Traženje podgrupa iz višepoglednih podataka

Martinić, Mislav

Sveučilište u Zagrebu
Prirodoslovno-matematički fakultet
Matematički odsjek

Mislav Martinić

Traženje podgrupa iz višepoglednih podataka

Diplomski rad

Voditelj rada:
doc. dr. sc. Matej Mihelčić

Zagreb, studeni 2021.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____ , predsjednik
2. _____ , član
3. _____ , član

Povjerenstvo je rad ocijenilo ocjenom _____ .

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

1 Uvod

1.1 Traženje podgrupa

Traženje podgrupa je široko prihvaćena tehnika dubinske analize podataka kojoj je cilj otkriti zanimljive skupove primjera koji imaju neuobičajenu distribuciju jednog posebno zanimljivog atributa, koji zovemo ciljna labela. Ukoliko imamo skup podataka u kojem su primjeri ljudi oboljeli od dijabetesa, tada je ciljna labela ima li osoba dijabetes ili nema. Zanimljivo je otkriti zašto neki primjeri imaju točno određenu vrijednost te ciljne labele. Odnosno, ciljna labela u ovom slučaju je binarna ciljna labela sa samo 2 vrijednosti. Moguće je da u nekim skupovima podataka postoji više ciljnih labela. Također, one mogu biti binarnog tipa kao u danom primjeru, kategorijske ili numeričke. Kategorijske ciljne labele su one koje poprimaju vrijednost iz nekog konačnog skupa vrijednosti. Podskupovi primjera koji imaju neobičnu distribuciju vrijednosti ciljne labele nazivaju se podgrupe i opisuju se pravilima kod zadatka traženja podgrupa.

Traženje podgrupa ima širok raspon primjene u različitim područjima znanosti gdje je bitno kako su određeni deskriptivni atributi i njihove vrijednosti povezani sa specifičnom ciljnom labelom.
U medicini se često koristi za detekciju rizika koronarnih srčanih bolesti koristeći bitne faktore poput visoke razine kolesterola i triglicerida. Druga zanimljiva primjena je traženje podgrupa kod pacijenata koji posjete hitnu psihijatrijsku pomoć u odnosu na vrijeme posjete. Navedena primjena prikazana je u radu autora Carmona C.J. i ostali [1] gdje se koristi jedan od novijih algoritama genetskog SDIGA (*Subgroup Discovery Iterative Genetic Algorithm*). U bioinformatici se koristi za traženje podgrupa koje imaju visoku stopu pojavnosti različitih tipova tumora, a ponajviše raka dojke. Postoje studije [5, 6] koje su otkrile grupu gena koja doprinosi razvoju limfoblastične leukemije koristeći traženje podgrupa. Traženje podgrupa se još koristi u margetingu [10] i prometu [7, 8], ovo su samo neki primjeri, postoje još mnogi drugi.

1.2 CN2SD

Jedan od algoritama za traženje pogrupa, a u isto vrijeme bazni algoritam ovog rada, je CN2SD. Napravljen je kao nadogradnja starijeg CN2 algoritma. CN2 algoritam je algoritam za indukciju pravila. Inducirana pravila su oblika "If *Cond* then *Class*" gdje je Cond (uvjet) konjunkcija konačno mnogo jednostavnih uvjeta. *Class* reprezentira vrijednost cijlne labele pravila. Za pravilo ćemo koristiti notaciju *Class* $\leftarrow$*Cond*.

Također ćemo definirati pojmove jednostavan uvjet i kompleksan uvjet. Jednostavan uvjet – relacija između atributa i vrijednosti (GLUKOZA ≤ 6.0) Kompleksan uvjet – konjunkcija više jednostavnih uvjeta (GLUKOZA ≤ 6.0 i DOB < 60)

Primjer pravila:

Ako boja cvijeta = "crvena" & trnje = "true" onda Klasa ="ruža"

CN2 koristi zrakasto pretraživanje (eng. beam search) koje koristi klasifikacijsku točnost (eng. Classification Accuracy, kraće *Acc*) definiranu kao:

$$Acc(Class \leftarrow Cond) = p(Class \mid Cond) = \frac{p(Class.Cond)}{p(Cond)}$$

kao metodu ocjenjivanja pravila kako bi donji proces (koristeći zrakasto pretraživanje) inducirao jedno pravilo te proces koji kontrolira to zrakasto pretraživanje kako bi algoritam inducirao skup pravila. Algoritam ignorira primjere skupa podataka pokrivene pronađenim pravilom kako bi u idućoj iteraciji zrakasto pretraživanje pronašlo novo pravilo koje opisuje drugačije, prije neopisane primjere. U slučaju kad kontrolni proces nebi izbacivao pokrivene primjere, algoritam bi vraćao jedno te isto pravilo.

