stabilnije, što poboljšava generalizaciju modela na test podacima. Međutim, edicione metode ne uklanjaju unutrašnje tačke koje ne doprinose direktno granicama odluka. Ova strategija omogućava poboljšanje tačnosti modela na test skupu, ali često ne rezultira visokom stopom redukcije podataka, jer veliki broj unutrašnjih tačaka ostaje netaknut.[4]

Hibridne metode kombinuju pristupe kondenzacije i edicije kako bi postigle najbolji mogući balans između redukcije podataka i poboljšanja tačnosti modela. Cilj hibridnih metoda je da pronađu najmanji mogući podskup instanci koji ne samo da održava, već i poboljšava tačnost modela na test podacima. Ove metode omogućavaju uklanjanje i unutrašnjih i graničnih tačaka, koristeći kombinaciju kriterijuma iz oba prethodna pristupa. Kao rezultat, hibridne metode često postižu odlične rezultate sa veoma malim brojem odabranih instanci. KNN (k-nearest neighbors) klasifikator je posebno pogodan za ove metode, jer se njegove performanse mogu značajno poboljšati čak i kada se koristi mali podskup instanci.[4]

Sve tri vrste selekcije imaju svoje specifične prednosti i izazove. Kondenzacija postiže visoku redukciju podataka, ali može negativno uticati na generalizaciju. Edicija poboljšava tačnost modela uklanjanjem šumovitih graničnih tačaka, ali ne pruža značajnu redukciju. Hibridne metode kombinuju najbolje od oba sveta, smanjujući broj instanci, a pritom poboljšavajući tačnost modela na test podacima. Na osnovu prirode problema, odabir odgovarajuće metode selekcije može značajno uticati na performanse modela.

4.2. Smer pretrage

Prilikom traženja podskupa S prototipova koji će biti sačuvani iz skupa za obuku T_R, postoji nekoliko različitih smerova u kojima pretraga može ići: inkrementalna, dekrementalna, serijska, mešovita i fiksna pretraga. Svaka od ovih metoda ima svoje prednosti i mane, a izbor zavisi od prirode problema, dostupnih resursa i ciljeva smanjenja podataka.

Inkrementalna pretraga: Inkrementalna pretraga počinje sa praznim podskupom S i dodaje svaku instancu iz T_R u S ako zadovoljava određene kriterijume. Ovaj algoritam zavisi od redosleda prikazivanja instanci, što može biti veoma važno. Redosled prikazivanja instanci u T_R trebalo bi da bude nasumičan, jer inkrementalni algoritam po definiciji mora biti sposoban da obradi nove instance kako postanu dostupne, bez potrebe da sve budu prisutne od početka. Međutim, neki noviji inkrementalni pristupi su nezavisni od redosleda, jer dodaju instance u S na inkrementalan način, ali istovremeno pregledaju sve dostupne instance kako bi odabrali koja instanca će biti dodata sledeća.[4]

Jedna od prednosti inkrementalnih šema je ta što, ukoliko se instance pojave kasnije, posle završetka obuke, mogu se dodavati u S prema istim kriterijumima. Ova sposobnost može biti od velike pomoći prilikom rada sa tokovima podataka ili učenjem na mreži. Još jedna prednost inkrementalnih algoritama je ta što mogu biti brži i koristiti manje memorije tokom faze učenja nego neinkrementalni algoritmi. Glavni nedostatak inkrementalnih algoritama je taj što moraju donositi odluke na osnovu malog broja informacija i stoga su podložni greškama sve dok više informacija ne postane dostupno.

Dekrementalna pretraga: Dekrementalna pretraga počinje sa $S=T_R$, i zatim traži instance koje treba ukloniti iz *S*. Redosled prikazivanja instanci je ponovo važan, ali za razliku od inkrementalnog procesa, svi primeri za obuku su dostupni za pregledanje u bilo kom trenutku.[4]

Jedan od nedostataka dekrementalnog pristupa je taj što ima viši računski trošak u poređenju sa inkrementalnim algoritmima. Pored toga, faza učenja mora biti izvedena offline jer dekrementalni pristupi zahtevaju dostupnost svih mogućih podataka. Međutim, ako primena dekrementalnog algoritma rezultira značajnim smanjenjem memorije, dodatni računski napori tokom faze učenja mogu se isplatiti kasnijim uštedama prilikom izvršavanja.

Serijska (batch) pretraga: Drugi način primene PS procesa je u batch modu. To podrazumeva odlučivanje o tome da li svaka instanca zadovoljava kriterijum za uklanjanje pre nego što se bilo koja od njih ukloni. Zatim se sve instance koje zadovoljavaju kriterijum uklanjaju odjednom. Kao i kod dekrementalnih algoritama, serijska obrada pati od povećane vremenske složenosti u poređenju sa inkrementalnim algoritmima.[4]

Mešovita pretraga: Mešovita pretraga počinje sa unapred odabranim podskupom *S* (nasumično ili selektovano inkrementalnim ili dekrementalnim procesom) i iterativno može dodavati ili uklanjati bilo koju instancu koja zadovoljava specifični kriterijum. Ovaj tip pretrage omogućava ispravke prethodno izvršenih operacija, a njena glavna prednost je to što omogućava lakše postizanje dobrih podskupova instanci prilagođenih za tačnost modela. Obično ima iste nedostatke kao i dekrementalni algoritmi, ali to u velikoj meri zavisi od specifičnog predloga. Važno je napomenuti da su ovi algoritmi usko povezani sa inkrementalnim pristupima koji su nezavisni od redosleda, ali u ovom slučaju je dozvoljeno uklanjanje instanci iz *S*.[4]

Fiksna pretraga: Fiksna pretraga je podfamilija mešovite pretrage u kojoj broj dodatih i uklonjenih instanci ostaje isti. Tako je broj konačnih prototipova određen na početku faze učenja i nikada se ne menja.[4]

4.3. Evaluacija pretrage

KNN je jednostavna tehnika koja se može koristiti za usmeravanje pretrage u algoritmu za selekciju prototipova (PS). Cilj je napraviti predikciju na osnovu nepotpune selekcije i uporediti različite selekcije. Ova karakteristika utiče na kriterijum kvaliteta i može se podeliti na:

1. **Filter**: Kada se pravilo kNN koristi za delimične podatke kako bi se odredili kriterijumi za dodavanje ili uklanjanje instanci, bez primene validacije "leave-one-out" kako bi se dobila dobra procena tačnosti generalizacije. Upotreba podskupa podataka za svaku odluku povećava efikasnost ovih metoda, ali tačnost možda neće biti poboljšana.[4]
2. **Wrapper**: Kada se pravilo kNN koristi za celokupni obučeni skup sa primenom validacije "leave-one-out". Kombinacija ova dva faktora omogućava dobijanje odlične procene tačnosti generalizacije, što pomaže u postizanju bolje tačnosti na testnim podacima. Međutim, svaka odluka zahvata potpunu računsku obradu kNN pravila preko obučnog skupa, što može učiniti fazu učenja računski skupom.[4]

4.4. Kriterijumi za upoređivanje metoda selekcije prototipova

Kada se upoređuju metode selekcije prototipova (PS), koriste se različiti kriterijumi za ocenjivanje relativnih snaga i slabosti svake metode. Ovi kriterijumi uključuju smanjenje potrebe za skladištenjem, toleranciju na šum, tačnost generalizacije i vremenske zahteve.

