UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET
Katedra za računarstvo

# IZBOR INSTANCI PODATAKA (INSTANCE SELECTION)

PRIKUPLJANJE I PREDOBRADA PODATAKA
ZA MAŠINSKO UČENJE

**Profesor:** Prof. Dr Aleksandar Stanimirović **Student:** Nikola Petrović 1466

Niš, 2023.

# Sadržaj

# Uvod

Prikupljanje i predobrada podataka predstavlja granu masinskog učenja koja je krucijalni deo svakog projekta. Od velike je važnosti da podaci koji se koriste u projektu budu maksimalno očišćeni, bez šuma u podacima, nedostajućih podataka, dupliranih podataka.. Ovo doprinosi kvalitetu rešenja, boljem radu algoritama mašinskog učenja, manjem opterećenju hardverskih resursa računara na kome se vrše izračunavanja..

U predobradi podataka, veliki problem može predstavljati količina podataka. Podaci su pre obrade u većini slučajeva u tabelarnom obliku, gde svaka kolona predstavlja jedan od atributa(eng. Feature), dok vrsta(eng. Instance) predstavlja skup vrednosti atributa koji ima neko značenje. Algoritmi mašinskog učenja koriste podatke iz ovakvih skupova za treniranje modela koji trebaju da predvide vrednost nekog atributa na osnovu ostalih atributa iz određene vrste i na osnovu ostalih instanci u setu podataka.

Brzina rada ovih algoritama direktno proporcionalno zavisi od količine podataka koja se obrađuje. Kako količina podataka na internetu iz dana u dan raste, srazmerno raste i broj instanci određenih setova podataka, s tim u vezi traže se rešenja koja će biti u stanju da obrađuju velike količine podataka, a koja će pritom davati precizna izračunavanja. Jedan od pristupa koji se primenjuje jeste korišćenje Big Data sistema, gde se veliki skupovi podataka (reda veličine GB, TB..) postavljaju na neki udaljeni server koji simulira neograničene hardverske resurse.

Međutim, postoji mogućnost fokusiranja na smanjivanje količine podataka (eng. Data reduction) u samim setovima podataka, gde će se odstraniti atributi i instance koji nisu od značaja za izračunavanje (eng. Knowledge discovery), ovakvo rešenje treba da obezbedi najmanje iste, ili bolje rezultate, od modela koji je treniran nad svim podacima. U ovu svrhu, kreirano je nekoliko pristupa [1]:

- Diskretizacija (eng. **Discretization**) - proces transformacije numeričkih u diskretne atribute(npr. starost osobe se može diskretizovati u kategorije kao sto su: “mlad”, “srednjih godina” i “stariji”). Izazov je kako pronaći najbolje opsege ili intervale u koje treba da se podele numeričke vrednosti.

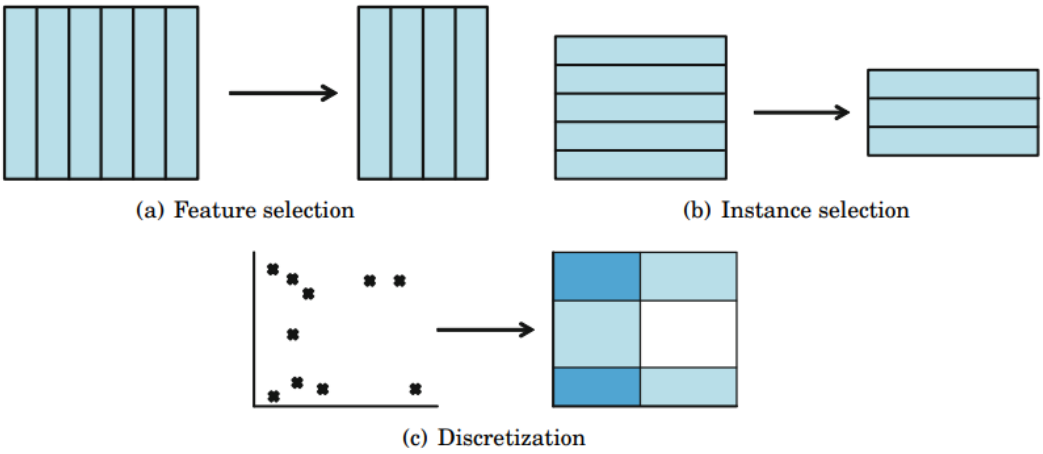
- Ekstrakcija karakteristika (eng. **Feature extraction**) - uključuje nekoliko modifikacija atributa kao što je uklanjanje jednog ili više atributa, spajanje njihovog podskupa ili kreiranje novih veštačkih.

- Generisanje instance (eng. **Instance generation**) - u ovom zadatku se kreiraju nove veštačke instance kao rezime prvobitnih. Nove instance su kreirane sa ciljem da se poboljša reprezentativnost celog skupa podataka uz smanjenje njegove veličine.

- Izbor atributa (eng. **Feature selection**) - proces odabira podskupa relevantnih atributa iz većeg skupa atributa za korišćenje u izgradnji modela. Cilj izbora

atributa je poboljšanje performansi mašinskog učenja smanjujući dimenzionalnost podataka i uklanjajući nerelevantne podatke.

- Izbor instanci podataka (eng. **Instance selection**) - pokušava da se pronađe najreprezentativniji podskup početnog skupa podataka, bez smanjenja prediktivnih mogućnosti originalnog. Drugim rečima, ako treniramo jedan algoritam sa originalnim skupom podataka, a drugi sa izabranim podskupom, oba algoritma moraju da rade na sličan način. Izbor instance se može posmatrati kao poseban slučaj generisanja instance, gde su instance koje treba generisati ograničene na originalne.

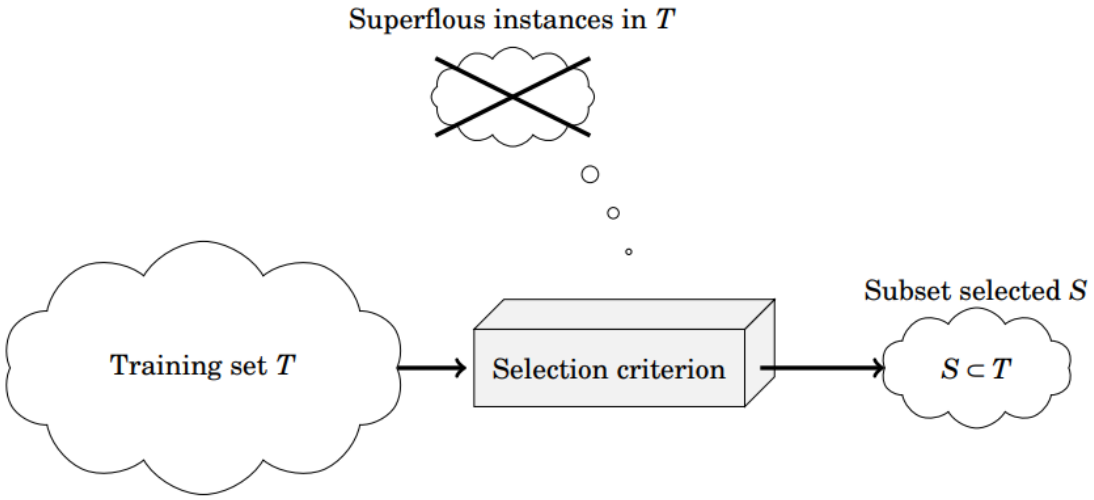



*Slika 1. Osnovni načini redukcije podataka*

Ove metode igraju centralnu ulogu u procesima smanjenja podataka. Dok izbor karakteristika ili procesi diskretizacije smanjuju složenost, izbor instance smanjuje veličinu skupa podataka.

## Izbor instanci podataka (Instance selection)

Kao što je prethodno objašnjeno, algoritmi za odabir instance imaju za cilj da smanje složenost algoritama za učenje smanjenjem broja primera skupova. Svrha ovih algoritama je da izdvoje najznačajniji podskup instanci odbacivanjem onih koji ne pružaju vredne informacije. Slika 2. ilustruje proces odabira instance. Smanjenje skupa podataka ima tri glavne prednosti: smanjuje i prostorne zahteve sistema i vreme obrade zadataka učenja, ali i uklanjanje šuma i suvišnih instanci [2].




*Slika 2. Proces selekcije instanci*

Odabrani skup instanci se može koristiti za obuku bilo koje vrste algoritama, ali tradicionalno, mnogi algoritmi za odabir instanci su razvijeni za klasifikator k najbližih suseda, ili skraćeno kNN. Iz tog razloga, termin koji se koristi za proces selekcije je takođe odabir prototipa. U ovom radu, termin selekcija instance se koristi za označavanje zadatka koji uključuje izbor podskupa instanci iz originalnog skupa podataka, bez razmatranja naknadnog algoritma koji treba da se obuči.

Kada se ispitaju skupovi podataka iz stvarnog sveta, imperativna potreba za algoritmima za izbor instanci postaje sve jasnija. S jedne strane, prosečna veličina skupa podataka postaje sve veća i veća, s druge strane, stvarni skupovi podataka obično sadrže bučne instance (eng. Noisy data), izuzetke (eng. Outliers) i anomalije (eng. Anomalies). Pokušaji da se obuči klasifikator, na primer, na osnovu miliona primera može biti težak, pa čak i nerešiv zadatak. Izbor odgovarajućeg podskupa slučajeva je stoga dobra opcija za smanjenje veličine skupa, omogućavajući njegov naknadni tretman

Izbor instanci podataka (eng. Instance selection) ima ključnu ulogu u zadatku smanjenja podataka zbog činjenice da obavlja suprotan proces u odnosu na izbor atributa (eng. Feature selection - FS). Iako je nezavistan od FS-a, u većini slučajeva, oba procesa se zajednički primenjuju. Suočavanje sa ogromnim količinama podataka može se postići smanjenjem količine podataka kao alternativa za poboljšanje skaliranja algoritama za

upravljanje podacima. FS već postiže ovaj cilj, kroz uklanjanje nerelevantnih i nepotrebnih atributa. Na ortogonalan način, uklanjanje instanci može se smatrati jednakim ili čak još efikasnijim načinom sa stanovišta smanjenja podataka u određenim aplikacijama.

Instance selection se odnosi na odabir podskupa podataka kako bi se postigao originalni cilj aplikacije za predikciju podataka kao da se koriste svi podaci. Međutim, odabir samog podskupa podataka ne predstavlja uvek izbor instanci, ovaj pristup se smatra ozbiljnom, inteligentnom operacijom kategorizacije instanci, u odnosu na relevantnost ili zavisnost od šuma u podacima u okviru zadatka koji je potrebno rešiti. S tim u vezi, popularni „Data sampling“ se ne smatra izborom instanci, jer on predstavlja metod selekcije instanci po slučajnom principu(može doći do gubitka bitnih informacija za izračunavanje modela) kako bi se dobilo na brzini izračunavanja, ne uzimajući u obzir logiku koja je potrebna za očuvanje kvaliteta kasnijeg izračunavanja.

Optimalan rezultat IS-a je minimum podskupa podataka, nezavistan od modela, koji može obavljati isti zadatak bez gubitka performansi.

$$P(A_s) = P(A_t)$$

Gde ***P*** predstavlja performanse, ***A*** predstavlja neki od algoritama masinskog učenja, $_s$ je podskup podataka koji je izdvojen i $_t$ je kompletan ili trening skup instanci podataka.
Zadaci izbora instanci [3]:

- Omogućavanje (eng. Enabling) – Kada je skup podataka preveliki, možda izvršenje algoritma ne bude moguće ili algoritam ne može da bude primenjen adekvatno. Selekcija instanci omogućava algoritmu da radi sa velikim količinama podataka, smanjujući njihov obim na manji, reprezentativan uzorak koji zadržava ključne karakteristike originalnog skupa, čime se poboljšava efikasnost i izvodljivost algoritma.