U radovima [2] [3] predstavljena je nova varijanta ocjenjivanja pravila nazvana *Težinska relativna točnost* (eng. Weighted relative accuracy, kraće *WRAcc*), definirana kao:

$$WRAcc(Class \leftarrow Cond) = p(Cond) \cdot (p(Class \mid Cond) - p(Class)).$$

Bazni algoritam ovog rada, CN2SD, nadogradnja je CN2 algoritma. Kako bi algoritam bio pogodan za traženje podgrupa, odnosno kako bi primjeri pokriveni prethodnim pravilom bili smatrani manje relevantnim umjesto eliminirani iz skupa podataka, uvedeno je "težinsko pokrivanje" primjera pokrivenih pravilima umjesto izbacivanja istih. *Modificirana težinska relativna točnost*, (eng. Modified Weighted Relative Accuracy, kraće *mWRAcc*) definirana je na sljedeći način:

$$mWRAcc(Class \leftarrow Cond) = \frac{n'(Cond)}{N'} \cdot \left(\frac{n'(Class.Cond)}{n'(Cond)} - \frac{n'(Class)}{N'} \right).$$

N' predstavlja sumu težina svih primjera, $n'(Cond)$ je suma težina svih pokrivenih primjera, $n'(Class.Cond)$ je suma svih težina pravilno pokrivenih primjera, a $n'(Class)$ suma svih težina primjera klase $Class$. Pokriven primjer je onaj koji koji je opisan pravilom, odnosno zadovoljava sve uvjete pravila. Pravilno pokriven primjer je onaj koji je opisan pravilom i ima vrijednost ciljne labele jednaku vrijednosti ciljne labele za koju se računa težinska točnost.

1.2.1 Zrakasto pretraživanje

Zrakasto pretraživanje (eng. Beam search) je pohlepan algoritam pretraživanja sličan algoritmima pretraživanja u širinu (eng. Breadth-First Search, kraće BFS) i pretraživanju korištenjem najboljeg prvog (eng. Best First Search, kraće BeFS). Zapravo, vidjet ćemo da su BFS i BeFS specijalni slučajevi zrakastog pretraživanja.
Pretpostavimo da imamo graf G koji želimo prijeći kako bismo došli do određenog čvora. Prvi korak je taj da proširimo korijen sa svim mogućim čvorovima (djecom). Tada, u svakom sljedećem koraku algoritma, izaberemo unaprijed određen specifični broj čvorova, koji zovemo *širina zrake* (označimo sa β). Izabrani čvorovi su oni najbolji odabrani prema nekoj heuristici.
Primijetimo, kada je $\beta = 1$, tada zrakasto pretraživanje postaje BeFS budući da ćemo izabrati jedan najbolji mogući čvor u svakom koraku algoritma, upravo kako BeFS radi. S druge strane, kada je $\beta = \infty$, zrakasto pretraživanje postaje BFS. Razlog je taj što u svakom koraku algoritma prilikom biranja čvorova izaberemo sve moguće, odnosno svu djecu čvora, bez korištenja heuristike.

1.2.2 Pseudokod CN2SD

U ovom pododjeljku ćemo opisati CN2SD na način da krenemo od pisanja pseudokoda **main** funkcije te dalje napišemo pseudokod pozivanih funkcija redom.

Algorithm 1 CN2SD traženje podgrupa

```
1: function MAIN
2:     Podesi glavne parametre algoritma (beamSize, maxIterations, min-
   Significance, gamma, xValidationFolds, mWRAccFactor)
3:     Obradi skup podataka iz .csv dokumenta
4:     folds[ ] ← Stvaranje stratificiranih uzoraka
5:     for fold : folds do
6:         CN2SD cn2sd ← new CN2SD(fold.train, glavni parametri)
7:         rules[ ] ← cn2sd.traziPodgrupe()
8:         for rule : rules do
9:             r.evaluiraj(fold.test)
10:        end for
11:        ruleSets.add(rules)
12:    end forreturn ruleSets
13: end function
```

Algorithm 2 CN2SD traži podgrupe

```
1: function TRAŽIPODGRUPE(skup podataka)
2:     targets [ ] ← Odredi ciljne labele skupa podataka
3:     for target : targets do
4:         bestRule ← traziPravilo (target, minConfidence)
5:         if bestRule != null then
6:             minmWRAcc ← bestRule.mWRAcc · mWRAccFactor
7:         end if
8:         while bestRule != null do
9:             ruleSet.add(bestRule)
10:            bestRule.cover(skup podataka, gamma)
11:            bestRule ← traziPravilo(target, minConfidence)
12:        end while
13:    end for
14:    return ruleSet
15: end function
```

Algorithm 3 Generator jednostavnih uvjeta

```
1: function GENERIRAJJEDNOSTAVNEUVJETE(skup podataka)
2:     atributi←atributi skupa podataka
3:     for atribut : atributi do
4:         if atribut==ciljna labela then
5:             continue
6:         end if
7:         if atribut==kategorijski then unique[ ]
8:             for primjer : skup podataka do
9:                 value ← vrijednost atributa
10:                if !unique.contains(value) then unique.add(value)
11:                end if
12:            end for
13:            for value : unique do
14:                cond1←new SimpleCondition(atribut jednak value)
15:                cond2←new SimpleCondition(atribut razlicit value)
16:                simpleConditions.add(cond1, cond2)
17:            end for
18:        end if
19:        if atribut==numericki then unique[ ]
20:            for primjer : skup podataka do
21:                value ← vrijednost atributa
22:                if !unique.contains(value) then unique.add(value)
23:                end if
24:            end for
25:            for value : unique do
26:                cond1←new SimpleCondition(atribut veći od value)
27:                cond2←new SimpleCondition(atribut manji ili jednak value)
28:                simpleConditions.add(cond1, cond2)
29:            end for
30:        end if
31:    end for
32:    return simpleConditions
33: end function
```