Smanjenje potreba za skladištenjem: Jedan od glavnih ciljeva metoda PS je smanjenje zahteva za skladištenjem. Osim toga, blisko povezan cilj je ubrzanje klasifikacije. Smanjenje broja instanci koje se čuvaju obično vodi smanjenju vremena potrebnog za pretragu kroz ove primere i klasifikaciju novog ulaznog vektora. Efikasnije skladištenje može značiti bržu obradu i manje zahteve za memorijom.[4]

Tolerancija na šum: Dva glavna problema mogu nastati u prisustvu šuma. Prvi problem je da će vrlo malo instanci biti uklonjeno jer su mnoge instance potrebne za održavanje šumovitih granica odluka. Drugi problem je što tačnost generalizacije može patiti, posebno ako se šumovite instance zadrže umesto dobrih instanci. Dakle, sposobnost metode da se nosi sa šumom može značajno uticati na njen ukupni učinak.[4]

Tačnost generalizacije: Uspešna metoda će često biti u mogućnosti da značajno smanji veličinu obučavačkog skupa bez značajnog smanjenja tačnosti generalizacije. To znači da bi metoda trebala da očuva visok nivo tačnosti na neviđenim podacima, čak i nakon smanjenja broja instanci u obučavačkom skupu.[4]

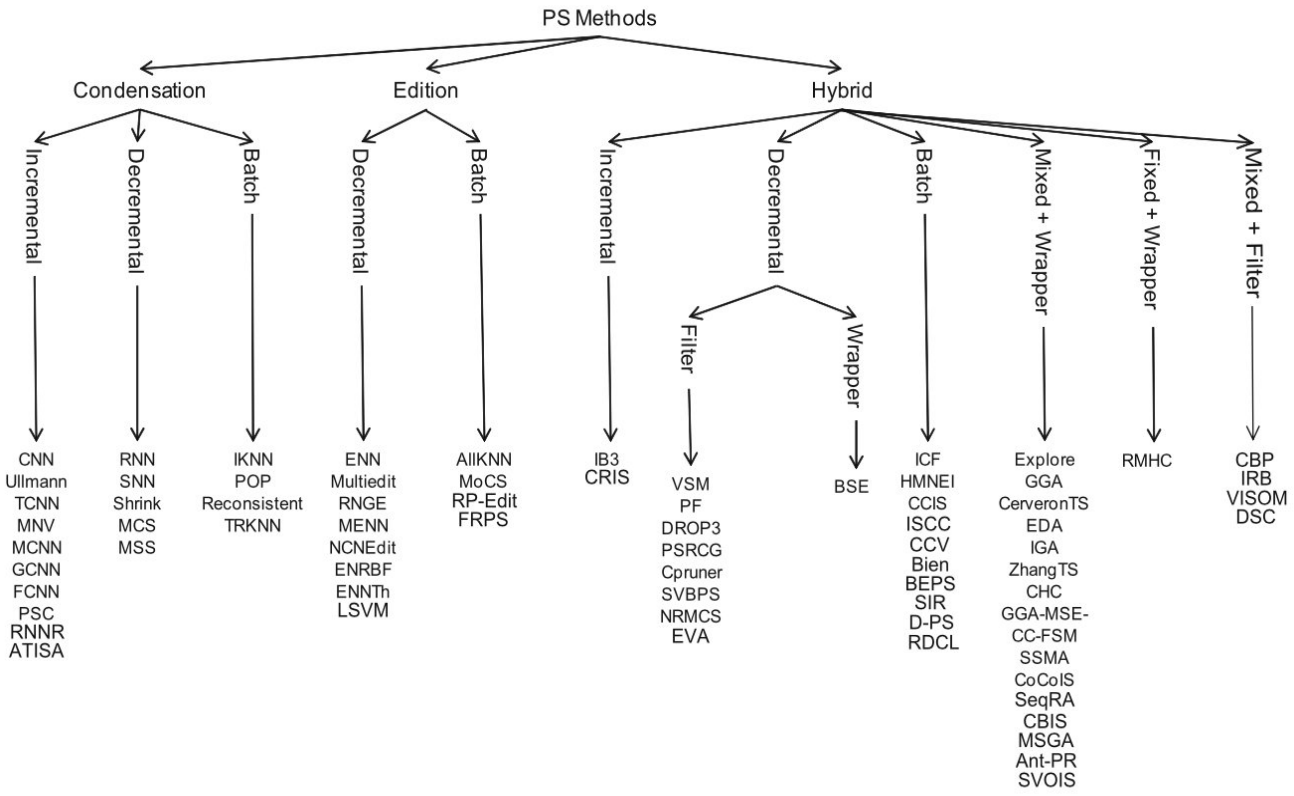
Vremenski zahtevi: Obično se proces učenja vrši samo jednom na obučavačkom skupu, pa se može činiti da vremenski zahtevi nisu presudni. Međutim, ako faza učenja traje predugo, može postati nepraktična za stvarne aplikacije. Efikasne metode treba da optimizuju vreme potrebno za učenje kako bi bile korisne u praktičnim scenarijima.[4]

Ovi kriterijumi pomažu u proceni različitih metoda selekcije prototipova kako bi se odabrala ona koja najbolje odgovara specifičnim potrebama i zahtevima projekta ili aplikacije.

5. Prototype selection metode

Osobine koje su prethodno proučavane mogu se koristiti za kategorizaciju metoda selekcije prototipova (PS) predloženih u literaturi. Smer pretrage, vrsta selekcije i evaluacija pretrage mogu se razlikovati među PS metodama i čine skup osobina koje su specifične za način funkcionisanja PS metoda. Ovaj deo predstavlja taksonomiju PS metoda na osnovu ovih osobina.