- Fokusiranje (eng. Focusing) – Podaci su formirani od mnogo informacija iz gotovo svih oblasti, ali konkretan zadatak prediktovanja se usredsređuje samo na jedan od aspekata interesa. Selekcija instanci se fokusira na podatke relevantne za traženi zadatak, uklanjajući one instance koje ne doprinose značajno rešavanju problema ili su čak štetne po performanse algoritma, čime se poboljšava kvalitet predikcije i efikasnost algoritma.

- Čišćenje (eng. Cleaning) – Odabirom relevantnih instanci uklanjaju se suvišni podaci i šum u podacima, čime se poboljšava kvalitet ulaznih podataka algoritma i time dobija poboljšanje rezultata dobijenih primenom algoritama mašinskog učenja. Selekcija instanci može da identifikuje i ukloni izuzetke, redundancije i nejasnoće u podacima, što dovodi do čistijeg i efikasnijeg skupa podataka za treniranje i testiranje algoritama, što na kraju rezultira boljim performansama modela.

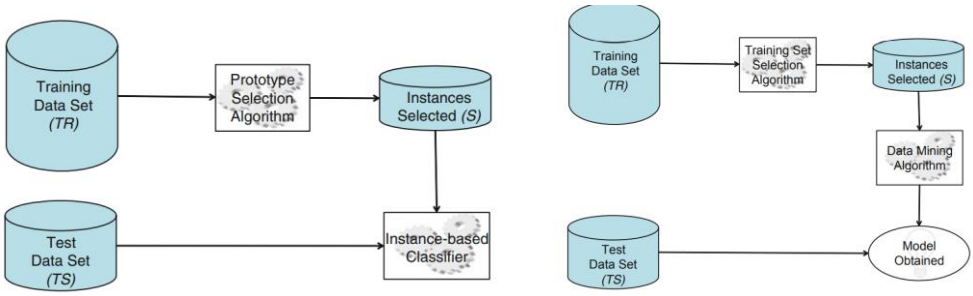
## Razlika između *Training set selection* i *Prototype selection*

U početku je nekoliko predloga za odabir najrelevantnijih podataka iz skupa za obuku predloženo razmišljajući uglavnom u KNN algoritmu. Kasnije, kada je termin učenje zasnovano na instanci (eng. Instance-based learning), takođe poznat kao lenjo učenje (eng. Lazy learning), smišljen za okupljanje svih onih metoda koje ne izvode fazu obuke tokom učenja, pojavljuje se termin za odabir prototipa (eng. Prototype selection). Danas porodica IS metoda takođe uključuje predlog za koji se smatralo da radi sa drugim metodama učenja, kao što su stabla odlučivanja, ANN ili SVM. Međutim, nije postojao način da se odredi konkretan slučaj u kojem je IS metoda validna i može se primeniti na bilo koju vrstu DM algoritma (unutar iste paradigme učenja, naravno). Iz tog razloga razlikujemo dve vrste procesa: odabir prototipa (Prototype selection - PS) i izbor skupa za obuku (Training set selection - TSS). Ove metode imaju za cilj da prilagode skupove podataka za različite algoritme, kako bi se poboljšala njihova efikasnost i tačnost, optimizujući resurse i poboljšavajući performanse.

PS metode su IS metode koje očekuju da pronađu skupove za treniranje koji pružaju najbolju tačnost klasifikacije i stopu smanjenja korišćenjem klasifikatora zasnovanih na instanci koji uzimaju u obzir određenu sličnost ili meru udaljenosti. Nedavno su PS metode postale popularnije u oblasti smanjenja podataka. Ova metoda se koristi za smanjivanje veličine skupa podataka u cilju ubrzavanja klasifikacije. Primeri koji se odabiraju kao prototipovi su oni koji se smatraju najreprezentativnijim za celokupni skup podataka. Bavi se odabirom određenog broja primera koji će predstavljati ostale primere u skupu podataka. Pored smanjenja vremena potrebnog za klasifikaciju, PS metode doprinose i boljem razumevanju podataka, olakšavajući analizu i interpretaciju rezultata, što je posebno korisno u oblastima poput medicinske dijagnostike, finansija i marketinga.

TSS metode su definisane na sličan način kao i PS metode. One su poznate kao primena IS metoda nad skupom za trening koji se koristi za izgradnju bilo kog prediktivnog modela. Dakle, TSS se može koristiti kao način da se poboljša ponašanje prediktivnih modela, preciznost, i interpretabilnost. Ova metoda se koristi kada postoji veliki broj primera u skupu podataka i želimo da smanjimo veličinu skupa kako bismo smanjili vreme potrebno za treniranje modela i smanjili mogućnost prenaučenja. TSS metode omogućavaju bolje iskorišćavanje resursa, pružajući efikasnije rešenje za primene mašinskog učenja u različitim industrijskim i istraživačkim domenima, kao što su prepoznavanje obrazaca, analiza slike, i detekcija anomalija. - *Izbor seta za obuku bavi se odabirom skupa primera za treniranje modela koji će biti korišćen u klasifikaciji*.

Činjenica je da se danas mnogo više istražuju PS metode. Okvirna procena predloga prijavljenih u specijalizovanoj literaturi koja se posebno razmatra za TSS može biti oko 10% od ukupnog broja tehnika. Iako PS monopolizuje skoro sve napore u IS, TSS trenutno pokazuje uzlazni trend.




Slika 3. PS proces vs TSS proces

# Prototype selection

Postojeći efikasni algoritmi klasifikacije uveliko troše vreme i prostor prilikom obrade velikih skupova podataka, a naročito tokova podataka.
Kako bi se smanjila veličina skupa podataka i vreme izvršavanja klasifikacije, a istovremeno zadržali visoko referentni obrasci efikasnih klasifikacionih doprinosa, postalo je istraživačko žarište u oblasti klasifikacije obrazaca. Stoga je predložena efikasna strategija obrade, koja se naziva selekcija prototipa, a koja predstavlja neophodno smanjenje originalnog skupa podataka na osnovu određenih tehnika [4].

Korišćenjem selekcije prototipa moguće je dobiti skup referentnih prototipova koji mogu odražavati distribucione i klasifikacione karakteristike originalnog skupa podataka. Strategijom selekcije prototipa moguće je postići brzo vreme izvršavanja klasifikacije smanjujući osetljivost na veličinu skupa podataka i šum, bez gubitka tačnosti klasifikacije. Na taj način, problem neprihvatljive potrošnje vremena i prostora delimično je rešen. Međutim, algoritmi selekcije prototipa sami po sebi imaju mnoge nedostatke, kao što su osetljivost na redosled čitanja obrazaca, outlier-e i šum. Stoga, kako bi se prevazišla osetljivost na redosled čitanja obrazaca u algoritmu Kondenzovanog Najbližeg Suseda (CNN) i postigli referentni prototipovi koji su bliže granici klasifikacije na efikasniji način.

## Smer pretrage (Direction of search)

Selekcija instanci može se shvatiti kao problem traženja, dat je određeni set podataka, cilj je da se pronađe najreprezentativniji podskup primera za taj set. Može se definisati pet smerova u nastojanju za dobijanje referentnog skupa podataka:

- Dodavanje (eng. **Incremental/Forward selection**) - Inkrementalna pretraga počinje sa praznim podskupom S, i dodaje svaki primer iz seta podataka (TR) u S ako ispunjava neke kriterijume. U ovom slučaju, algoritam zavisi od redosleda prikazivanja i ovaj faktor može biti veoma važan. Trebalo bi da bude slučajan, ali postoje specijalni slučajevi. Jedna prednost inkrementalnog načina rada je da ako su podaci kasnije dostupni, nakon što je obuka završena, oni se mogu i dalje dodavati u S prema istim kriterijumima. Ova funkcionalnost može biti veoma korisna kod rada sa tokovima podataka ili onlajn učenjem. Još jedna prednost je

što su brzi i koriste manje skladišnog prostora tokom faze učenja nego neinkrementalni algoritmi. Glavni mana je da inkrementalni algoritmi moraju da donesu odluku na osnovu malo informacija i stoga su skloni greškama dok nije dostupno više informacija.

- Uklanjanje (eng. **Decremental/Backward selection**) - počinje sa S = TR i zatim traži primere za uklanjanje iz S. Opet, redosled prikazivanja je važan, ali suprotno od inkrementalnog procesa, svi trening primeri su dostupni za pregled bilo kada.
  Jedna mana uklanjanja jeste da vremenski skuplju operaciju od inkrementalnih algoritama. Osim toga, faza učenja mora da se izvede van mreže, jer uklanjanje zahteva sve moguće podatke. Međutim, ako primena dekrementalnog algoritma može da dovede do većeg smanjenja memorije, onda dodatno izračunavanje tokom učenja (koje se radi samo jednom) može pomoći u smanjenju vremena izračunavanja.

- **Batch** - Još jedan način primene PS procesa je u batch modu. Ovo podrazumeva odlučivanje da li svaki primer zadovoljava kriterijum za uklanjanje pre nego što se bilo koji od njih ukloni. Zatim se svi oni koji zadovoljavaju kriterijum istovremeno uklanjaju. Kao i kod dekrementalnih algoritama, batch obrada pati od povećane vremenske složenosti u odnosu na inkrementalni algoritam.

- Mešoviti (eng. **Mixed**) - početak mešane pretrage se obavlja sa unapred odabranim skupom S (slučajno ili odabranim inkrementnim ili dekrementnim procesom) i iterativno može dodati ili ukloniti bilo koju instancu koja zadovoljava specifičan kriterijum. Ovaj tip pretraga omogućava ispravke već izvršenih operacija i njegova glavna prednost je lakše dobijanje skupa instanci sa dobrom preciznošću. Obično pati od istih nedostataka koji su prijavljeni kod dekrementnih algoritama, ali ovo u velikoj meri zavisi od specifičnih predloga.

- **Fixed -** je podgrupa mešane pretrage u kojoj broj dodavanja i uklanjanja ostaje isti. Dakle, broj konačnih prototipova određuje se na početku faze učenja i nikada se ne menja.

## Tipovi selekcije

Ovaj faktor je uglavnom uslovljen vrstom pretrage koju sprovode PS algoritmi, bilo da nastoje da zadrže granične tačke, centralne tačke ili neki drugi skup tačaka:

- Kondenzacija (eng. **Condensation**) - ovaj skup uključuje tehnike koje imaju za cilj da zadrže tačke koje su bliže granicama odluke, koje se takođe nazivaju granične tačke. Intuicija koja stoji iza zadržavanja graničnih tačaka je da unutrašnje tačke ne utiču toliko na granice odlučivanja kao granične tačke, i stoga se mogu ukloniti sa relativno malim efektom na klasifikaciju. Ideja je da se sačuva tačnost nad skupom za obuku, ali na tačnost generalizacije nad skupom za testiranje može negativno uticati. Ipak, sposobnost redukcije metoda kondenzacije je obično visoka zbog činjenice da u većini podataka ima manje graničnih tačaka nego unutrašnjih tačaka.

- Izdanje (eng. **Edition**) – ove vrste algoritama umesto toga nastoje da uklone granične tačke uklanjaju tačke koje su „bučne“ ili se ne slažu sa susedima. Ovo uklanja granične tačke, ostavljajući glatke granice odluke iza sebe. Međutim, takvi algoritmi ne uklanjaju unutrašnje tačke koje ne doprinose nužno granicama odluke. Dobijeni efekat se odnosi na poboljšanje tačnosti generalizacije u podacima testa, iako je dobijena stopa redukcije niska.

- Hibrid (eng. **Hybrid**) – ove metode pokušavaju da pronađu najmanji podskup S koji održava ili čak povećava tačnost generalizacije u podacima testa. Da bi se ovo postiglo, omogućava uklanjanje unutrašnjih i graničnih tačaka na osnovu kriterijuma koje slede dve prethodne strategije. KNN klasifikator je veoma prilagodljiv ovim metodama, postižući velika poboljšanja čak i sa veoma malim podskupom odabranih instanci.