Algorithm 4 CN2SD težinsko pokrivanje primjera

```
1: function TEŽINSKOPOKRIVANJE(skup podataka, gamma)
2:     for primjer : set podataka do
3:         if primjer zadovoljava pravilo then
4:             primjer.pokriven += 1
5:             primjer.težina = pow(gamma, primjer.pokriven)
6:         end if
7:     end for
8: end function
```

Algorithm 5 CN2SD evaluiraj podgrupu

```
1: function EVALUIRAJPODGRUPU(skup podataka)
2:     Postavi nCond, nClassCond, CondW, ClassCondW, nW, nClass na 0
3:     for primjer : skup podataka do
4:         supported[ ]
5:         nW + = primjer.težina
6:         satisfies ← primjer.satisfies(rule)
7:         if satisfies then
8:             nCond++
9:             condW + = primjer.težina
10:            if primjer.Class == rule.target then
11:                supported.add(index primjera)
12:                nClassCond++
13:                ClassCondW + = primjer.težina
14:            end if
15:            Dodaj primjer klasnoj distribuciji
16:        end if
17:        if primjer.Class == rule.target then
18:            ClassW + = primjer.tezina
19:            nClass + = 1
20:        end if
21:    end for
22:    mWRAcc = CondW/nW · (ClassCondW/CondW − ClassW/nW)
23:    pCond = nCond/n gdje je n veličina skupa podataka
24:    pClassCond = nClassCond/nCond
25:    pClass = nClass/n gdje je n veličina skupa podataka
26:    WRAcc = pCond · (pClassCond − pClass)
27: end function
```

Algorithm 6 CN2SD traži pravilo

```
1: function TRAŽIPRAVILO(target, minConfidence, jednostavni uvjeti)
2:     bestRule ← null
3:     ruleList[ ] ← null
4:     for jednostavan uvjet : jednostavni uvjeti do
5:         rule ← new Rule(jednostavan uvjet, target)
6:         rule.evaluiraj()
7:         ruleList.add(rule)
8:     end for
9:     ruleList.sort()
10:    Izbriši pravila ruleLista, osim prvih beamSize njih
11:    currentIteration ← 0
12:    while currentIteration < maxIterations do
13:        for i ← 0; i < beamSize; i + + do
14:            for jednostavan uvjet : jednostavni uvjeti do
15:                validan ← ruleList[i].uvjet.isValid(jednostavan uvjet)
16:                if validan then
17:                    kompleksan uvjet ← ruleList[i].uvjet(jednostavan uvjet)
18:                    newRule ← new Rule(kompleksan uvjet)
19:                    newRule.evaluiraj()
20:                    ruleList.add(novo pravilo)
21:                end if
22:            end for
23:        end for
24:        ruleList.sort()
25:        Izbriši pravila ruleLista, osim prvih beamSize njih
26:        if ruleList[0] zadovoljava uvjete iz main-a then
27:            if (
28:                bestRule == null or
29:                ruleList[0].mWRAcc > bestRule.mWRAcc
30:                ) then
31:                bestRule = prvo pravilo ruleList-a
32:            end if
33:        end if
34:        currentIteration++
35:    end while
36:    return bestRule
37: end function
```

1.2.3 Opis rada CN2SD

CN2SD počinje tako da se u main funciji odrede glavni parametri algoritma, učita se skup podataka iz .csv dokumenta te se na njemu napravi k stratificiranih uzoraka, odnosno *xValidationFolds* skupova. Skup podataka se podijeli na train i test uzorak tako da se $k-1$ uzorak koristi za treniranje, a k-ti se koristi za testiranje te se tako podijeli za svaki $0 < k < xValidationFolds$. Na train skupu CN2SD inducira pravila, a na test skupu ih testira. Zatim se za svaki skup unakrsne validacije kreira klasa CN2SD u koju se spremaju glavni parametri algoritma te se pokreće funkcija *generirajJednostavneUvjete* (Algoritam 3). Ova funkcija iterira po svim atributima skupa podataka te svim njihovim vrijednostima, zatim ako je atribut kategorijski, za svaku jedinstvenu vrijednost tog atributa, Algoritam 3 (8-12), stvori 2 jednostavna uvjeta, atribut jednak vrijednost, Algoritam 3 (14), i atribut različit od vrijednosti, Algoritam 3 (15). U slučaju da je atribut numerički, analogno spremi sve jedinstvene vrijednosti tog atributa, Algoritam 3 (20-24), te za svaku od njih stvori 2 jednostavna uvjeta, atribut veći od vrijednosti, Algoritam 3 (26), atribut manji od ili jednak vrijednosti, Algoritam 3 (27). Nakon toga funkcija spremi sve jednostavne uvjete u klasu CN2SD.