Slika 5. ilustruje kategorizaciju prema hijerarhiji zasnovanoj na sledećem redosledu: vrsta selekcije, smer pretrage i evaluacija pretrage. Ovo omogućava razlikovanje među porodicama metoda i procenu veličine svake od njih. Jedan od ciljeva ovog poglavlja je da se istaknu najbolje metode u zavisnosti od njihovih osobina, uzimajući u obzir da osobine mogu odrediti pogodnost korišćenja određenog sistema.



Slika 5. PS taksonomija[4]

5.1. Condensed Nearest Neighbor (CNN)

Većina metoda selekcije prototipova zasnovanih na k-NN klasifikatoru predložena je na osnovu rada Covera i Harta (1967). Jedna od prvih metoda je Condensed Nearest Neighbor (CNN) koju je predstavio Hart 1968. Glavni cilj CNN-a je da se kreira smanjeni skup podataka za obuku koji očuvava sposobnost tačne klasifikacije novih instanci. Ova metoda funkcioniše na osnovu principa odbacivanja instanci koje ne doprinose dodatnim informacijama u procesu klasifikacije, čime se pojednostavljuje skup podataka, a njegova klasifikaciona tačnost se održava.[5]

Pregled Algoritma [2]:

1. Inicijalizacija: Algoritam CNN-a započinje nasumičnim odabirom jedne instance iz svake klase u originalnom skupu za obuku *T* i dodaje te instance u odabrani skup *S*. Početni korak osigurava da *S* sadrži barem jednog predstavnika svake klase.
2. Klasifikacija i ažuriranje: Svaka instanca u preostalom skupu *T* se zatim klasifikuje koristeći k-NN (pri čemu sada *S* postaje obučeni skup, a *T* testni skup). Ako je neka instanca iz *T* pogrešno klasifikovana, ona se dodaje u *S*. Razlog za ovaj pristup je što pogrešno klasifikovana instanca verovatno pruža vredne informacije koje su nedostajale u početnom skupu *S*, a njeno uključivanje pomaže u poboljšanju ukupne tačnosti klasifikacije.
3. Iteracija: Ovaj proces klasifikacije instanci iz *T* i ažuriranja *S* se ponavlja sve dok sve instance u *T* ne budu tačno klasifikovane. Očekuje se da će finalni skup *S* sadržati instance koje su ključne za očuvanje tačnosti klasifikacije celog skupa podataka.

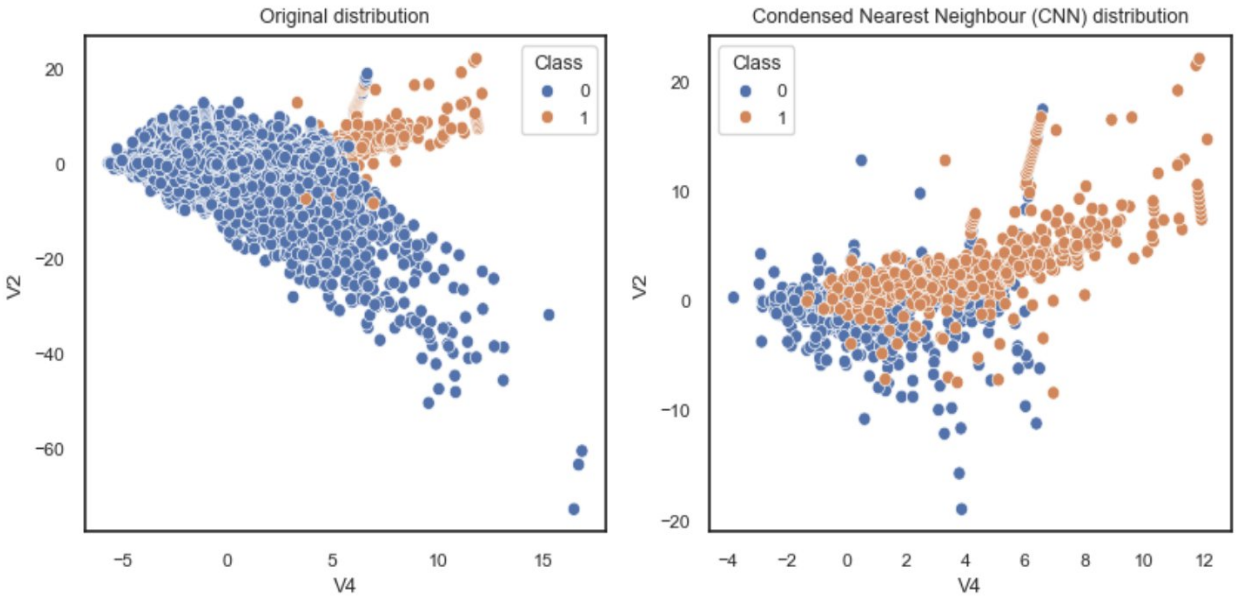
Algorithm 1 CNN algorithm

Require: $\mathbf{T}$
  $m \leftarrow |\mathbf{T}|$;
  $\mathbf{S} \leftarrow \mathbf{x}_1$;
  $p \leftarrow 0$;
  while $|\mathbf{S}| > p$ **do**
    $p \leftarrow |\mathbf{S}|$
    for $i = 1 \ldots m$ **do**
      $\bar{C}(\mathbf{x}_i) = \mathrm{kNN}(\mathbf{S}, \mathbf{x}_i)$
      if $\bar{C}(\mathbf{x}_i) \neq C(\mathbf{x}_i)$ **then**
        $\mathbf{S} \leftarrow \mathbf{S} \cup \mathbf{x}_i$;
        $\mathbf{T} \leftarrow \mathbf{T} \setminus \mathbf{x}_i$;
      end if
    end for
  end while
  return $\mathbf{S}$

Slika 5.1.1. CNN algoritam[2]

Prednosti: CNN je relativno brz i pruža umereni nivo kompresije podataka. Njegova inkrementalna priroda omogućava efikasno rukovanje novim instancama, što ga čini korisnim za skupove podataka gde se instance dodaju tokom vremena. Pored toga, CNN pomaže u smanjenju zahteva za skladištenje i računarskog opterećenja fokusiranjem na bitne instance.