## Evaluacija pretrage

KNN je jednostavna tehnika i može se koristiti za usmeravanje pretrage PS algoritma. Cilj kojem se teži je da se napravi predviđanje o nedefinitivnoj selekciji i da se uporede izbori. Ova karakteristika utiče na kriterijum kvaliteta i može se podeliti na:

- Filter (eng. **Filter**) - Kada se pravilo kNN koristi za parcijalne podatke kako bi se odredili kriterijumi dodavanja ili uklanjanja, a nije korišćena *leave-one-out* validaciona šema kako bi se dobila dobra procena generalizacijske tačnosti. Činjenica da se koriste podskupovi trening podataka pri svakoj odluci povecava efikasnost ovih metoda, ali tačnost možda neće biti poboljšana. - *Odluka se donosi korišćenjem neke heuristike ili pravila i nije zasnovana na klasifikatoru.*
- Omotač (eng. **Wrapper**) - Kada se pravilo kNN koristi za celokupni skup podataka za obuku uz primenu *leave-one-out* validacione procedure. Kombinacija ova dva faktora nam omogućava da dobijemo odličnu procenu opšte tačnosti, što pomaže da dobijemo bolju tačnost na test podacima. Međutim, svaka odluka uključuje kompletnu računsku obradu pravila kNN nad skupom za obuku, a faza učenja može biti računski skupa.. - *Odluku o odabiru ili brisanju instance donosi klasifikator*

## Kriterijumi za poređenje

U ovom delu će biti opisani osnovni kriterijumi po kojima se porede algoritmi selekcije instanci. Svaki od kriterijuma je bitan u konačnom zbiru i svaki set podataka ima neki kriterijum koji mu je najznačajniji kako bi rezultati obrade bili što bolji. Osnovni krijetijumi se mogu podeliti u četiri grupe:

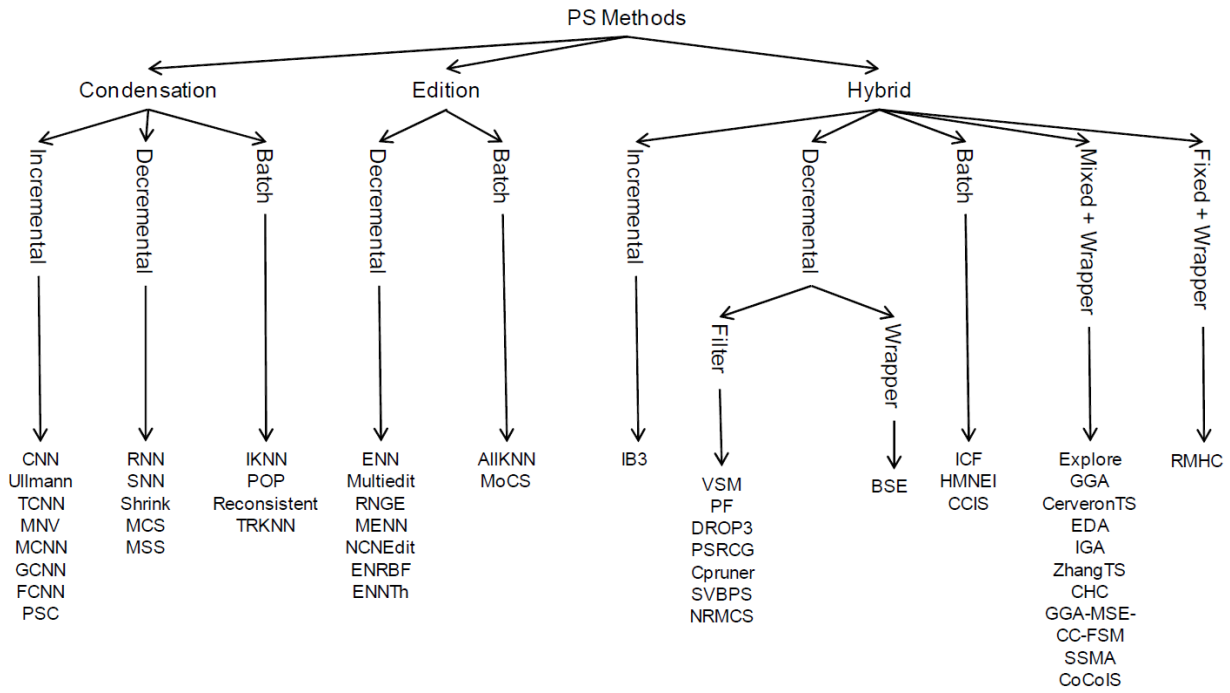
- Smanjenje zauzeća memorijskog prostora (eng. **Storage reduction**) - jedan od glavnih ciljeva PS metoda je smanjenje zahteva za memorijom. Štaviše, još jedan cilj koji je usko povezan sa ovim je da se ubrza klasifikacija. Smanjenje broja sačuvanih instanci obično će dovesti do odgovarajućeg smanjenja vremena potrebnog za pretragu ovih primera i klasifikaciju novog ulaznog vektora.

- Tolerancija buke (eng. **Noise tolerance**) - dva glavna problema mogu se pojaviti u prisustvu buke. Prvi je da će vrlo malo instanci biti uklonjeno jer je potrebno mnogo instanci da bi se održale bučne granice odlučivanja. Drugo, tačnost generalizacije može da trpi, posebno ako se zadrže bučne instance umesto dobrih instanci.

- Tačnost generalizacije (eng. **Generalization accuracy**) – uspešan algoritam često može značajno smanjiti veličinu skupa za obučavanje bez značajnog smanjenja opste tačnosti generalizacije.

- Vremenska zahtevnost (eng. **Time requirements**) - obično se proces učenja obavlja samo jednom na setu za obuku, tako da izgleda da to nije veoma važan metod evaluacije. Međutim, ako faza učenja traje predugo, može postati nepraktična za stvarne primene.

# Prototype selection algoritmi




*Slika 4. Prototype selection algoritmi*

Kao sto se može primetiti na slici 4., postoji veliki broj algoritama koji sluze za odabir manjeg skupa podataka koji će biti reprezentativan za veći skup koji predstavlja. Algoritmi su podeljeni u grupe po svojim osobinama i načinu rada, svaki od njih ima svoje prednosti i mane gledano u odnosu na ostale. Algoritmi su podeljeni u tri osnovne grupe: Condensation, Edition i Hybrid. U nastvaku će gradacijski biti opisani najbitniji predstavnici grupa i objašnjen sam rad algoritama, njihove prednosti i mane, kao i specifičnosti za pojedine predstavnike.

## Condensed nearest neighbor(CNN)

Istorijski gledano, od nastanka same potrebe za redukcijom dimenzionalnosti podataka, prvo rešenje koje se pojavilo kada je u pitanju selekcija instanci jeste Condensed Nearest Neighbor(CNN), davne 1968. godine. Danas, iako je prošlo vise od pedeset godina od nastanka ovog algoritma, isti i dalje predstavlja polaznu osnovu za rešenje problema i mnogi savremeni i unapređeni algoritmi su nastali kao nadogradnja na postojeću ideju. U skladu sa tim, za početak priče o algoritmima za redukciju dimenzionalnosti će biti opisan rad CNN algoritma [5].

U osnovi, CNN je undersampling tehnika koja bira podskup podataka iz većeg skupa podataka koji ne dovodi do smanjenja performansi modela, a istovremeno bira minimalan skup instanci podataka. Ovo se postiže dodavanjem instanci podataka u

skup ako i samo ako se ne mogu pravilno klasifikovati već postojećim sadržajem skupa podataka, sam algoritam je nastao kao rešenje problema zahteva za memorijom algoritma k-najbližih suseda(KNN).
Sam algoritam je implementiran u *scikit* biblioteci *imbalanced-learn* i dostupan je na korišćenje nakon instaliranja ove biblioteke. Tokom izvršavanja algoritma koristi se KNN algoritam za klasifikaciju tačaka kako bi se odredilo da li će instanca biti dodata u podskup ili ne. Algoritam se izvršava veoma sporo i preporučeno je korišćenje manjih skupova podataka [6].

U nastavku se nalazi pseudo kod:

```
def condensed_k_nearest_neighbors(train):
    samples = set()

    # Choose a random starting observation
    random_sample = train.pop(random.randint(0, len(train) - 1))
    samples.add(random_sample)

    additions = True
    while additions:
        additions = False
        for _ in range(len(train)):
            x = random.choice(train)

            min_distance = float('inf')
            closest_sample = None
            for sample in samples:
                distance = x.calcDistance(sample)
                if distance < min_distance:
                    min_distance = distance
                    closest_sample = sample

            if x.classifier != closest_sample.classifier:
                samples.add(x)
                train.remove(x)
                additions = True

    print("Number of samples selected: " + str(len(samples)))
    return samples
```

*Slika 5. CNN algoritam*

Algoritam je iterativan, uzevši u obzir da radi sa velikim količinama podataka, jasno je da je tvrdnja da je veoma spor ispravna.

Ako pažljivije pogledamo metod, možemo zaključiti da nemamo osiguranje da izlazni podskup neće biti u potpunosti isti kao i ulazni skup podataka, to se može zavesti kao jedan od nedostataka ovog algoritma, međutim u praksi je drugačije i retki su slučajevi u kojima se ovo može desiti, dok se sanse za ovakav ishod direktno proporcionalno smanjuju sa rastom broja instanci u datom ulaznom skup podataka, uzimajući u obzir da će algoritam biti korišćen za bas takve skupove podataka zanemarljiva je ova činjenica, ali treba imati na umu tu mogućnost.

Na samom početku algoritma može se uočiti prazan početni skup *samples*, kroz iteracije ovaj skup se proširuje, sto nam govori o osnovnoj osobini i opredeljenosti ovog algoritma, a to je da pripada grupi inkrementalnih algoritama za redukciju dimenzionalnosti.

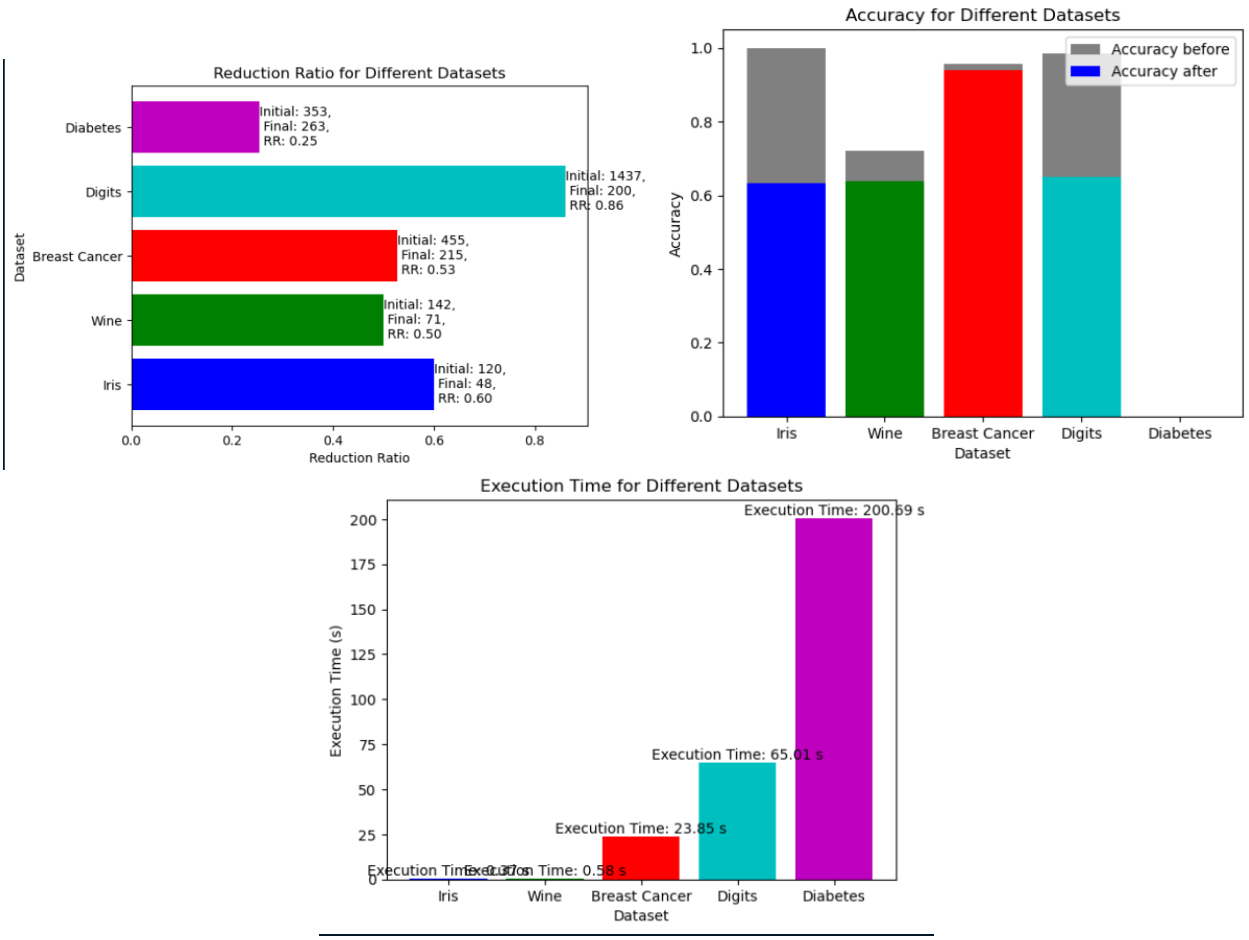
Ovaj metod takođe dokazuje i svoju osobinu *condensed* na taj način sto bira samo granične tačke, tj. instance trening skupa podataka koje određuju njegovu specifičnost, međutim ovo može biti okidač za smanjenje tačnosti predviđanja novih test podataka, zato sto se mogu pojaviti nove granične tačke koje pripadaju nekoj klasi, međutim neće biti približne odabranim graničnim tačkama.