Potom se pokreće funkcija *traziPodgrupe* (Algoritam 2) koja predstavlja vanjski proces koji kontrolira i pokreće unutarnje zrakasto pretraživanje za induciranje pravila. Prvo odredi sve ciljne labele skupa podataka, Algoritam 2 (2), zatim za svaku od njih, Algoritam 2 (3-13), zove funkciju traziPravilo koja predstavlja unutarnje zrakasto pretraživanje. *traziPodgrupe* pozove *traziPravilo* (Algoritam 6) jednom tako da bi se odredio prag relevantnosti, Algoritam 2 (6), za daljnja pravila. Zatim zove funkciju *traziPravilo* dok postoje dosta dobra pravila, Algoritam 2 (8-12). Dosta dobra pravila su ona koja zadovoljavaju prag relevantnosti, Algoritam 2 (6), minimalnu uvjerljivost i minimalnu signifikantnost. Za svako vraćeno pravilo pokrije primjere skupa podataka funkcijom *težinskoPokrivanje* (Algoritam 4) tako da iterira kroz sve primjere skupa podataka, Algoritam 4 (2), te ako oni zadovoljavaju pravilo, Algoritam 4 (3), *težinskoPokrivanje* ih pokriva tako da im smanji težinu na način da podigne početni parametar $0 < \gamma < 1$ na p-tu potenciju gdje p predstavlja broj pravila koja pokrivaju primjer.

Zrakasto pretraživanje koje je predstavljeno funkcijom *traziPravilo* (Algoritam 6) na ulaz prima ciljnu labelu za koje se pravilo inducira te jedan kontrolni parametar. Za svaki spremljeni jednostavan uvjet, Algoritam 6 (3-7), generira pravilo, Algoritam 6 (4), koje se sastoji od tog uvjeta te ih ocjeni, Algoritam 6 (5), funkcijom *evaluirajPravilo* (Algoritam 5) i doda u listu pra-

vila, Algoritam 6 (6). Ta lista se sortira silazno po $mWRAcc$-u, Algoritam 6 (8), i brišu se pravila s liste osim prvih $beamSize$ njih, Algoritam 6 (9). Sada krećemo u pretraživanje koje ide u dubinu $maxIterations$, Algoritam 6 (11-33), odnosno svako generirano pravilo može imati najviše $maxIterations$ jednostavnih uvjeta. Za svako pravilo na zraci, Algoritam 6 (12-22), proširi kompleksan uvjet tog pravila sa svakim jednostavnim uvjetom, Algoritam 6 (13-20), na način da se prvo provjeri bi li proširen uvjet bio validan, Algoritam 6 (14). Validan uvjet je onaj koji ne sadrži više jednostavnih uvjeta s istim nazivom atributa i relacijom između tog atributa i vrijednosti (npr. AGE $<$ 60 i AGE $<$ 50 nije validan uvjet) te onaj koji sadrži više jednostavnih uvjeta s istim nazivom atributa, ali različitom relacijom između tog atributa i vrijednosti i njihov presjek nije prazan (npr. AGE $<$ 60 i AGE $>$ 90 nije validan uvjet). Ako je uvjet validan, Algoritam 6 (15-20), proširi ga tim jednostavnim uvjetom, Algoritam 6 (16), stvori novo pravilo s tim kompleksnim uvjetom, Algoritam 6 (17), ocjeni ga, Algoritam 6 (18), te ga doda listi pravila, Algoritam 6 (19). Lista pravila se ponovo sortira prema $mWRAccu$ i odbacuju se sva pravila osim najboljih $beamSize$ (veličina zrake) njih, Algoritam 6 (24, 25). Sada, ako trenutno najbolje pravilo zadovoljava minimalne uvjete zadane na početku, Algoritam 6 (25), te ako još nismo našli najbolje pravilo ili ako je trenutno najbolje pravilo bolje od prethodno spremljenog najboljeg pravila, Algoritam 6 (26-29), onda se spremi trenutno pravilo kao najbolje pravilo nađeno dosad, Algoritam 6 (30). *traziPravilo* nakon $maxIterations$ iteracija vraća najbolje nađeno pravilo.

Pravila se ocjenjuju funkcijom *evaluirajPravilo* (Algoritam 5) tako da za svaki primjer skupa podataka, Algoritam 5 (3-21), njegova težina se spremi u zbroj svih težina nW, Algoritam 5 (5) te se provjeri zadovoljava li primjer pravilo, Algoritam 5 (6). Ako primjer zadovoljava pravilo, povećaj broj primjera koji zadovoljavaju pravilo $nCond$, Algoritam 5 (8), te pribroji težinu primjera zbroju težina svih primjera koje pravilo pokriva - $condW$, Algoritam 5 (9). Zatim ukoliko je primjer klase *Class* i ta klasa je jednaka ciljnoj labeli pravila, Algoritam 5 (10-15), dodaj indeks primjera u niz indeksa primjera koji zadovoljavaju uvjete, Algoritam 5 (7), i, Algoritam 5 (10), te povećaj broj primjera koji zadovoljavaju pravilo i vrijednosti njihove ciljne labele su jednake ciljnoj labeli pravila - $nClassCond$, Algoritam 5 (11), pribroji težinu primjera zbroju težina svih primjera koje pravilo pokriva i vrijednosti njihove ciljne labele su jednake ciljnoj labeli pravila $ClassCondW$, Algoritam 5 (13). Ako primjer zadovoljava pravilo, dodaj ga u distribuciju vrijednosti ciljne labele pravila. Distribucija vrijednosti ciljne labele pravila nam govori koliko primjera koje klase je pravilo pokrilo.
Dalje, za sve primjere koji su klase jednake ciljnoj labeli pravila, Algoritam

5 (17-20), zbroji ih, Algoritam 5 (18), te sumiraj njihove težine, Algoritam 5 (19).
Na kraju izračunaj *mWRAcc, pCond, pClassCond, pClass, WRacc* pravila prema njihovim formulama, Algoritam 5 (22-26).