Ograničenja: Kvalitet rezultata dobijenih pomoću CNN-a može biti pogođen nivoom šuma u podacima i nasumičnim redosledom u kojem se instance evaluiraju. Pošto CNN može zadržati šumne instance zbog njihove sklonosti ka pogrešnoj klasifikaciji, često se preporučuje primena algoritma za filtriranje šuma (kao što je Edited Nearest Neighbor, ENN) pre primene CNN-a kako bi se poboljšala njegova efikasnost.



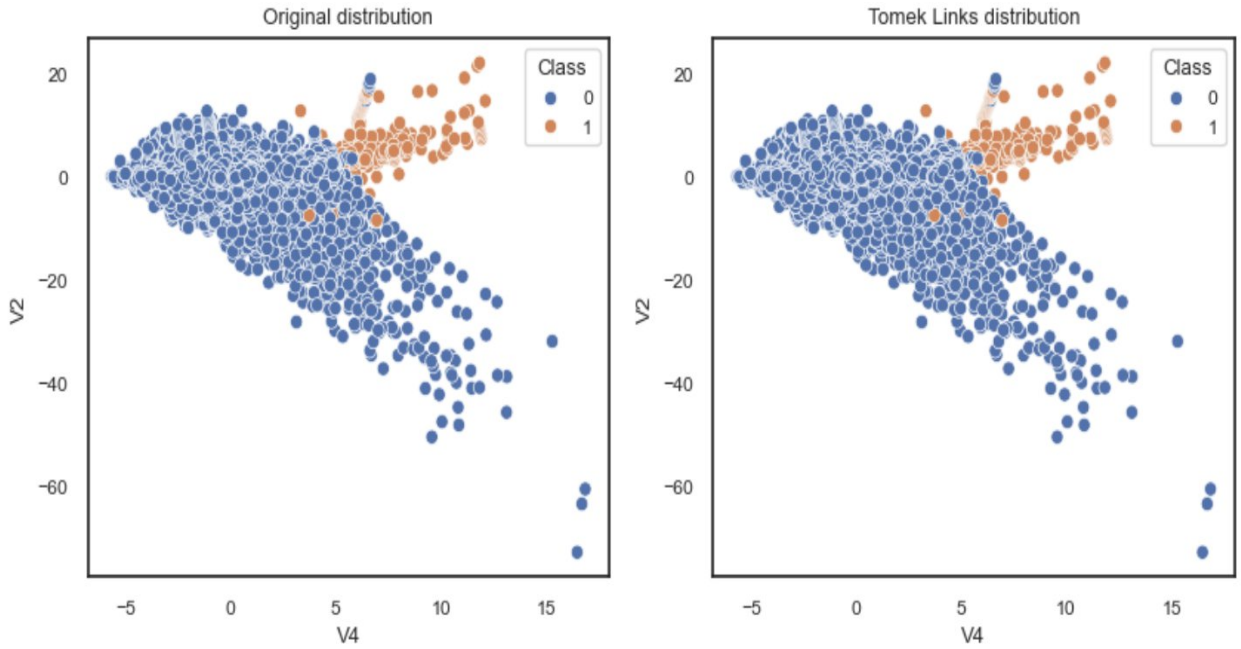
Slika 5.1.2. Primer CNN algoritma

5.2. Tomek Condensed Nearest Neighbor (TCNN)

Tomek Condensed Nearest Neighbor (TCNN) predstavlja dve modifikacije originalnog CNN algoritma, koje su predložene kako bi se unapredila efikasnost selekcije instanci. Tomekove modifikacije CNN algoritma imaju za cilj poboljšanje preciznosti i efikasnosti selekcije prototipova tako što dodatno obrađuju pogrešno klasifikovane instance i fokusiraju se na relevantne podskupove podataka.[4]

Prva modifikacija: Ova metoda je slična CNN algoritmu, ali postoji ključna razlika u načinu na koji se rukuje sa pogrešno klasifikovanim instancama. Kada se neka instanca *Xi* pogrešno klasifikuje (jer je njen najbliži sused iz skupa *S*, *s*, iz suprotne klase), umesto da se ta instanca *Xi* odmah doda u skup *S*, algoritam traži najbližeg suseda od *s*, koji pripada istoj klasi kao i *Xi*, i dodaje tu instancu u skup *S*. Ovaj pristup pomaže u boljoj identifikaciji relevantnih instanci iz klase kojoj pripada *Xi*, umesto automatskog dodavanja pogrešno klasifikovane instance.

Druga modifikacija: Ova metoda se razlikuje od originalnog CNN algoritma po tome što, umesto da koristi sve instance iz originalnog skupa *T* za izgradnju skupa *S*, koristi samo podskup *F* iz skupa *T*. Skup *F* se sastoji od instanci iz *T* koje imaju najbliže susede iz iste klase. Na taj način, samo one instance koje su već dobro klasifikovane sa svojim najbližim susedima se uzimaju u obzir, što može doprineti boljoj efikasnosti algoritma.



Slika 5.2. Primer Tomek Links algoritma

5.3. Editted Nearest Neighbor (ENN)

ENN (Editovani Najbliži Susedi) je algoritam za filtriranje šuma koji je razvio Vilson (Wilson). Glavni cilj ovog algoritma nije kompresija podataka, već poboljšanje tačnosti klasifikacije uklanjanjem šumovitih i netačnih instanci iz trening skupa. Na ovaj način, ENN pomaže u stvaranju čistijeg i pouzdanijeg skupa podataka za obučavanje modela. Iako njegova primena ne rezultira značajnim smanjenjem broja instanci, čisti skup podataka postaje ključ za precizniju klasifikaciju.

Algoritam funkcioniše tako što koristi k-NN (k-najbližih suseda) algoritam da predvidi klasu svake instance u trening skupu. Ako predviđena klasa instance ne odgovara njenoj stvarnoj klasi, ta instanca se smatra šumom ili greškom. Sve takve instance, koje su pogrešno klasifikovane, bivaju obeležene za uklanjanje. Nakon što su sve šumovite instance identifikovane, one se brišu iz trening skupa. To omogućava modelu da radi sa podacima koji su precizniji i manje skloni greškama zbog prisustva šuma.[2]