Algoritam je takođe zavistan od raspoređenosti instanci u datasetu, neće za sve rasporede instanci davati isti rezultujući podskup podataka, sto nas navodi na razmišljanje, da li je izlazni set podataka ustvari i najreprezentativniji podskup ili postoji neki podskup, sto je vrlo verovatno, koji bolje reprezentuje ulazni skup.

U slučajevima kada se koristi za balansiranje setova podataka, ovaj konačni podskup podataka se sastoji od svih instanci koje pripadaju manjinskoj klasi i samo primeri iz dominantne klase koji se ne mogu pravilno klasifikovati se postepeno dodaju u podskup.

Kompleksnost Condensed Nearest Neighbor algoritma zavisi od nekoliko faktora, kao što su broj instanci u trening skupu podataka, broj karakteristika (atributa) i broj klasa. U najgorem slučaju, kompleksnost algoritma je O(n^3), gde je n broj instanci u trening skupu podataka.
Ovaj slučaj se javlja kada algoritam mora da prođe kroz sve instance više puta kako bi izvršio redukciju. Međutim, u praksi, CNN algoritam često može da postigne zadovoljavajuće rezultate sa manje iteracija i time smanji vreme izvršenja. U stvarnim scenarijima, kompleksnost može biti znatno manja od gornje granice.

Važno je napomenuti da je ovaj algoritam pogodan za inkrementalno učenje i može se poboljšati kroz razne optimizacije, kao što su korišćenje efikasnijih struktura podataka ili ubrzavanje procesa računanja udaljenosti.




*Slika 6. Rezultati primene CNN algoritma na različite setove podataka*

Na slici 6. prikazani su rezultati izvršavanja CNN algoritma, prvo je prikazana efikasnost algoritma kada je u pitanju redukcija, zatim tačnost klasifikacije pre redukcije i nakon redukcije i na kraju vreme koje je bilo potrebno za izvršenje redukcije.
Zapažanja su sledeća:

- Najveće smanjenje podataka se dogodilo kada je u pitanju data set „digits“, međutim, kada uzmemo u obzir tačnost klasifikacije i vreme izvršenja, sa smanjenjem broja instanci od 80% smanjena je i tačnost izračunavanja za otprilike 25% sto je ogroman gubitak u konačnom skoru i može se zaključiti da ovaj algoritam nije bio bas efikasan kada je u pitanju ovaj skup podataka, iako je smanjenje bilo ogromno, slično se desilo i sa skupom podataka „Iris“, uzevši u obzir količinu podataka.

- Set podataka koji se odnosi na dijabetes apsolutno nije dobro odreagovao na smanjenje količine podataka, međutim može se reci da je u ovom slučaju sam algoritam najbližih suseda praktično neprimenjiv za ovakav skup podataka, tako da bi se rezultati u svakom slučaju trebali zanemariti u konačnom zapažanju.

- Skupovi podataka koji su dali najbolje rezultate jesu „Wine“ i „Breast Cancer Data set“, smanjenje podataka za oko 50%, sto je odličan rezultat i tačnost klasifikacije koja je skoro pa zanemarljivo smanjena su pokazatelji da je u ovom slučaju CNN algoritam u velikoj meri doprineo redukciji dimenzionalnosti, a da je postignut cilj održavanja tačnosti klasifikacije.

U daljem radu će sva ispitivanja biti izvršena nad datim setovima podataka kako bi se stvorila realna slika i poređenje različitih pristupa nad istim podacima.

Opis rada algoritma:

Ulazni parametar metode jeste trening skup(*train*) podataka koji treba da se redukuje.
*samples* na kraju treba da predstavlja reprezentativni, smanjen podskup trening skupa podataka.
Početnom skupu se na početku dodaje slučajni element iz trening skupa podataka.
Promenljiva *additions* sluzi za izlazak iz beskonačne *while* petlje nakon sto se prekine dodavanje elemenata u podskup.

Nakon dodavanja početnog elementa u traženi podskup, počinje beskonačna petlja koja je aktivna sve do trenutka kada se prestaje sa dodavanjem novih elemenata u podskup.
Za svaku instancu iliti element ulaznog, već redukovanog skupa podataka, se bira slučajan element ulaznog skupa, zatim se vrši proračun i trazi najmanja udaljenost između slučajnog elementa i elemenata već dodatih u podskup samples, kada se pronađe najbliži element, proverava se da li je pronađeni element u istoj klasi sa slučajnim elementom, ako jeste to znaci da nema potrebe dodati ga u podskup, ako nije dodaje se u podskup jer predstavlja novitet i graničnu vrednost.

Nedostaci ovog algoritma:

1. Izabrani skup podataka je u velikoj meri nasumičan
2. Vremenski zahtevan
3. Zahtevan u pogledu memorijskih resursa
4. Postoji mogućnost preterane adaptacije podskupa na trening skup
5. Ne radi dobro za podatke koji sadrže *noise* ili greške
6. Može smanjiti interpretabilnost podataka
7. Ne garantuje smanjenje broja instanci
8. Ne izbacuje duplikate iz trening podataka

Prednosti CNN algoritma:
1. Smanjenje količine podataka
2. Sprečava overfitting

## Unapređenje CNN - Tomek Condensed Nearest Neighbor(TCNN)

„Two modifications of CNN“ – Ivan Tomek [7]

Ovaj algoritam je nastao kao nadogradnja CNN algoritma iz razloga sto se navedeni algoritam pokazao nepouzdanim, pre svega zbog slučajne selekcije instanci koje će biti sačuvane u podskupu trening skupa podataka, dok s druge strane postoji mogućnost odbacivanja bitnih instanci za dalju klasifikaciju. Ovaj algoritam je implementiran od strane Ivana Tomeka i sadrži dve osnovne modifikacije u odnosu na CNN algoritam, modifikacije se zasnivaju na intuitivnim aproksimacijama pojma granične tačke.

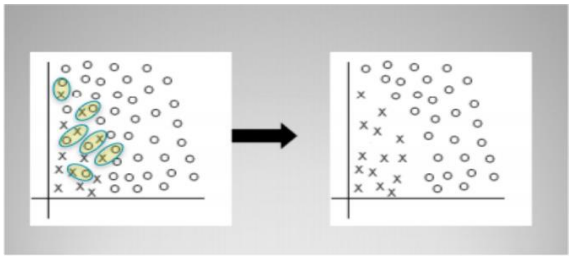
1. Kada je instanca x suprotne klase od sebi najbliže instance y (slučaj kada dodajemo element x u podskup samples), umesto da se ta instanca doda u podskup, metod pronalazi instancu koja je suprotne klase od klase instance x, koja je najbliži sused instanci y (y i instanca koja se trazi pripadaju istoj klasi) i dodaje je u podskup samples.
Ovim se postiže izbacivanje graničnih instanci koje mogu dovesti do nejasnoća pri klasifikovanju. Intuicija je da su ove instance blizu granice odlučivanja i stoga su važne za klasifikaciju.

2. Odabir instanci koje indirektno pomazu u klasifikaciji drugih instanci. Ova modifikacija ima za cilj da otkrije instance koje su neophodne za ispravnu klasifikaciju, ali možda nisu direktno uključene u granicu odlučivanja. Na primer, pretpostavimo da u određenoj fazi generisanja redukovanog skupa podataka E postoji instanca z koja se klasifikuje netačno jer nema tačaka iz neophodnog skupa A u E. U tom slučaju, pronađemo najbližeg suseda z u E, koji se zove u. Pošto se z klasifikuje netačno i jeste najbliži sused u, u mora pripadati suprotnoj klasi. Sada pronađemo najbližeg suseda u, koji se zove v, koji klasifikuje z ispravno. Instanca v pripada skupu A, koji je neophodan za tačnu klasifikaciju.

Iako ove modifikacije pomažu da se generiše redukovani skup podataka E koji

1. klasifikuje originalni skup podataka D ispravno
2. sadrži samo granice odlučivanja,

i dalje se ne garantuje da su sve poželjne tačke granice uključene. Ove modifikacije mogu se posmatrati kao pokušaj poboljšanja performansi CNN algoritma fokusiranjem na instance koje definišu granicu odlučivanja, slično konceptu Tomek Linkova.

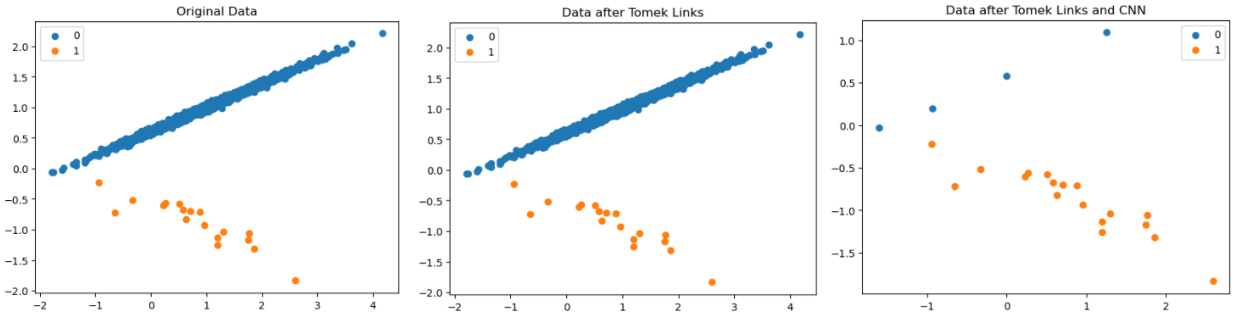


*Slika 7. Način rada tomek Links algoritma*

Na primer, Tomekovi linkovi su parovi instanci suprotnih klasa koji su najbliži susedi jedni drugima. Drugim rečima, to su parovi suprotnih instanci koji su vrlo blizu jedni drugima. Tomekov algoritam traži takve parove i uklanja instancu većinske klase iz tog para. Ideja je da se razjasni granica između manjinske i većinske klase, čineći region(e) manjinske klase jasnijim.