2 Proširivanje algoritma iz jednopoglednog u višepogledni

Suština ovog rada bila je proširiti navedeni CN2SD algoritam kako bi on tražio podgrupe iz skupa podataka koji se sastoji od dva pogleda, a ne samo od jednog kako radi originalan CN2SD. Pogledi skupa podataka koje prošireni algoritam prima na ulaz moraju imati iste primjere opisane različitim atributima s istom ciljnom labelom. Koristeći iznad navedeni primjer za dijabetes, možemo imati jedan pogled koji sadržava ljude kao primjere s ciljnom labelom jesu li oboljeli ili ne, opisani svojim jednostavnim biološkim karakteristikama kao što su dob, spol, starost, težina, BMI i slično te drugi pogled tih istih primjera (ljudi) s istom ciljnom labelom, samo što su opisani, npr. nekim medicinskim nalazima poput razine glukoze u krvi, krvni tlak, razina koncentracije eritrocita, leukocita, trombocita i slično.

Za ovakvo proširenje, svaka podgrupa će biti sada opisana parom pravila. Odnosno, podgrupu čine primjeri koji se nalaze u presjeku podgrupa određenih tim parom pravila. No, tada ćemo morati definirati novu mjeru dobrote budući da sada više nemamo jedno pravilo, već par pravila.

Primjer para pravila:

Ako DOB > 60 & SPOL = "muški" onda Klasa ="true"
Ako GLUKOZA > 6.5 & SISTOLIČKI TLAK > 140 onda Klasa ="true"
Pod Klasa="true" se podrazumijeva da osoba pati od dijabetesa.

2.1 Multiview score

2.1.1 Jaccard indeks

Jaccard indeks, poznat kao Jaccardov koeficijent sličnosti, je statistička mjera korištena za mjerenje sličnosti, odnosno različitosti nekih konačnih skupova. Razvio ga je Paul Jaccard, originalno mu dajući ime *coefficient de communauté*. Jaccardov indeks je definiran kao kvocijent dijeljenja kardinaliteta

presjeka skupova s kardinalitetom njihove unije.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Primjer:

$$A = \{2, 3, 5, 7, 11, 13\}$$
$$B = \{1, 2, 3, 5, 8, 13\}$$

$$Jaccard(A, B) = \frac{4}{8} = 0.5$$

Budući da je svaka podgrupa sada označena parom pravila, svako pravilo ima zasebni *mWRAcc* u odnosu na pogled višepoglednog skupa podataka na koji se pravilo odnosi. Pojavila se potreba za novom metodom ocjenjivanja para pravila. Prvi prijedlog bio je uzeti aritmetičku sredinu *mWRAcc*-a pravila u paru te pomnožiti s Jaccard indeksom dva skupa primjera označena tim pravilima. Tako definirana metoda ocjenjivanja se nije pokazala dobrom budući da aritmetička sredina *mWRAcc*-a nije gledala na presjek primjera para pravila tako da sam na mentorov prijedlog umjesto te aritmetičke sredine uveo novu ocjenu *Modified Weighted Relative Accuraccy Intersection*, kraće *mWRAccIs*, definiranu na sljedeći način:

$$mWRAccIs(Class \leftarrow Cond1 \cap Cond2) =$$
$$\frac{n'(Cond1 \cap Cond2)}{N'} \cdot \left(\frac{n'(Class.(Cond1 \cap Cond2))}{n'(Cond1 \cap Cond2)} - \frac{n'(Class)}{N'} \right)$$

gdje $n'(Cond1 \cap Cond2)$ označava sumu težina presjeka primjera, N' sumu težina svih primjera, $n'(Class.(Cond1 \cap Cond2))$ sumu težina presjeka primjera para pravila podgrupe pravilno pokrivenih primjera, a $n'(Class)$ sumu težina svih primjera skupa podataka te klase.

Nova mjera za evaluaciju višepoglednih pravila sada glasi:

$$Multiview\ score = mWRAccIs \cdot Jaccard$$

Ocjena je definirana na ovaj način kako bi dobili podgrupe u kojoj su za svaki pogled skupovi primjera pokriveni pravilima čim sličniji, o čemu nam govori *Jaccard*, a da u isto vrijeme imaju visok *mWRAccIs*.

2.2 Naivni pristup proširenju

Na moj je prijedlog prvo bila obrađena naivna metoda za računanje višepoglednih podgrupa. Ideja ovog pristupa proširenja bila je izvršiti CN2SD na svakom pogledu skupa podataka zasebno te nakon toga napraviti kartezijev produkt skupova pravila.