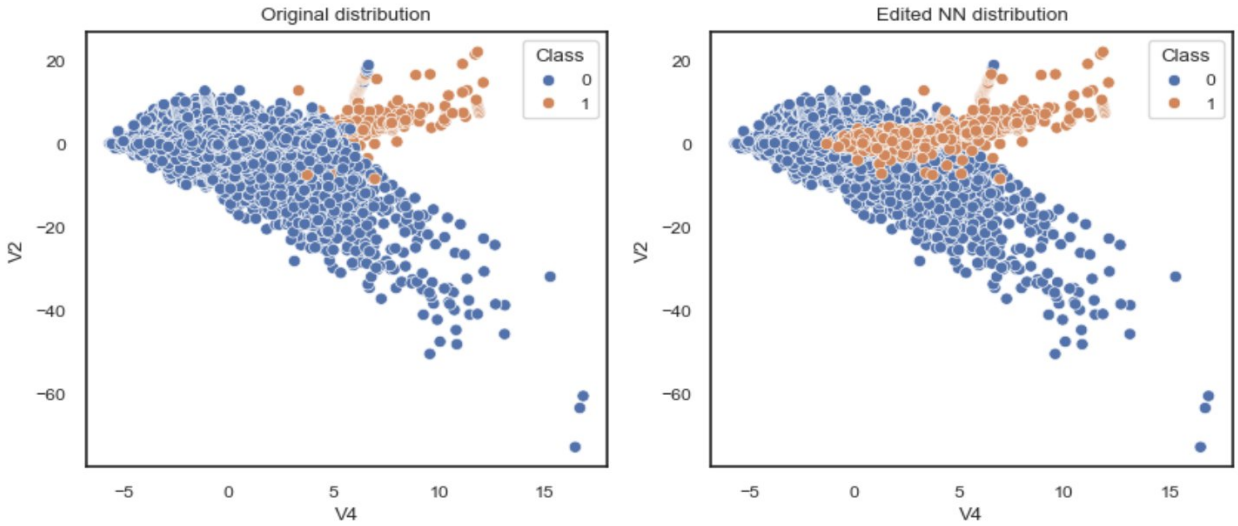
Iako ENN obično ne postiže značajnu kompresiju podataka (tipično uklanja manje od 10% instanci), njegova glavna snaga leži u poboljšanju tačnosti klasifikacije. Zbog toga se često koristi kao uvodni korak pre primene algoritama za selekciju instanci, poput CNN (Condensed Nearest Neighbor) i drugih kondenzacionih metoda.[2]

Važno je napomenuti da redosled primene ovih algoritama igra ključnu ulogu. ENN treba uvek prvo primeniti, jer uklanjanjem šumovitih instanci on pomaže da se eliminišu lažni susedi i lažne granice odluke. Ovo čini da kasniji algoritmi za selekciju instanci, poput CNN-a, funkcionišu efikasnije, jer se fokusiraju na tačne instance i prave granice između klasa.

Algorithm 2 ENN algorithm

Require: $\mathbf{T}$
  $m \leftarrow |\mathbf{T}|$;
  for $i = 1 \ldots m$ **do**
    $marked_i = 0$;
    $\bar{C}(\mathbf{x}_i) = \text{kNN}(\mathbf{T} \setminus \mathbf{x}_i, \mathbf{x}_i)$;
    if $C(\mathbf{x}_i) \neq \bar{C}(\mathbf{x}_i)$ **then**
      $marked_i = 1$;
    end if
  end for
  for $i = 1 \ldots m$ **do**
    if $marked_i == 1$ **then**
      $\mathbf{T} = \mathbf{T} \setminus \mathbf{x}_i$;
    end if
  end for
  $\mathbf{S} \leftarrow \mathbf{T}$;
  return $\mathbf{S}$

Slika 5.3.1. ENN algoritam[2]



Slika 5.3.2. Primer ENN algoritma

5.4. All-kNN i RENN

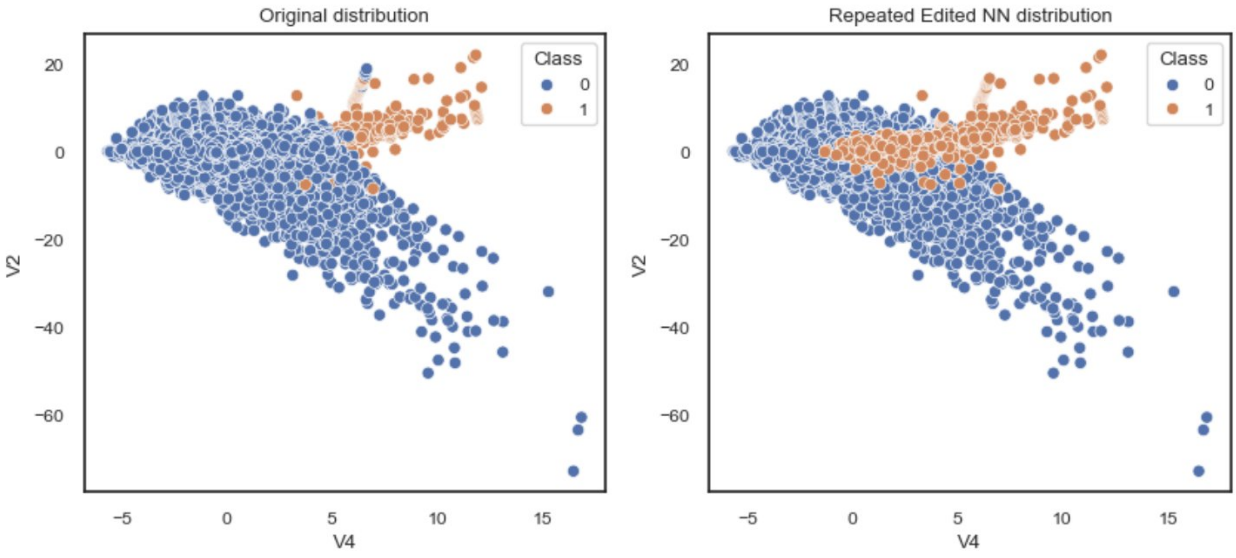
All-kNN je unapređena verzija ENN algoritma, koja radi na principu ponavljanja ENN postupka sa različitim vrednostima za broj najbližih suseda (k). Svaki put kada se ENN primeni sa drugačijim k, različite instance mogu biti prepoznate kao šumovite i odstranjene iz trening skupa. Ideja je da se temeljitije uklone nebitni ili problematični podaci jer će za različite vrednosti k različite instance biti ocenjene kao šum. Ovo može dovesti do boljih rezultata u nekim slučajevima, jer pruža dublju analizu podataka iz različitih uglova. Ipak, rezultati mogu varirati u zavisnosti od prirode problema – u nekim slučajevima, osnovni ENN može biti dovoljni ili bolji.[2]