Instance koje se nalaze u Tomek linkovima su ili granične instance ili noisy instance. To je zato što će samo granične instance i noisy instance imati najbliže susede koji pripadaju suprotnoj klasi.

Izbor kombinovanja Tomek linkova i CNN-a je prirodan, jer se može reći da Tomek linkovi uklanjaju granične i *noisy* instance, dok CNN uklanja redundantne instance.




*Slika 8. Originalni set podataka, nakon primene Tomek linkova i nakon kombinacije Tomek linkova i CNN*

## Edited Nearest Neighbour (ENN)

Algoritam uređivanja najbližih suseda (eng. Edited Nearest Neighbour - ENN) predstavljen je 1972. godine od strane D. L. Wilsona. Wilsonov rad se bavio ograničenjima algoritama Najbližeg suseda (eng. Nearest Neighbor - NN) i k-Najbližih suseda (eng. k - Nearest Neighbour k-NN) u vezi sa klasifikacijom *noisy* podataka.

Wilson je predložio metodu za uređivanje trening skupa podataka, uklanjanjem instanci koje su verovatno pogrešno klasifikovane. ENN je poboljšao performanse klasifikacije, smanjujući uticaj šuma, autsajdera ili pogrešno označenih instanci, čime su NN i k-NN klasifikatori postali robustniji i precizniji.

Od tada je ENN proučavan, proširen i prilagođen za razne primene u mašinskom učenju i „rudarenju“ podataka. Istraživači su predložili modifikacije kao što su varijacije metrika udaljenosti, različite strategije za odabir k i alternativne tehnike uređivanja. Razvoj ENN-a doveo je do istraživanja drugih tehnika za smanjenje šuma i metoda izbora instanci za poboljšanje performansi klasifikatora zasnovanih na NN[5].

Navedeni algoritam po svojim preferencama u potpunosti predstavlja suprotnost ranije opisanom CNN algoritmu. Pre svega radi se o algoritmu iz *Edition* grupe algoritama koji je dekrementalan, dakle, ovaj algoritam nastoji da od početnog trening skupa podataka

stvori novi, redukovan skup podataka, izbacivanjem instanci koje mogu dovesti do greške u klasifikovanju test podataka.

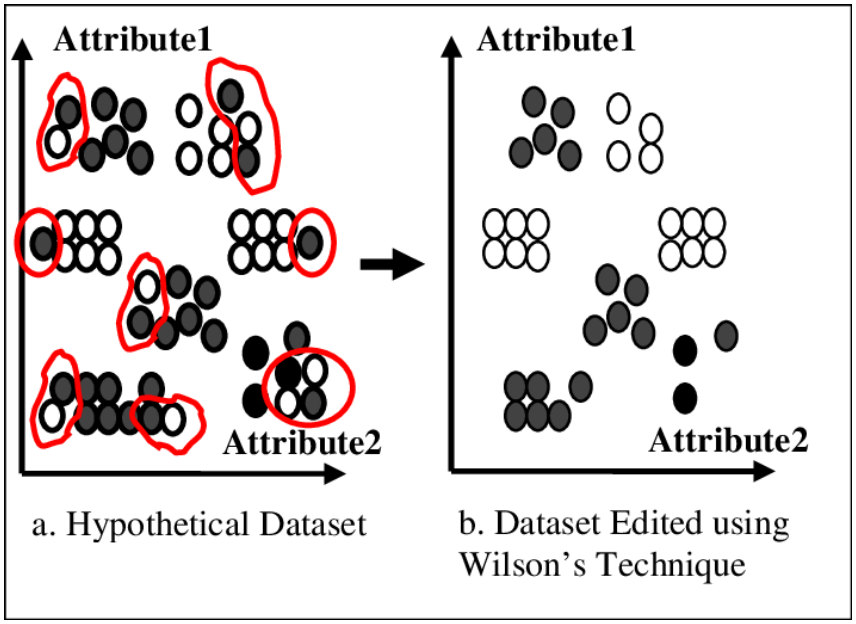
Ovaj algoritam se u programskom jeziku *Python* može implementirati kroz biblioteku *imbalanced-learn* kao *EditedNearestNeighbours.*

ENN metod radi tako što prvo pronalazi K-najbližih suseda za svaku instancu, a zatim proverava da li je većinska klasa iz K-najbližih suseda instance ista kao i klasa instance ili ne. Ako se većinska klasa K-najbližih suseda instance i klasa instance razlikuju, tada se instanca i njenih K-najbližih suseda brišu iz skupa podataka. Podrazumevano, broj najbližih suseda koji se koristi u ENN metodi je K=3.

Koraci u ENN algoritmu su sledeći:

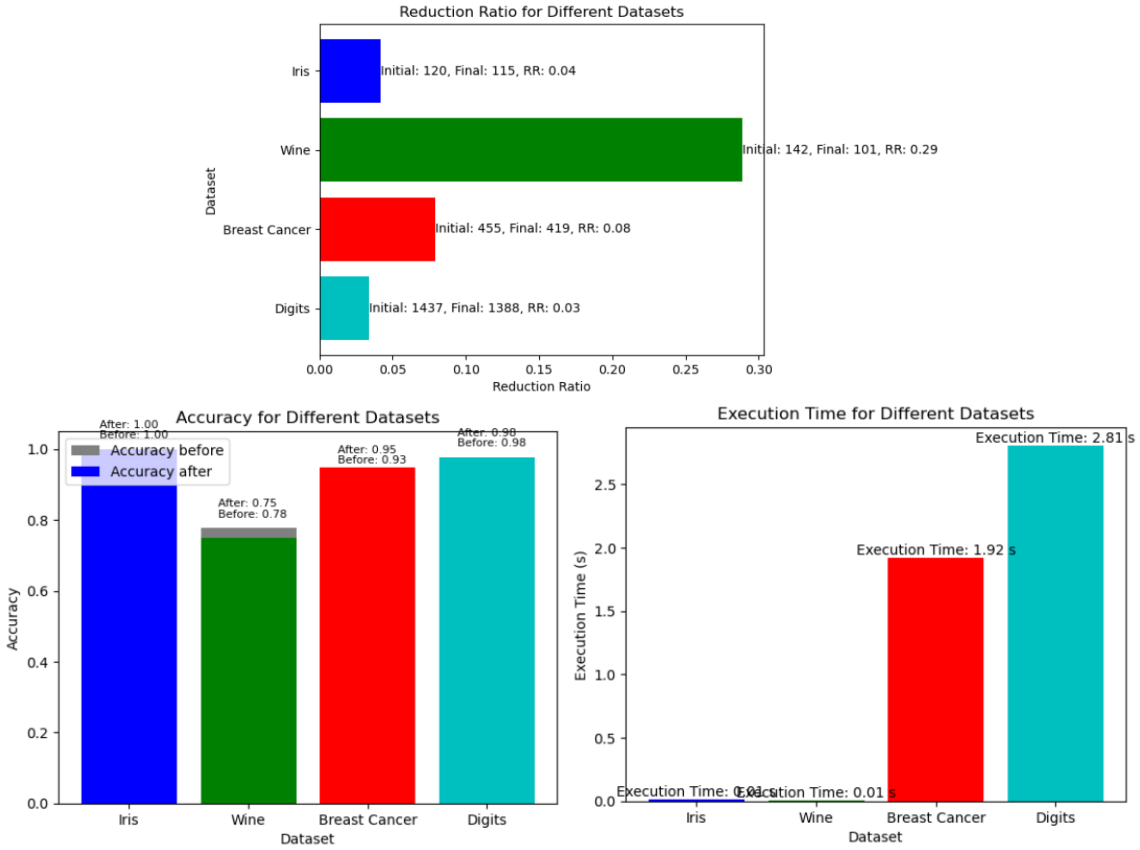
1. Dat je ulazni set trening podataka za N instanci, proverava se postavljena vrednost za K(broj najbližih suseda). Ako nije postavljeno K, uzima se difoltna vrednost 3.
2. Pronalazi se K najbližih suseda instance u setu podataka u odnosu na ostale instance.
3. Ako je instanca u klasi koja je različita od dominantne klase svoje okoline, instanca se izbacuje iz trening skupa podataka.
4. Ponavljati korake 2 i 3 dok se ne postigne željena posednutost klasa

Kompleksnost ovog algoritma zavisi od dva faktora, a to su pronalaženje najbližih suseda i uklanjanje instanci koje ne zadovoljavaju uslov. Prvi faktor ima složenost $O(n^2)$ dok drugi faktor ima složenost O(n*k) sto nas dovodi do zaključka da je složenost ovog algoritma O(n*(n+k)).




*Slika 9. Rezultati rada Wilsonovog algoritma editovanja najbližih suseda (ENN)* [8]

Na slici 9. se može videti rad Wilsonovog algoritma i rezultati koje daje. Lako se može zaključiti da ovaj algoritam efikasno sklanja *noisy* podatke i smanjuje količinu podataka u trening skupu podataka. Međutim, jasno je da ovakvih podataka nema mnogo i da se broj instanci neće smanjiti u svim slučajevima za veliki broj, cak naprotiv, uglavnom će broj instanci u setovima podataka ostati skoro pa nepromenjen. S druge strane efikasno su izbačeni podaci koji mogu dovesti do pogrešnih zaključaka, a uz kombinaciju sa nekim drugim algoritmom koji nema ove osobine, može se stvoriti idealno rešenje za izbor podskupa trening skupa podataka.




*Slika 10. Rezultati primene ENN algoritma na različite setove podataka*

Na slici 10. su prikazani rezultati primene ENN algoritma na ugrađene setove podataka. Za broj suseda u ENN algoritmu je odabrano 3 i vršena je klasifikacija sa jednom najbližim susedom k-NN algoritma.

Zapažanja su sledeća:

1. Set podataka koji sadrži podatke o vinu je najbolje odreagovao na ENN algoritam, procenat podataka koji su izbačeni iz trening skupa podataka je 30, sto govori o tome da je u trening skupu podataka bilo dosta „lutajućih“ instanci, međutim, smanjen je accuracy za 3%, može se zaključiti da je primena ovog algoritma bila uspešna kada su podaci o vinu bili u pitanju.

2. Specifičnost je i to da se Breast Cancer set podataka odlično adaptirao na ENN algoritam, sto je rezultovalo smanjenjem broja instanci za skoro 10% i povećanjem tačnosti klasifikacije za cak 3% sto je ogroman rast uzevši u obzir da se radi o tačnosti koja je blizu savršenoj.

3. Ostali skupovi podataka su pokazali raspoređenost i minimalnu *noise* u podacima, pa u skladu sa tim nije mnogo instanci podataka izbačeno iz početnog skupa, ali ovim setovima podataka tačnost obrade nije smanjena, tako da se i ovde može reci da je algoritam bio uspešan u određenoj meri, ali njegova primena nije bila potrebna ako se faktor vremena uzme u obzir.

4. Takođe se može primetiti da kod ovog algoritma veličina skupa podataka igra glavnu ulogu po pitanju vremena izvršenja, sto je i logično, jer je za svaku instancu potrebno pronaći najbliže susede. Ovo može biti zabrinjavajuće ako uzmemo u obzir da se lako može pojaviti set podataka sa brojem instanci reda stotina hiljada ili nekoliko miliona.

Ovaj algoritam je veoma moćno oružje kada se radi o „zalutalim“ podacima u zadacima klasifikacije. Jasno je da efekti nisu zadovoljavajući kada je u pitanju redukcija podataka, međutim, primenom na primer Breast Cancer seta podataka je dokazano da se ovaj algoritam može izuzetno dobro odraziti na same rezultate klasifikacije.