Algorithm 7 CN2SD Traženje podgrupa naivni višepogledni

```
1: function MAIN
2:     Podesi glavne parametre algoritma (beamSize, maxIterations, min-
   Significance, gamma, xValidationFolds, mWRAccFactor)
3:     Obradi oba pogleda skupa podataka iz .csv dokumenata
4:     folds1[ ] ←stratificirani uzorci(DataView1, xValidationFolds)
5:     folds2[ ] ← stratificirani uzorci(DataView2, xValidationFolds) rule-
   CompFoldList [ ]
6:     for i ← 0, i <xValidationFolds do
7:         CN2SD cn2sd1 ← new CN2SD(folds1[i].train, glavni parametri)
8:         CN2SD cn2sd2 ← new CN2SD(folds2[i].train, glavni parametri)
9:         rules1[ ] ← cn2sd1.run()
10:        rules2[ ] ← cn2sd2.run()
11:        for rule : rules1 do
12:            r.evaluiraj(folds1.test)
13:        end for
14:        for rule : rules2 do
15:            r.evaluiraj(folds2.test)
16:        end for
17:        ruleSets1.add(rules1)
18:        ruleSets2.add(rules2)
19:        ruleCompList [ ]
20:        for rule1:rules1 do
21:            for rule2:rules2 do
22:                rc = new ruleComp(rule1, rule2)
23:                rc.evaluiraj
24:                ruleCompList.add(rc)
25:            end for
26:        end for
27:        ruleCompFoldList.add(ruleCompList)
28:    end for
29:    ruleCompFoldList.sort
30:    return ruleCompFoldList
31: end function
```

Algorithm 8 Naivni višepogledni CN2SD evaluiraj podgrupu (višepogledni)

function EVALUIRAJ(proizvoljni pogled skupa podataka, presjek, ciljna labela)
    Postavi nCond, nClassCond, CondW, ClassCondW, nW, nClass na 0
    for primjer : jedan pogled skupa podataka **do**
        supported[ ]
        nW + = primjer.težina
        satisfies ← presjek.contains(primjer)
        if satisfies **then**
            nCond++
            condW + = primjer.težina
            if primjer.Class == ciljna labela **then**
                supported.add(index primjera)
                nClassCond++
                ClassCondW + = primjer.težina
                Dodaj primjer klasnoj distribuciji
            end if
        end if
        if primjer.Class == ciljna labela **then**
            ClassW + = primjer.tezina
            nClass + = 1
        end if
    end for
    $mWRAccIs = \frac{CondW}{nW} \cdot (\frac{ClassCondW}{CondW} - \frac{ClassW}{nW})$
    $pCond = \frac{nCond}{n}$ gdje je n veličina skupa podataka
    $pClassCond = \frac{nClassCond}{nCond}$
    $pClass = \frac{nClass}{n}$ gdje je n veličina skupa podataka
    $WRAccIs = pCond \cdot (pClassCond - pClass)$
    $MultiviewScore = Jaccard \cdot mWRAccIs$
end function

Inducirani parovi pravila bi zatim bili ocjenjeni prema višepoglednoj mjeri za evaluaciju podgrupa te sortirani po istom. Za početak smo trebali definirati novu klasu pod nazivom ruleComp koja je u sebi sadržavala par pravila i mjere tog para kao što su *Jaccard, mjera za dobrotu, mWRAccIs, presjek primjera.*

Spajali smo pravila po stratificiranim uzorcima, Algoritam 7 (20-25), kako bi par mogli ocijeniti, Algoritam 7 (23), prema odgovarajućim uzorcima. Isto, radi mogućnosti ocjenjivanja, spajali smo pravila koja imaju iste ciljne labele.

Mana mu je što se spajaju pravila koja su inducirana nevezano jedno

uz drugo, tako da je naivni višepogledni CN2SD inducirao parove s niskim *Jaccard indeks*-om.

2.3 Kompleksan pristup višepoglednom traženju podgrupa

Algorithm 9 CN2SD Traženje podgrupa kompleksni višepogledni

```
1: function MAIN
2:     Podesi glavne parametre algoritma (beamSize, maxIterations, min-
       Significance (uvjet za pravilo), gamma, xValidationFolds, mvScore-
       Factor (uvjet za pravilo), minBinDistThreshold (uvjet za pravilo,
       prag za pVal))
3:     Učitaj oba pogleda skupa podataka iz .csv dokumenata
4:     folds1[ ] ← Stvaranje stratificiranih foldova za prvi pogled
5:     folds2[ ] ← Stvaranje stratificiranih foldova za drugi pogled koji su
       identični prvom
6:     for i ← 0, i <xValidationFolds do
7:         CN2SDMV cn2sd ← new CN2SDMV(folds1.train, folds2.train,
           glavni parametri)
8:         rules[ ] ← cn2sd.run()
9:         for rule : rules do
10:            r.evaluiraj(fold.test)
11:        end for
12:        ruleCompSets.add(rules)
13:    end for
14:    return ruleCompSets
15: end function
```

Algorithm 10 Kompleksni višepogledni CN2SD traži podgrupe

```
1: function TRAŽIPODGRUPE
2:     ruleCompSet[ ]
3:     targets [ ] ← Odredi ciljne labele skupa podataka
4:     for target : targets do
5:         bestRuleComp ← traziPravilo (target, minConfidence, ruleComp-
           Set)
6:         if bestRuleComp != null then
7:             minMvScore ← bestRuleComp.mvScore · mvScoreFactor
8:         end if
9:         while bestRuleComp != null do
10:            ruleCompSet.add(bestRuleComp)
11:            bestRuleComp.cover(dataset, gamma, presjek)
12:            bestRuleComp ← traziPravilo(target, minConfidence, ruleCompSet)
13:        end while
14:    end for
15:    return ruleCompSet
16: end function
```