RENN (Ponavljani Editovani Najbliži Susedi) je proširenje ENN algoritma, pri čemu se postupak uklanjanja šumovitih instanci ponavlja sve dok se više nijedna instanca ne može ukloniti. Dakle, RENN nastavlja da čisti podatke dok se svi šumoviti podaci ne odstrane. Ovaj metod je koristan za dublje čišćenje podataka, jer omogućava uklanjanje više šumovitih instanci koje možda nisu prepoznate u prvim iteracijama. Zbog ovoga, RENN može biti efikasniji u slučajevima gde podaci sadrže više šuma, ali opet, zavisi od specifičnog problema koliko će biti uspešan u poređenju sa osnovnim ENN algoritmom.[2]

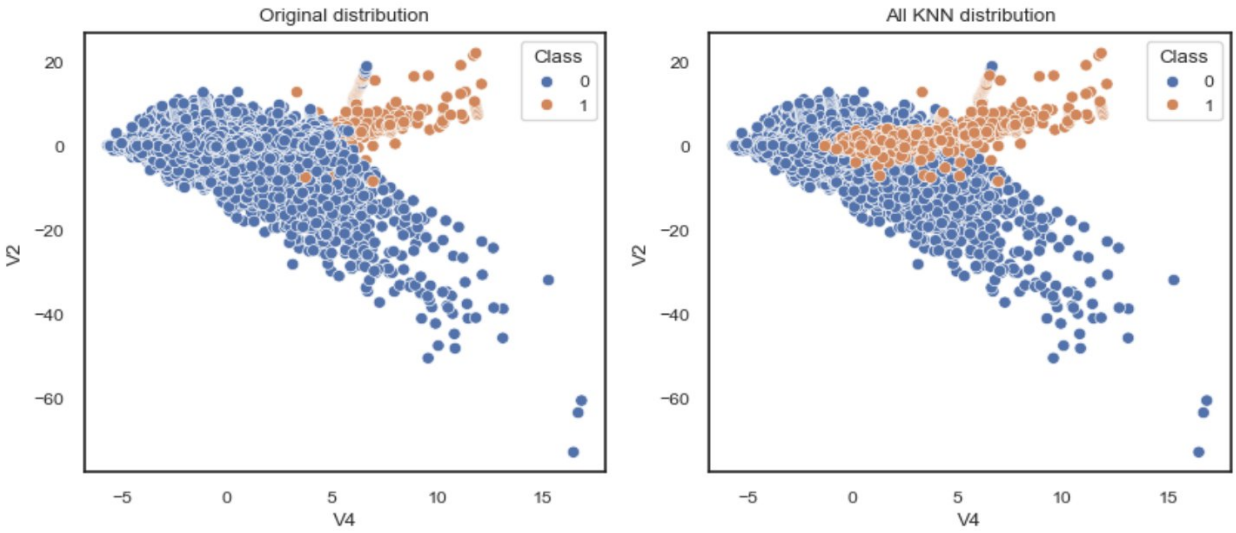
Algorithm 3 RENN algorithm

Require: $\mathbf{T}$
  $flag \leftarrow$ **true**
  while flag **do**
    $flag \leftarrow$ **false**
    $\bar{\mathbf{S}} = \mathrm{ENN}(\mathbf{S}, \mathbf{T})$
    if $\bar{\mathbf{S}} \neq \mathbf{S}$ **then**
      $flag \leftarrow$ **true**
    end if
  end while
  return $\mathbf{S}$

Slika 5.4.1. RENN algoritam[2]



Slika 5.4.2. Primer RENN algoritma



Slika 5.4.3. Primer All-kNN algoritma

5.5. Instance-Based Learning Algorithms Family

Porodica algoritama zasnovanih na instancama (Instance-Based Learning, IBL) predstavlja niz metoda učenja koje koriste specifične primere iz trening skupa za donošenje odluka o klasifikaciji. Ovi algoritmi ne grade eksplicitni model tokom treninga, već memoriraju primere (instance) i koriste ih direktno prilikom klasifikacije novih podataka. Najjednostavniji algoritam iz ove grupe je IB1, koji je zapravo implementacija 1-NN algoritma (algoritam najbližeg suseda). IB1 jednostavno pronalazi najbližeg suseda nove instance iz trening skupa i koristi njegovu klasu za klasifikaciju. Ovaj algoritam služi kao polazna tačka za poređenje sa naprednijim metodama učenja zasnovanim na instancama.[4]

Međutim, IB1 ima određene nedostatke, kao što je čuvanje svih instanci iz trening skupa, što može dovesti do sporih klasifikacija i prekomernog memorijskog zauzeća. Da bi se ovi problemi ublažili, razvijeni su složeniji algoritmi kao što su IB2 i IB3, koji koriste strategije selekcije instanci za optimizaciju performansi.

IB2 algoritam

IB2 predstavlja unapređeni algoritam za učenje zasnovano na instancama, koji je dizajniran da smanji broj instanci koje se čuvaju za klasifikaciju. Njegova osnovna ideja je da se memoriraju samo one instance koje su neophodne za tačnu klasifikaciju. IB2 je efikasan jer memorijalno kompresuje trening skup, ali može zadržati nepotrebne instance, naročito u prisustvu šuma u podacima. Algoritam funkcioniše na sledeći način[4]:

1. Početak: IB2 započinje sa praznim setom instanci (S).
2. Dodavanje instanci: Svaka instanca iz trening skupa (T) se klasifikuje koristeći trenutni set S. Ako instanca nije pravilno klasifikovana, dodaje se u set S. Ovo osigurava da se samo one instance koje pružaju dodatne informacije (neophodne za ispravnu klasifikaciju) čuvaju.
3. Razlika u odnosu na CNN: Za razliku od CNN algoritma, IB2 ne započinje sa po jednom instancom iz svake klase, niti ponavlja proces nakon prvog prolaska kroz trening skup. Svaka instanca se dodaje u set samo jednom, ako nije pravilno klasifikovana.

IB3 algoritam

IB3 unapređuje IB2 algoritam korišćenjem statističkih kriterijuma za odlučivanje o tome koje instance treba sačuvati u memoriji. Njegov cilj je da filtrira nepotrebne i šumovite instance, čime dodatno optimizuje performanse algoritma.[4]

1. Dodavanje instanci: Instanca se dodaje u set S samo ako se proceni da je njena tačnost statistički značajna. To znači da algoritam procenjuje da li je instanca dovoljno korisna za buduće klasifikacije.
2. Uklanjanje instanci: Algoritam takođe može ukloniti instancu iz seta S, ako statistički podaci pokazuju da ona nije dovoljno tačna. Ovaj postupak smanjuje memorijski prostor zauzet nepotrebnim instancama i poboljšava tačnost klasifikacije.
3. Statistički pragovi: IB3 koristi statističke granice (gornje i donje granice intervala pouzdanosti) za odlučivanje o dodavanju ili uklanjanju instanci, koristeći nivoe pouzdanosti od 90% za prihvatanje i 70% za odbacivanje.