Prednosti:
1. Početni elementi nisu slučajno odabrani
2. Uklanja *noisy* podatke
3. Pozitivno utiče na tačnost klasifikacije

Nedostaci:
1. Neznatno smanjuje broj instanci
2. Sa povećanjem količine podataka, povecava se vreme potrebno za izvršenje algoritma

### SMOTE + ENN

Kao sto je ranije navedeno, ENN je efikasniji u kombinaciji sa drugim algoritmima. Jedan od takvih algoritama jeste SMOTE. Ova kombinacija ima za cilj smanjenje veličine skupa podataka, smanjenje preprilagođavanja (overfitting), poboljšanje klasifikacije manjinskih klasa i smanjenje šuma u podacima.

Šta je SMOTE?

SMOTE (eng. Synthetic Minority Over-sampling Technique) je algoritam za prevazilaženje problema neuravnoteženosti klasa u kojima je ciljna klasa manje

zastupljena. Ova tehnika koristi generisanje sintetičkih instanci ciljne klase na osnovu klasa manjka. SMOTE bira slučajne instance iz ciljne klase i zatim generiše nove instance interpoliranjem atributa izabrane instance i njenih k-nearest suseda. Ovo se ponavlja dok se ne dostigne željeni nivo balansiranosti klasa. SMOTE pomaže u povećanju performansi algoritama klasifikacije koji su osjetljivi na neuravnoteženost klasa.

Nekoliko ključnih karakteristika SMOTE algoritma su:

- Generiše nove sintetičke instance za ciljnu klasu, čime se poboljšava njen broj i omogućava bolje modelovanje.
- Izbor suseda se vrši na osnovu euklidske udaljenosti u prostoru atributa, tako da SMOTE funkcioniše samo u prostoru numeričkih atributa.
- Prilagođavanjem parametra k, korisnik može da kontroliše koliko novih sintetičkih instanci će biti generisano za ciljnu klasu.
- SMOTE može biti primenjen samostalno ili u kombinaciji sa drugim metodama poput *undersampling* metoda.

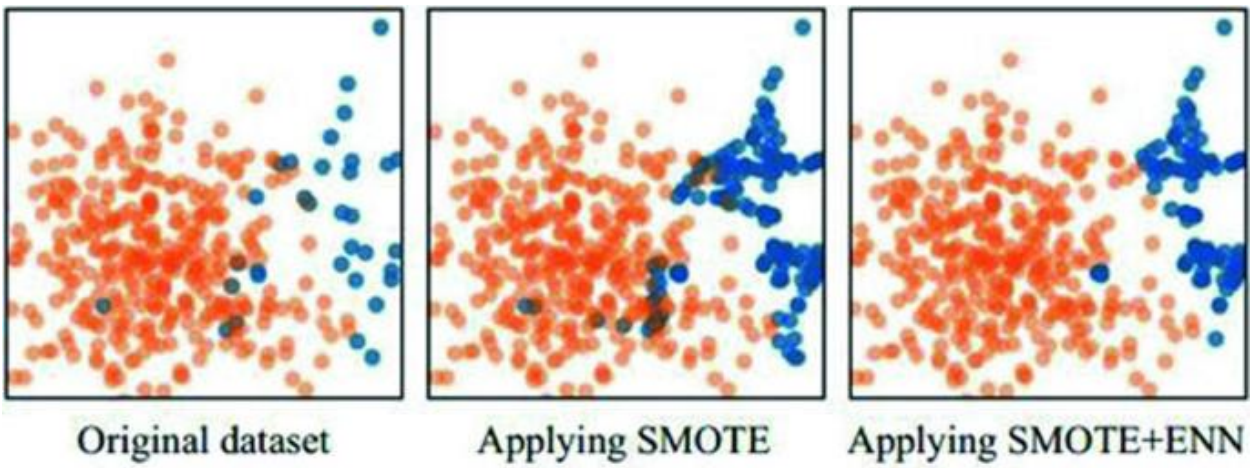
### *SMOTEENN metod*

Ovaj metod, razvijen od strane Gustava Batiste 2004. god., kombinuje sposobnost SMOTE algoritma da generiše sintetičke primere za manje zastupljenu klasu i sposobnost ENN algoritma da obriše neke instance iz obe klase za koje je uočeno da imaju suprotnu klasu u odnosu na klasu njihovih K-najbližih suseda koji pripadaju većinskoj klasi [9].
Ovaj algoritam se takođe može pronaći u *imblearn* biblioteci kao *SMOTEENN*.
Proces SMOTE-ENN algoritma se može objasniti na sledeći način:
Početak SMOTE-a

1. Izabrati slučajne podatke iz manje zastupljene klase.
2. Izračunati udaljenost između slučajnih podataka i njihovih K-najbližih suseda.
3. Pomnožiti razliku sa slučajnim brojem između 0 i 1, a zatim dodati rezultat manje zastupljenoj klasi kao sintetički uzorak.
4. Ponavljati korake 2 i 3 sve dok se ne postigne željeni udeo manje zastupljene klase. (Kraj SMOTE-a)

Početak ENN-a

5. Odrediti K kao broj najbližih suseda. Ako nije određen, onda K = 3.
6. Pronaći K-najbližih suseda instance među ostalim instancama u skupu podataka, a zatim vrati većinsku klasu od K-najbližih suseda.
7. Ako klasa instance i većinska klasa njenih K-najbližih suseda nisu iste, onda se instanca i njenih K-najbližih suseda brišu iz skupa podataka.
8. Ponavljati korake 2 i 3 sve dok se ne ispuni željeni udeo svake klase. (Kraj ENN-a)




*Slika 11. Primer rezultata korisćenja SMOTEENN algoritma*

## Decremental Reduction Optimization Procedure Family(DROP)

Grupa ovih algoritama predstavlja hibridne algoritme, ciji je cilj osim smanjenja broja instanci u setu podataka i povećanje tačnosti klasifikacije algoritama za klasifikaciju. Ova grupa algoritama pripada dekrementalnim algoritmima, sto znaci da početnu tačku algoritma čini trening skup podataka, novitet je to sto ovi algoritmi imaju filter osobinu, a to znaci da koriste pravili kNN za aproksimaciju pripadnosti neke instance određenoj klasi.

Da bismo predstavili ove tehnike redukcije, potrebno je definisati neke pojmove. Svaka instanca $X_i$ ima k najbližih suseda, gde je k obično mali neparan broj. $X_i$ takođe ima najbližeg neprijatelja, koji je najbliža instanca sa različitom izlaznom klasom. One instance koje imaju $x_i$ kao jednog od svojih k najbližih suseda nazivaju se pridruženim instancama $X_i$.

### DROP1

Koristi sledeće osnovno pravilo kako bi odlučio da li je bezbedno ukloniti instancu iz skupa instanci S(podskup početnog skupa), gde je S=TR (Ulazni trening skup podataka):
Ukloniti $X_i$ ako bi bar isti broj njegovih pridruženih instanci u S bio klasifikovan ispravno bez $X_i$.

Pseudo kod ovog algoritma:

```
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.
```

Pre svega se za svaku instancu u početnom trening skupu podataka trazi k+1(gde je k definisan broj traženih suseda) njenih najbližih suseda i svakom njenom susedu se u listu povezanosti dodaje ta instanca.
Zatim se ponovo prolazi ceo skup i broji koliko instanci iz grupe povezanosti trenutne instance može pravilno klasifikovati u slučaju da se trenutna instanca ne nalazi u grupi povezanosti. Takođe se broji i koliko njih ne.
Proverava se da li je broj instanci iz grupe povezanosti koje se mogu klasifikovati bez trenutne instance veći od broja koje ne mogu. Ako je broj veći u tom slučaju se trenutna instanca $X_i$ izbacuje iz početnog skupa podataka, takođe se $X_i$ izbacuje iz svih grupa povezanosti za svaku instancu u njegovoj grupi povezanosti, dok se svakoj instanci iz grupe povezanosti pronalaze novi najbliži susedi i dodaju u listu. Takođe je potrebno izbaciti $X_i$ iz komšijskih grupa povezanosti.
Iz ovoga se može zaključiti da je ovaj algoritam opredeljen dekrementalno i da ima osobine iterativnog algoritma, sto nije tako pogodno za obradu kod velikih skupova podataka.

Kompleksnost ovog algoritma zavisi od nekoliko faktora, kao sto su broj instanci n, broj najbližih suseda k i složenost traženja najbližih suseda. Ukupna kompleksnost je uzevši u obzir ove faktore i kompleksnost poziva petlji približno O(n*(3k+2))~O(n*k).

## DROP2

DROP1 je samo jedan od algoritama iz DROP familije, tako da je on osnovni u ovoj grupi, njegovo unapređenje može se pronaći u DROP2 algoritmu:

U ovoj metodi izmenjen je pre svega kriterijum za uklanjanje:
Ukloniti $X_i$ ako bi bar isti broj njegovih pridruženih instanci u TR bio klasifikovan ispravno bez $X_i$.

Koristeći ovu modifikaciju, svaka instanca $X_i$ u originalnom trening skupu TR nastavlja da održava listu svojih k + 1 najbližih suseda u S, čak i nakon što je $X_i$ uklonjen iz S. To znači da instance u S imaju asocijacije koje su i unutar i izvan S, dok instance koje su uklonjene iz S nemaju asocijacije.
DROP2 takođe menja redosled uklanjanja instanci. Inicijalno sortira instance u S prema udaljenosti do njihovog najbližeg protivnika. Provera za uklanjanje instanci započinje od instance najudaljenije od svog najbližeg protivnika.

Izmene u kodu su prikazane u nastavku.

Linija dodata pre dela koda za izbacivanje instanci:

```
# Sort instances in S by distance to their nearest enemy
S = sorted(S, key=lambda x: x.distance_to_nearest_enemy, reverse=True)
```

Izmenjeni deo koda za izbacivanje instanci po pravilima DROP2:

```
If without ≥ with:
    Remove Xi from S.
Else:
    For each associate a of Xi:
        If a not in S:
            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:
        If b not in S:
            Remove Xi from b's lists of associates.
```

Složenost ovog algoritma je: O(n*(3k+log(n)+1))~O(n*k) za k>>log(n) ili O(n*log(n)) za k~log(n) - u aproksimaciji je uzet najdominantniji član.

## DROP3

Ovaj algoritam je kombinacija DROP2 i ENN algoritma. Prvo se filtriraju *noisy* podaci, a zatim primenjuje ENN algoritam, nakon toga se nastavlja sa DROP2 algoritmom. Sto ovaj algoritam predstavlja kao nadogradnju prethodno već nadograđenog DROP2.

Uvedeno je dosta novih pojmova.

Za početak je algoritam nazvan **CPruner.**

1. Za instancu $X_i$ u TR skupu podataka se kreira skup podataka koji su "dosegabilni" $X_i$ (eng. k-reachability set), koji se označava sa k-reachability($X_i$).
2. Za instancu $X_i$ u TR kreira se skup podataka sa instancama slične klase kao $X_i$ i koje imaju $X_i$ kao jednog od svojih suseda. Ovaj skup se naziva skupom "pokrivaca" (eng. k-coverage set), koji se označava sa k-coverage($X_i$).
3. Za instancu $X_i$ u TR koja se može klasifikovati tacno pomoću skupa k-reachability kaže se da je podrazumevana kroz skup dosegabilnosti i ova instanca je suvisna (eng. superfluous) u skupu podataka.
4. Za instancu $X_i$ u TR skupu podataka se kaže da je "kritična", ako važe sledeći uslovi:
    - Bar jedna instanca *Xj* u k-coverage($X_i$) nije podrazumevana kroz k-reachability(*Xj*)
    - Nakon sto je Xi izbrisana, bar jedna instanca *Xj* u k-coverage($X_i$) nije podrazumevana kroz k-reachability(*Xj*)
5. Za instancu Xi u TR, ako *Xi* nije suvisna instanca i |k-reachability(*Xi*)| > |k-coverage(*Xi*)| tada je *Xi noisy* instanca.