Algorithm 11 CN2SD nađi par pravila (kompleksni višepogledni)

```
1: function NADIPARPRAVILA(target, minConfidence, ruleCompSet, jednostavni uvjeti oba pogleda, queue (za evaluaciju))
2:     bestRuleComp ← null
3:     ruleListView1[ ]
4:     ruleListView2[ ]
5:     priorityQueueRuleComp = new PriorityQueue
6:     for jednostavan uvjet : jednostavni uvjeti prvog pogleda do
7:         rule ← new Rule(jednostavan uvjet, target)
8:         rule.evaluiraj
9:         ruleListView1.add(rule)
10:    end for
11:    for jednostavan uvjet : jednostavni uvjeti drugog pogleda do
12:        rule ← new Rule(jednostavan uvjet, target)
13:        rule.evaluiraj
14:        ruleListView2.add(rule)
15:    end for
16:    ruleListView1.sort()
17:    ruleListView2.sort()
18:    for i ← 0; i < startingRuleCompSize; i++ do
19:        for rule : ruleListView1 do
20:            rc = new ruleComp (rule, ruleListView2[i])
21:            queue.add(rc)
22:        end for
23:        for rule : ruleListView2 do
24:            rc = new ruleComp (ruleListView1[i], rule)
25:            queue.add(rc)
26:        end for
27:    end for
28:    ruleCompList = new Array (priorityQueueRuleComp)
29:    currentIteration ← 0
30:    while currentIteration < maxIterations do
31:        for i ← 0; i < beamSize; i++ do
32:            for jednostavan uvjet : jednostavni uvjeti prvog pogleda do
33:                validan ← ruleCompList[i].rule1.uvjet.isValid(jednostavan uvjet)
34:                if validan then
35:                    kompleksan uvjet ← ruleCompList[i].rule1.uvjet.dodajUvjet(jednostavan uvjet)
36:                    newRule ← new Rule(kompleksan uvjet, target)
37:                    spec(ruleCompList[i].rule1, newRule, jedan pogled)
38:                    rc(newRule, ruleCompList[i].rule2)
```

```
39:                     if (
40:                         rc.Jaccard>ruleCompList[i].Jaccard and
41:                         rc.intersection.size≥minIntersectionSize) then
42:                         newRuleComp = new ruleComp(rc)
43:                         queue.add(newRuleComp)
44:                     end if
45:                 end if
46:             end for
47:             Analogna for petlja za uvjete drugog pogleda (32-46)
48:         end for
49:         ruleCompList.clear
50:         ruleCompList.add(parove pravila s prioritetnog reda)
51:         for ruleComp : ruleCompList do
52:             pVal ← p-vrijednost para pravila
53:             if ruleCompList zadovoljava minScore, uvjete iz maina then
54:                 if bestRuleComp==null then
55:                     Provjera reduntantnosti ruleComp
56:                     if nije reduntantno then
57:                         bestRuleComp←ruleComp
58:                         break
59:                     end if
60:                 end if
61:                 if ruleComp.mvScore>bestRuleComp.mvScore then
62:                     Provjera reduntantnosti ruleComp
63:                     if nije reduntantno then
64:                         bestRuleComp←ruleComp
65:                         break
66:                     end if
67:                 end if
68:             end if
69:         end for
70:         currentIteration++
71:     end while
72:     return bestRuleComp
73: end function
```

Koncepciju ovog pristupa dizajnirao je mentor ovog diplomskog rada doc.dr.sc. Mihelčić koji je temeljen na tome da se modificira zrakasto pretraživanje CN2SD. Budući da mi imamo dva pogleda jednog skupa podataka, u funkciji *traziPravilo* (Algoritam 11) moramo inducirati pravila od svih jednostavnih