IB3 je složeniji od IB2 jer koristi statističke metode da bi zadržao samo one instance koje doprinose tačnosti modela, dok uklanja nepotrebne ili šumovite instance. Na ovaj način, postiže bolju memorijsku kompresiju uz zadržavanje visokog nivoa tačnosti u klasifikaciji.

$$Bound = \frac{p + z^2/2n \pm z\sqrt{p(1-p)/n + z^2/4n^2}}{1 + z^2/n}$$

For each instance X_i in TR
  Let a be the nearest *acceptable* instance in S to X_i.
  (if there are no acceptable instances in S,
  let a be a random instance in S)
  If $class(a) \neq class(X_i)$ then add X_i to S.
  For each instance s in S
    If s is at least as close to X_i as a is
      Update the classification record of s
      and remove s from S its classification
      record is significantly poor.
Remove all non-acceptable instance from S.

Slika 5.5. IB3 algotitam[4]

5.6. Decremental Reduction Optimization Procedure Family (DROP)

Da bismo razumeli tehnike redukcije koje koristi Decremental Reduction Optimization Procedure (DROP) algoritam, prvo je potrebno definisati nekoliko ključnih pojmova. Svaka instanca *Xi* u skupu podataka ima *k* najbližih suseda, pri čemu je *k* obično mali neparan broj. Ovi najbliži susedi su primerci koji su najbliži *Xi* prema nekoj metriki udaljenosti, kao što je euklidska distanca. Pored toga, svaka instanca *Xi* ima i najbližeg neprijatelja, koji je instanca najbliža *Xi* ali pripada različitoj klasi. Osim toga, one instance koje imaju *Xi* kao jednog od svojih *k* najbližih suseda nazivaju se asocijati *Xi*. Ovi asocijati su direktno povezani sa *Xi* na osnovu udaljenosti u skupu podataka.[4]

DROP1

Koristi se sledeće osnovno pravilo za odlučivanje da li je bezbedno ukloniti instancu iz skupa instanci *S* (gde je $S=T_R$ prvobitno): Ukloni *Xi* ako barem onoliko njenih asocijata u *S* može biti pravilno klasifikovano i bez prisustva *Xi*. Algoritam DROP1 se nastavlja na sledeći način.

```
DROP1 (Training set TR): Selection set S.
Let S = TR.
For each instance Xi in S:
   Find the k + 1 nearest neighbors of Xi in S.
   Add Xi to each of its lists of associates.
For each instance Xi in S:
   Let with = # of associates of Xi classified
   correctly with Xi as a neighbor.
   Let without = # of associates of Xi classified
   correctly without Xi.
   If without ≥ with
      Remove Xi from S.
      For each associate a of Xi
         Remove Xi from a's list of neighbors.
         Find a new nearest neighbor for a.
         Add a to its new list of associates.
      For each neighbor b of Xi
         Remove Xi from b's lists of associates.
   Endif
Return S.
```

DROP2

DROP2 je metoda optimizacije skupa podataka koja se koristi za efikasnije uklanjanje instanci koje nisu od suštinskog značaja za model. Ova metoda se oslanja na modifikovano pravilo za odlučivanje o uklanjanju instanci iz skupa podataka. Osnovno pravilo je: uklonite instancu *Xi* ako barem toliko njenih asocijata u originalnom skupu T_R može biti pravilno klasifikovano i bez prisustva *Xi*.

Kada se instanca *Xi* ukloni iz skupa podataka *S*, metod omogućava da se nastavi sa radom tako što svaka instanca u *S* zadržava informacije o svojim *k+1* najbližim susedima, uključujući one koji su izvan *S*. Takođe, instanca koja je uklonjena iz *S* više ne može biti asocijat za druge instance. Ovo osigurava da instanca *Xi* može biti uklonjena samo ako njen izostanak ne negativno utiče na sposobnost klasifikacije drugih instanci u skupu.[4]

Osim toga, DROP2 menja redosled uklanjanja instanci tako što prvo sortira instance u *S* prema udaljenosti od njihovog najbližeg neprijatelja (tj. najbližeg suseda iz različite klase). Proces uklanjanja zatim počinje od instanci koje su najdalje od svog najbližeg neprijatelja, što pomaže u očuvanju ključnih informacija i smanjenju potencijalne greške u modelu.

DROP3

DROP3 je metoda za optimizaciju skupa podataka koja kombinuje pristupe iz DROP2 i ENN algoritama. Prvo, koristi ENN za uklanjanje šumnih instanci iz skupa podataka, što pomaže u čišćenju podataka pre nego što se nastavi sa selekcijom. Nakon toga, DROP3 primenjuje kriterijume slične onima u DROP2 za dodatno smanjenje skupa. U ovom procesu, instanca se uklanja samo ako zadovoljava specifične uslove koji se odnose na njen uticaj na druge instance u skupu. Ova kombinacija omogućava efikasnije smanjenje skupa podataka dok se očuvava njegova tačnost.[2]

5.7. Iterative Case Filtering (ICF)

Iterativno Filtriranje Instanci (ICF) je metod za optimizaciju skupa podataka, čiji cilj je poboljšanje tačnosti klasifikatora uklanjanjem nepotrebnih instanci. Algoritam funkcioniše u dva ključna koraka.

Prvi korak algoritma ICF koristi ENN za prepoznavanje i uklanjanje šumnih instanci iz skupa podataka. Ovaj korak pomaže u pripremi skupa podataka tako što se uklanjaju instanci koje su loše klasifikovane i koje mogu ometati proces selekcije instanci.[2]

Nakon što je skup podataka očišćen od šumnih instanci, prelazi se na drugi korak gde se definišu lokalni skupovi za svaku instancu u skupu. Za svaku instancu *xi*, **lokalni skup** *L(xi)* se definiše kao skup svih instanci koje se nalaze unutar najveće hiper sfere centrirane u *xi* i koje pripadaju istoj klasi kao *xi*. Ova hiper sfera obuhvata sve instance koje su bliže *xi* nego najbliži sused *xi* koji pripada drugoj klasi, nazvan najbližim neprijateljem.[2]

ICF koristi ove lokalne skupove za kreiranje dva važna skupa instanci: **pokriće (Coverage)** i **dohvat (Reachability)**. **Pokriće** instance *xi* predstavlja sve instance koje se nalaze u lokalnom skupu *L(xi)*. **Dohvat** predstavlja skup instanci koje koriste *xi* kao jednog od svojih najbližih suseda.[2]

Sledeći korak je procena da li da se instanca *xi* ukloni iz skupa podataka. Ako je broj instanci u dohvatu veći od broja instanci u pokriću, instanca *xi* se smatra suvišnom i uklanja se. Ovo znači da informacije koje pruža instanca *xi* mogu biti izražene pomoću drugih instanci koje se nalaze u njenoj blizini, pa je stoga nepotrebna za tačnost modela.