Pored pojmova postoje i nova pravila izbacivanja instanci:

- **Pravilo(rule) 1.** "Odsecanje" instance (eng. instance pruning rule).
  Instanca Xi u TR se može odseci ako zadovoljava jedan od sledeća dva uslova:
    - Ona je *noisy* instance
    - Ona je suvisna (superfluous) instanca, ali nije kritična (critical)
- **Pravilo(rule) 2.** za određivanje redosleda uklanjanja instanci.
  Neka H-kNN(Xi) bude broj instanci njegove klase u kNN($X_i$), I D-NE($X_i$) bude udaljenost $X_i$ do njenog najbližeg neprijatelja.
  Za dve odsecive instance $X_i$ i $X_j$ u TR:
    - Ako je H-kNN(*Xi*) > H-kNN(*Xj*), *Xi* treba ukloniti pre *Xj*
    - Ako je H-kNN(*Xi*) = H-kNN(*Xj*) i D-NE(*Xi*) > D-NE(*Xj*), *Xj* treba ukloniti pre *Xi*
    - Ako je H-kNN(*Xi*) = H-kNN(*Xj*) i D-NE(*Xi*) = D-NE(*Xj*), redosled uklanjanja se odlučuje nasumično.

Pseudo kod DROP3 ili CPruned algoritma:

```
S=TR
For all Xi ϵ do
   Compute k-reachability (Xi) and k-coverage (Xi)
For all Xi ϵ S do
   If Xi is a noisy instance
      Remove Xi from S
      For all Xj ϵ k-coverage (Xi)
         Remove Xi from k-reachability (Xj)
         Update k-reachability(Xj)
      For all Xj ϵ k-reachability(Xi)
         Remove Xi from k-coverage (Xj)
Sort the order of instances in S according to rule 2
For all Xi ϵ S
   If Xi satisfies rule 1
      Remove Xi from S
      For all Xj ϵ k-coverage (Xi)
         Remove Xi from k-reachability(Xj)
         Update k-reachability (Xj)
Return S
```

Ovaj algoritam trenutno predstavlja najmoćniji algoritam iz porodice DROP i verodostojno opisuje sve karakteristike hibridnih algoritama. Ne samo da početni algoritam DROP1 predstavlja modifikaciju CNN algoritma, nego se taj algoritam dodatno nadograđuje u DROP2 verziji, dok se na kraju konačni oblik DROP algoritma povezuje sa ENN algoritmom i logikom i dobija Hibridni algoritam u pravom smislu te reci. Ovaj hibridni algoritam takođe ima osobine dekrementalnog i filter algoritma, sto doprinosi njegovoj složenosti ali u isto vreme i tačnosti i efikasnosti.

## Poređenje algoritama

| | CNN | ENN | DROP | Tomek-Links | SMOTEENN |
|---|---|---|---|---|---|
| **Smer pretrage** | Incremental | Decremental | Decremental | Decremental | Decremental |
| **Tip selekcije** | Condensed | Edition | Hybrid | Condensed | Hybrid |
| **Evaluacija pretrage** | / | / | Filter | / | / |
| **Redukcija dimenzionalnosti** | Velika | Mala | Srednja | Mala | Srednja |
| **Uklanja *noisy* instance** | Ne | Da | Da | Ne | Da |
| **Povecava/smanjuje/ne menja accuracy** | Ne menja/smanjuje | Povecava/smanjuje/ne menja | Povecava/ Ne menja | Povecava/ Ne menja | Povecava/ Ne menja |
| **Vremenski zahtevan** | Da | Ne | Ne | Ne | Ne |
| **Kompleksnost** | $O(n^3)$ | O(n*(k+n)) | O(n*k) | O(n) | O(n*k) |
| **Sprečava overfitting** | Da | Ne | Ne | Ne | Ne |
| **Garantuje redukciju** | Ne | Ne | Ne | Ne | Ne |
| **Izbacuje duplikate** | Ne | Ne | Ne | Ne | Ne |
| **Nasumičan rad** | Da | Ne | Ne | Ne | NE |

*Tabela 1. Osnovne karakteristike Prototype Selection algoritama*

# Praktični deo

U teorijskom delu opisan je način rada algoritama i njihove osobine, naravno, teoriju treba uvek uzimati sa rezervom, zbog toga će u nastavku praktično biti istražene date tvrdnje u vidu eksperimenata nad velikim skupovima podataka. Pre samog početka treba naglasiti da iz “tehničkih” razloga, a to je nedostatak implementacije nekih od obrađenih algoritama u teorijskom delu, neće biti isprobani svi navedeni algoritmi, obzirom na to da je glavna ideja primena algoritama koji su implementirani i dustupni u nekoj od biblioteka koje se koriste u mašinskom učenju, bilo bi neprofesionalno vršiti ručnu implementaciju algoritama i na osnovu njih iznositi zaključke, jer rezultati ne bi bili relevantni, a sami algoritmi nedostupni za korišćenje čitaocima ovog istraživanja.

## Osnovna ideja:

Primena algoritama za selekciju prototipa nad setovima podataka koji imaju veliki broj instanci i na koje do sada nije primenjivana nijedna metoda selekcije instanci. Pre svega će biti odrađena deskriptivna analiza podataka, zatim preprocesiranje, čišćenje podataka, kao i prikaz značenja podataka.
Glavni cilj jeste testiranje rada algoritama, zato će pre primene algoritama iz skupa podataka biti izbačen minimalan broj instanci i atributa, kako bi se videlo kakav efekat algoritmi imaju na anomalije u podacima. Biće proveren uticaj algoritama na *outlier-e,* zatim na duplikate u podacima, na količinu podataka, balansiranje podataka.. Na kraju će skupovi podataka biti dovedeni na minimalni oblik, uz pomoć dodatnih metoda biće pokusana maksimalna redukcija zauzeća memorijskog prostora, gde će akcenat biti na očuvanju tačnosti predviđanja novih, nepoznatih podataka. Nakon odabira najefikasnijeg metoda za selekciju instanci i maksimalne redukcije podataka, biće prikazani rezultati finalne redukcije dimenzionalnosti i čišćenja skupa podataka.

Setovi podataka koji su korišćeni u praktičnom delu:

- CSGO Round Winner (https://www.kaggle.com/datasets/christianlillelund/csgo-round-winner-classification)
- Citrus Fruit Classification(https://www.kaggle.com/code/gcdatkin/citrus-fruit-classification)

CSGO Round Winner Classification set podataka se sastoji od skupa podataka iz igre Counter Strike Global Offensive, skup je odabran nasumično, pre svega zbog velike količine instanci i velikog zauzeća memorije, a zatim i zbog zvučnosti imena kao i zanimljivosti same teme. Atributi su *snapshotovi* iz igre, to znaci da je u toku partija igre izvlačena trenutna statistika. Atributi se odnose na sve aspekte igre, od preostalog vremena, preko količine raspoloživog novca timovima do oružja koje timovi kupuju i da li su timovi postavili bombu u partiji. Svi ovi atributi bi trebali da daju informaciju o tome da li je partiju o kojoj se radi u *snapshotu* dobio tim terorista (T) ili specijalne jedinice (CT). Pobednika predstavlja atribut *round_winner*.

Karakteristike seta podataka CSGO:

- Broj kolona (fičera, atributa): 97
- Broj instanci: 122410
- Zauzeće memorije: 102.43 MB

Citrus Fruit Classification je dosta jednostavniji skup podataka, koristi se radi poređenja algoritama i pronalaženja eventualno najboljeg među njima u poređenju sa ostalim setovima podataka. U ovom skupu se na osnovu prečnika, tezine i boje određuje vrsta citrusnog voća, da li je u pitanju pomorandža ili grejpfrut. Jasno je da je ovaj set podataka pravljen za specifičnu namenu, a to je klasifikacija, pre svega zbog perfektnih brojki, a zatim i zbog perfektne balansiranosti podataka.

Karakteristike seta podataka Citrus:

- Broj kolona: 5
- Broj instanci: 10000
- Zauzeće memorije: 1.00 MB

## Algoritmi i funkcije

Pre same primene algoritama zapaženo je da 5% svih instanci CSGO seta podataka predstavlja duplikate, zatim, da postoje atributi koji imaju konstantne vrednosti, tj. atributi koji za svaku instancu imaju istu vrednost, sto govori o suvišnosti istih, tako da se na samom startu izbacuju konstantni atributi. Nakon izbacivanja ovih atributa ispostavilo se da 10% od ukupne količine instanci čine duplikati. Za početak duplikati će biti ostavljeni zbog provere efekta algoritama na duplikate. Set podataka je već smanjen za 10%.

Sledeći korak jeste implementacija funkcije za pozivanje nad različitim skupovima podataka i nad različitim algoritmima selekcije instanci.

```
from imblearn.under_sampling import (
    CondensedNearestNeighbour, NeighbourhoodCleaningRule,
    TomekLinks, EditedNearestNeighbours,
)
from imblearn.combine import SMOTEENN
```

Pored osnovnih importa koji su potrebni za implementaciju funkcije, potrebno je instalirati dodatnu biblioteku za *python* koja sadrži potrebne algoritme, a to je *imblearn,* ova biblioteka sadrži većinu dostupnih algoritama selekcije instanci, u radu su primenjeni gore navedeni algoritmi, SMOTEENN, kao hibridni algoritam predstavlja kombinaciju SMOTE i ENN algoritma i zbog toga je u grupi *combine.*

Funkcija prihvata parametre u vidu trening podataka i imena tehnike tj. algoritma koji treba da se izvrsi

```
def evaluate_under_sampling_technique(X, y, technique_name)
```

Ulazni podaci se prvo skaliraju, zatim dele na test i validacione setove podataka, nakon toga se primenjuje određeni algoritam nad njima, postoji uslov za proveru da li je u pitanju CNN algoritam, u njegovom slučaju se bira manji podskup podataka, zbog problema sa brzinom izvršavanja, taj algoritam je polazna osnova svih algoritama i on sluzi kao jedna od referentnih tačaka za poređenje, zbog toga nije i nikako ne može biti izostavljen.

```
# Apply under-sampling technique to the train set
if technique:
    # Limit the number of instances to 3000 for CNN
    if technique_name == "CNN" and X_train.shape[0] > 3000:
        X_train_resampled, y_train_resampled = technique.fit_resample(
                                X_train[:3000], y_train[:3000])
    else:
        X_train_resampled, y_train_resampled = technique.fit_resample(
                                X_train, y_train)
else:
    X_train_resampled, y_train_resampled = X_train, y_train
```

Poređenje algoritama se vrši po raznim parametrima, svi parametri za poređenje se vraćaju kao rezultat funkcije

```
results[technique_name] = {
        "time": elapsed_time,
        "accuracy": accuracy,
        "precision": report["weighted avg"]["precision"],
        "recall": report["weighted avg"]["recall"],
        "f1-score": report["weighted avg"]["f1-score"],
        "num_instances_start": num_instances_start,
        "num_instances_end": num_instances_end,
        "percentage_instances_end": percentage_instances_end,
        "num_duplicated_start": num_duplicated_start,
        "num_duplicated_end": num_duplicated_end,
    }
```

## Primena algoritama

U ovom delu biće prikazani i prokomentarisani rezultati primene algoritama za redukciju dimenzionalnosti.

| | algorithm | time | accuracy | percentage_instances_end | num_instances_start | num_instances_end | precision | f1-score | num_duplicated_start | num_duplicated_end |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Original | 31.907270 | 0.865983 | 100.000000 | 97928.0 | 97928.0 | 0.865989 | 0.865985 | 3739.0 | 3739.0 |
| 1 | ENN | 73.429134 | 0.807818 | 83.050813 | 97928.0 | 81330.0 | 0.830907 | 0.805023 | 3739.0 | 3315.0 |
| 2 | TomekLinks | 119.445626 | 0.864513 | 98.574463 | 97928.0 | 96532.0 | 0.864743 | 0.864525 | 3739.0 | 3739.0 |
| 3 | NCR | 123.184805 | 0.823176 | 85.470958 | 97928.0 | 83700.0 | 0.840756 | 0.821342 | 3739.0 | 3314.0 |
| 4 | SMOTEENN | 132.613221 | 0.796953 | 68.152112 | 97928.0 | 66740.0 | 0.797836 | 0.796929 | 3739.0 | 3052.0 |
| 5 | CNN | 421.006227 | 0.655216 | 72.833333 | 3000.0 | 2185.0 | 0.678859 | 0.645726 | 3739.0 | 5.0 |

*Slika 12. Rezultati primene algoritama na CSGO set podataka*

Na osnovu rezultata dobijenih primenom algoritama može se zaključiti sledeće:

- CNN algoritam, iako za redukovan skup podataka, zahteva značajno vise vremena nego ostali algoritmi, sto ga praktično dovodi u grupu algoritama koji se u praksi neće koristiti. Takodje tačnost predikcije je značajno smanjena, sto potvrđuje tvrdnje o slučajnom izboru istanči. Ne obraca pažnju na duplikate, ali u određenoj meri smanjuje postojanje *outlier*-a Ovaj algoritam predstavlja dobru polaznu osnovu, ali je definitivno kandidat za modifikacije, sto je i učinjeno u daljem razvoju.

- TomekLinks kao nadogradnja CNN algoritma zapravo ne smanjuje količinu podataka značajno, dok oduzima određeno vreme i tačnost čija cena ne može biti nadoknađena efikasnošću izvršenja, tako da ovaj algoritam uopšte ne mora biti primenjen, jer nema uticaja na smanjenje podataka.

- ENN algoritam se izuzetno dobro pokazao u skoro svim parametrima, vreme izvršavanja je zanemarljivo ako ga uporedimo sa obradom podataka bez primene algoritma, tačnost od 80% uz smanjenje broja instanci za 17% ga svrstava u red glavnih kandidata za redukciju dimenzionalnosti, međutim takodje je nakon ovog algoritma dodatno moguće redukovati skup podataka, jer ne utiče na duplikate.

- SMOTEENN kao nadogradnja ENN algoritma daje slične rezultate kao i on kada je u pitanju tačnost predikcije, međutim kada je u pitanju redukcija broja instanci, pokazao se najefikasnije od svih algoritama, gde su opravdana prirodna očekivanja, a to je da hibridni algoritmi daju najbolje rezultate, međutim i on ostavlja prostora za napredak i dodatno smanjenje broja instanci dok je vremenski zahtevniji od svog prethodnika.

- NCR algoritam je u ukupnom zbiru dao rezultate slične ENN algoritmu i on kao pripadnik porodice *condensed* algoritama predstavlja realnog predstavnika koji bi mogao biti primenjen nad nekim skupom podataka. Takodje ostavlja prostora za napredak.

Algoritmi su dali očekivane rezultate, od osnovnog ka naprednijima poboljšale su se performanse, ali i povećala vremenska zahtevnost. Dokazano je i da ima prostora za dodatno smanjenje broja instanci sto govori o tome da su rezultati konačne redukcije zavisni ne samo od ove vrste algoritama, nego i od dobre predobrade i analize podataka, sto vodi ka krajnjem zaključku da ljudski faktor mora biti prisutan kako bi bila izvršena kvalitetna redukcija broja instanci skupa podataka.

| | algorithm | time | accuracy | percentage_instances_end | num_instances_start | num_instances_end | precision | f1-score | num_duplicated_start | num_duplicated_end |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ENN | 0.330129 | 0.9140 | 89.950000 | 8000.0 | 7196.0 | 0.917002 | 0.913796 | 0.0 | 0.0 |
| 1 | TomekLinks | 0.399992 | 0.9160 | 97.050000 | 8000.0 | 7764.0 | 0.916067 | 0.915989 | 0.0 | 0.0 |
| 2 | Original | 0.484540 | 0.9180 | 100.000000 | 8000.0 | 8000.0 | 0.918035 | 0.918003 | 0.0 | 0.0 |
| 3 | SMOTEENN | 0.562431 | 0.9275 | 80.462500 | 8000.0 | 6437.0 | 0.927515 | 0.927502 | 0.0 | 0.0 |
| 4 | NCR | 0.785446 | 0.9120 | 90.100000 | 8000.0 | 7208.0 | 0.915129 | 0.911782 | 0.0 | 0.0 |
| 5 | CNN | 41.328188 | 0.9050 | 58.766667 | 3000.0 | 1763.0 | 0.907436 | 0.904906 | 0.0 | 0.0 |

*Slika 13. Rezultati primene algoritama na citrus set podataka*

Slika 13., pokazuje pravu snagu hibridnih algoritama, pored redukcije broja instanci za cak 20%(najviše od svih algoritama izuzev CNN), tačnost predikcije ne samo da je održana, nego je i povećana u odnosu na tačnost predikcije bez primene bilo kakve redukcije, sto SMOTEENN algoritam čini najefikasnijim algoritmom od svih obrađenih.

Rang lista efikasnosti algoritama:

1. SMOTEENN
2. NCR
3. ENN
4. TomekLinks
5. CNN

Nakon primene dodatnih metoda za redukciju dimenzionalnosti, početni set podataka koji je zauzimao 102MB memorije je smanjen na 54MB, bez velikih gubitaka u efikasnosti, sto se može smatrati dobrim rezultatom redukcije dimezionalnosti.

# Zaključak

Ovaj rad je posvećen istraživanju Instance Selection metoda i principa. Ova oblast u mašinskom učenju je nastala zbog povećanja količine podataka za obradu i treniranje modela, postalo je nemoguće obraditi tolike količine podataka pa se pribeglo metodama za redukciju dimenzionalnosti setova podataka. U ovom radu je opisan jedan način rešavanja tog problema, a to je redukcija dimenzionalnosti, oblast kompjuterskih nauka koja takođe studira ovaj problem jeste Bigdata iliti sistemi za obradu i preprocesiranje velikih količina podataka, koja ovaj problem Resava tako sto omogućava vurtuelnu neograničenu količinu hardverskih resursa na udaljenim serverima. Oblast slična ovoj jeste takođe i undersampling (uglavnom rešavanje problema sa balansiranjem), međutim iako su oblasti tesno povezane, u radu je pokusano maksimalno izbegavanje ovog termina, kako bi se istakla poenta, a to je smanjenje količine podataka bez gubitaka u performansama modela.

Prototype Selection (PS), kao osnovni pristup redukciji dimenzionalnosti baca u senku Training Set Selection (TSS), zbog čega je ovaj rad fokusiran na prvopomenuti pristup, sto ne umanjuje vrednost i efikasnost drugog pristupa. PS se fokusira na kreiranje podskupa seta podataka ulaznog trening seta podataka bez gubitaka tačnosti u aproksimacijama test podataka i sa smislenim rešavanjem problema redukcije, a ne slučajnim izbacivanjem određenih elemenata.

Postoji vise početnih tačaka (tipova pretrage) kod PS pristupa, uglavnom se počinje od praznog skupa podataka koji se kasnije povećava odabirom instanci od interesa (incremental) ili pristup suprotan ovome (decremental) koji počinje od trening skupa podataka i formira podskup podataka izbacivanjem nepotrebnih instanci podataka iz početnog skupa podataka. Takođe imamo mixed, batcx i fixed.

Pored toga postoje i tipovi selekcije od kojih svaki ima svoje specifičnosti, Condensation tip ima za cilj da zadrži granične tačke između klasa pri izboru instanci, dok se rešava instanci koje su unutrašnje tj. instanci koje nemaju uticaja na graničenja, ovaj pristup je veoma efikasan u redukciji dimenzionalnosti. Edition s druge strane uglavnom uklanja instance koje su *noisy,* sto automatski znači da se odbacuju instance koje su na graničnim pozicijama, sto rezultuje ne toliko uspešnom redukcijom podataka zbog viska nepotrebnih unutrašnjih instanci, ali i ne narušava tačnost evaluacije. Hibridne metode pokušavaju da nađu kompromis između Condensation i Edition metoda pa u skladu sa tim uglavnom implementiraju kombinacije algoritama iz dve prethodne grupe, ove metode bi trebale da obezbede bolju tačnost klasifikacije test podataka i zadovoljavajuću redukovanost podataka.

U radu je opisano pet algoritama za redukciju podataka, svaki od njih je pokazao neke dobre i loše strane. Obrađeni su sledeći algoritmi CNN, ENN, Tomek Links, DROP i SMOTEENN. Svaki od njih ima svoje prednosti i mane i svaki se u nekom segment pokazao najboljim. CNN algoritam se može koristiti kada želimo da redukujemo veliku količinu podataka gde se može zanemariti određeni gubitak performansi, vremenska zahtevnost je jedna od najvećih mana ovog algoritma. Algoritam koji se koristi u paru sa

CNN je Tomek Links i on ima dve metode kojima poboljšava i ubrzava rad svog partnera.
ENN je odličan u slučajevima kada želimo da izbrišemo *noisy* podatke iz skupa podataka, međutim on se preporučuje za kombinacije sa nekim drugim algoritmima, na primer SMOTE, zbog svoje osobine da ne redukuje značajno količinu podataka.
Algoritmi iz porodice DROP algoritama predstavljaju hibridne algoritme, koji koriste osobine prethodno opisanih algoritama kako bi redukovali podatke i povećali tačnost, a sve to za malo vremena, i oni su odlično konačno rešenje, kao jedan od njihovih nedostataka bih pored svih prednosti koje imaju, naveo i to da većina ovih algoritama nije implementirana u osnovnim bibliotekama programskih jezika Python i R, sto je takođe bitan faktor u implementaciji projekata, jer bi ovo trebali biti pomoćni algoritmi koji se koriste na poziv i tu su da asistiraju u boljem preprocesiranju podataka.

U praktičnom delu je pokazano da su hibridni algoritmi najbolje rešenje za problem velike dimenzionalnosti podataka, ali i to da sama primena ovih algoritama nad nekim skupom podataka ne rezultuje nužno maksimalnom redukcijom instanci podataka, obzirom na to da svi algoritmi ostavljaju duplikate za sobom (sto je dovoljno za tvrdnju da algoritmi nisu stopostotno efikasni), jasno je da je potrebno dodatno preprocesiranje i predobrada podataka kako bi se, uz selekciju atributa, postigli savršeni rezultati.

Ova oblast je veoma zanimljiva, ali nedovoljno istražena. Možda zbog činjenice da se u današnje vreme ne obraća pažnja na problem količine podataka zbog sve moćnijih hardverskih resursa računara i udaljenih servera koji simuliraju beskonačne resurse. Ali jedan od razloga je i taj sto je ovo preprocesiranje podataka koje vodi ka treniranju modela, koje se u sustini izvršava samo jednom, tako da programi u osnovi ne gube na performansama.

# Literatura


[1] J. L. Salvador García, in *Data Preprocessing in Data Mining*.

[2] J. A. O.-L. ·. J. A. C.-O. ·, in *A review of instance selection methods*, https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/pr/2010-Olvera-AIR.pdf.

[3] S. Herbst, "An Introduction to Instance Selection in Data Mining," https://medium.com/@sabrinaherbst/an-introduction-to-instance-selection-in-data-

mining-7b2ac94fb07.

[4] J. B. a. R. Tibshirani, in *PROTOTYPE SELECTION FOR INTERPRETABLE*, https://arxiv.org/pdf/1202.5933.pdf.

[5] J. Brownlee, in *Undersampling Algorithms for Imbalanced Classification*, https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/.

[6] E. Alpaydın, in *Voting over Multiple Condensed Nearest Neighbors*, https://www.researchgate.net/publication/2644628_Voting_over_Multiple_Condensed_Nearest_Neighbors#pf4.

[7] I. TOMEK, in *Two Modifications of CNN*, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4309452.

[8] N. Z. R. V. Christoph F. Eick, in *Using Representative-Based Clustering for Nearest Neighbor Dataset Editing*, https://www.researchgate.net/publication/4133603_Using_Representative-Based_Clustering_for_Nearest_Neighbor_Dataset_Editing.

[9] R. A. A. Viadinugroho, in *Imbalanced Classification in Python: SMOTE-ENN Method*, https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50.