Proces se ponavlja sve dok se ne postigne stabilnost, tj. dok se ne odstrane sve instance koje ne doprinose značajno tačnosti modela. Na taj način, ICF pomaže u smanjenju broja instanci u skupu podataka, čime se poboljšava efikasnost i preciznost klasifikatora.

$$Coverage(X_i) = \{X_i' \in TR : X_i \in L(X_i')\},$$

$$Reachability(X_i) = \{X_i' \in TR : X_i' \in L(X_i)\},$$

5.8. Modified Selective Algorithm (MSS)

Modifikovani selektivni podskup (MSS) predstavlja unapređenu verziju selektivnog algoritma koji je prvotno predložen od strane Ritter-a. Osnovni cilj MSS-a je da poboljša proces selekcije instanci tako što se fokusira na minimizaciju broja instanci koje su potrebne za preciznu klasifikaciju, dok istovremeno osigurava da odabrani podskup instanci bude što manji, ali još uvek dovoljno reprezentativan.[3]

U osnovnom algoritmu selektivnog podskupa, definicija selektivnog podskupa *P* uključuje sve one instance koje omogućavaju tačnu klasifikaciju svih uzoraka u skupu za obuku *T* koristeći pravilo najbližeg suseda na osnovu *P*. Drugim rečima, svaka instanca u *T* mora biti bliža instancama u *P* koje pripadaju istoj klasi. Ova metoda se fokusira na to da podskup *P* bude što manji, uz istovremeno očuvanje tačnosti klasifikacije.[3]

Modifikovani selektivni algoritam (MSS) unapređuje ovu osnovnu metodu dodavanjem specifične karakteristike. Umesto da se fokusira samo na bliskost instanci u istoj klasi, MSS takođe uzima u obzir "najbliže neprijatelje" instanci. Konkretno, MSS definiše za svaku instancu *Xi* u skupu za obuku T_R, skup *Ri* koji uključuje sve instance koje su iste klase kao *Xi* i koje su bliže *Xi* nego najbliži sused *Xi* iz različite klase. Zatim, MSS bira iz skupa T_R onu instancu iz *Ri* koja je najbliža instanci iz različite klase od klase *Xi*.[4]

Ovaj pristup omogućava MSS-u da bolje identifikuje ključne instance koje su značajne za pravilnu klasifikaciju. Dok osnovni selektivni algoritam samo minimizuje broj instanci, MSS dodaje sloj kompleksnosti tako što uključuje kriterijum "najbližeg neprijatelja" da bi osigurao da odabrani podskup ne samo da bude tačan, već i da bude otporniji na moguće greške u klasifikaciji koje mogu nastati zbog sličnosti među klasama. Ovim pristupom se smanjuje broj instanci, ali se istovremeno očuvava kvalitet klasifikacije, što je posebno korisno u scenarijima kada su podaci obimni ili kada je potrebno postići visoku tačnost u prisustvu sličnih klasa.

$Q = TR$
Sort the instances $\{X_j\}_{j=1}^{n}$ according to increasing values of enemy distance (D_j).
For each instance X_i do
    $add \leftarrow FALSE$
    For each instance X_j do
        If $x_j \in Q \wedge d(X_i, X_j) < D_j$ then
            $Q \leftarrow Q - \{X_j\}$
            $add \leftarrow TRUE$
    If add then $S \leftarrow S \cup \{X_i\}$
    If $Q = \emptyset$ then return S

Slika 5.8. MSS algoritam[4]

6. Zaključak

Selekcija instanci igra ključnu ulogu u optimizaciji performansi modela u mašinskom učenju, omogućavajući smanjenje složenosti podataka uz očuvanje ili čak poboljšanje tačnosti modela. Kroz pregled različitih metoda, kao što su selekcija prototipova i trening selekcija, jasno je da svaka metoda donosi jedinstven pristup u odabiru ključnih instanci koje najbolje predstavljaju skup podataka.

Metode kao što su CNN, TCNN, i ENN nude osnovne strategije za identifikaciju i selekciju instanci na temelju njihovih odnosa sa najbližim susedima i različitim klasama. Ove tehnike omogućavaju precizno filtriranje i zadržavanje samo onih instanci koje značajno doprinose tačnosti modela. S druge strane, naprednije metode poput IB2 i IB3 koriste statističke pristupe za procenu korisnosti svake instance, dok DROP1, DROP2, i DROP3 fokusiraju se na smanjenje skupa podataka na osnovu asocijacija i udaljenosti, čime se poboljšava efikasnost procesa selekcije.

ICF i MSS predstavljaju inovativne pristupe u filtriranju i selekciji instanci, oslanjajući se na lokalne skupove i bliskost u klasifikaciji, što dodatno unapređuje tačnost i efikasnost modela. Ove metode ne samo da poboljšavaju performanse modela, već i smanjuju potrebu za velikim računskim resursima, čime se omogućava rad sa većim i složenijim skupovima podataka.

Zaključno, efikasna selekcija instanci je ključ za uspeh u različitim problemima mašinskog učenja. Razumevanje i primena različitih tehnika omogućavaju bolje upravljanje podacima, povećavaju preciznost modela i smanjuju računsku složenost. U budućnosti, integracija naprednih metoda kao što su lokalno osetljive hashing tehnike i klasterizacija može dalje unaprediti sposobnost selekcije instanci, čime se otvaraju nove mogućnosti za istraživanje i primenu u različitim oblastima.

7. Literatura

1. https://medium.com/@sabrinaherbst/an-introduction-to-instance-selection-in-data-mining-7b2ac94fb07
2. https://www.kordos.com/pdf/Instance_Selection_in_Machine_Learning.pdf
3. https://www.mdpi.com/1099-4300/19/11/583
4. http://library.lol/main/10c17e8c176dc80f1ded582fcbccba0d
5. https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/pr/2010-Olvera-AIR.pdf