uvjeta, za oba pogleda zasebno, Algoritam 11 (3-7, 8-12), te ih analogno CN2SD sortirati prema *WRAcc*-u. Nadalje, potrebno je napraviti kartezijev produkt ta dva skupa pravila, no vremenska složenost bi u tom slučaju bila kvadratna, što nismo htjeli. Kako bi vremenska složenost ostala linearna, morali smo odrediti proizvoljan broj k najboljih početnih pravila (prema njihovom $mWRAcc$-u) jednog pogleda te napravili kartezijev produkt tih najboljih k pravila jednog pogleda sa svim početnim pravilima drugog pogleda i obratno. Tada je vremenska složenost ostala linearna, točnije $O(kn) = O(n)$. To nam ne garantira nalaženje najboljih parova početnih pravila, no i dalje je dobra heuristika s obzirom na to da nismo htjeli žrtvovati vremensku složenost. Prilikom kreiranja parova pravila, Algoritam 11 (17, 21), evaluiramo ih, Algoritam 11 (18, 22), tako da im odredimo *Multiview score*, *WRAccIs*, *mWRAccIs*, Jaccard, signifikantnost, pClassCond, pCond.
Nadalje, slično kao u originalnom CN2SD ulazimo u pretraživanje. Proširenje pravila se radi jednako, samo što u ovom slučaju imamo 2 zasebne petlje, Algoritam 11 (28-42, 43), koje dodaju svaka drugom pravilu u paru sve jednostavne uvjete odgovarajućeg pogleda. Opet, da bi uštedili na vremenu izvršavanja, implementirali smo funkciju specijalizacije , Algoritam 11 (33), vrlo sličnoj funkciji evaluiraj podgrupu (Algoritam 8) samo što umjesto da iterira po cijelom skupu podataka, da bi izračunali parametre para koristi pravilo na koje se dodaje uvjet budući da dodavanjem jednostavnog uvjeta možemo samo smanjiti broj primjera koje pravilo pokriva. Funkcija specijalizacije iterira po primjerima koje staro pravilo pokriva te gleda zadovoljava li novi uvjet koji dodajemo pravilu taj primjer. Dodali smo i minimalne uvjete na *Jaccard*, odnosno kako bi par pravila smatrali relevantnim, mora imati veći *Jaccard* u odnosu na par pravila prije dodavanja uvjeta, Algoritam 11 (36), kao i na broj primjera koje par pravila pokriva, Algoritam 11 (37). Dalje smo dodali računanje p-vrijednosti podgrupe, Algoritam 11 (48), što je koristilo kao dodatna provjera značajnosti podgrupe isto kao i provjeravanje reduntantnosti para pravila, Algoritam 11 (51, 58). Time smo eliminirali slučaj u kojem algoritam vraća dva ista para pravila. Provjera je izvedena na način da ako postoji identičan par pravila u već spremljenoj listi podgrupa, tada se gleda sljedeća najbolja podgrupu.
Također, kako bi izbjegli probleme sa prostornom složenošću, izmijenili smo implementaciju višedretvene evaluacije i zrake. Umjesto liste, koristili smo prioritetni red veličine zrake, na kojem se na glavi nalazi najgora podgrupa (što se tiče mjere dobrote) te kada na evaluaciju stigne bolja podgrupa, izbacuje se najgora podgrupa, a dodaje nova. Tako je izbjegnuto spremanje svih generiranih parova u listu i sortiranje iste budući da prioritetni red radi to za nas.

3 Analiza vremenske složenosti

3.1 CN2SD

CN2SD za računanje jednog pravila prvo izračuna početna pravila koja sadržavaju po jedan jednostavan uvjet tako da iterira po svim jednostavnim uvjetima ($O(u)$). U najgorem slučaju skup podataka za svaki atribut sadrži $2n$ jednostavnih uvjeta, odnosno kada bi svaki atribut skupa za svaki primjer imao sve različite vrijednosti. Tada je u najgorem slučaju iteriranje po svim uvjetima u $O(a2n) = O(an)$ gdje a označava broj atributa. Pravilo je implementirano kao klasa, koja sprema vrijednosti pravila u $O(1)$. Za evaluiranje iterira po svim primjerima skupa podataka $O(n)$, a svaka operacija unutar pojedine iteracije evaluiranja je vremenske složenosti $O(1)$. Zaključujemo da se početna pravila induciraju u $O(an^2)$. U nastavnku se sortira dobivena lista početnih pravila u najgorem vremenu $O(n \cdot logn)$ iz razloga što java koristi *mergesort* za sortiranje objekata. Nakon toga se konstruira podlista najboljih pravila koristeći metodu *subList* kojoj je vremenska složenost $O(1)$. Slijedi *while* petlja koja iterira unaprijed određenih $maxIterations$ puta, što je konstanta. Unutar vanjske *while* petlje ulazimo u for petlju koja se izvrši također unaprijed određen $beamSize$ puta, što je opet konstanta. Slijedi dodavanje uvjeta pravilima, unutar for petlje koja se izvršava u vremenu $O(an)$ (iterira po svim uvjetima kojih, u najgorem slučaju, ima $2an$). Provjera validnosti uvjeta se odvija u $O(1)$. Dodavanje uvjeta i kreiranje novog pravila se također odvija u $O(1)$. Dodavanje pravila u listu pravila je u $O(1)$. Evaluacija se odvija u $O(n)$. Nakon što algoritam iterira po svim pravilima na zraci, lista pravila se sortira. U listi se u najgorem slučaju (kad su svi uvjeti validni) nalazi $beamSize \cdot n$ pravila. Sortiranje takve liste je najgore u $O(beamSize \cdot nlog(beamSize \cdot n))$. Budući da je beamSize konstanta, sortiranje je u $O(n \cdot logn)$ Nakon toga je *subList* opet u $O(1)$. Provjera dobrote pravila je u $O(1)$.

Možemo zaključiti da induciranje jednog pravila ima najgoru i prosječnu vremensku složenost u $O(an^2) + O(n \cdot logn) + O(an^2) + O(n \cdot logn)$ što je zapravo u $O(an^2)$.

Funkciju *traziPravilo* (Algoritam 6) poziva vanjska funkcija *traziPodgrupe* (Algoritam 2) koja prvo odredi ciljne labele skupa podataka, to radi u vremenu $O(n)$ budući da iterira kroz sve primjere i provjerava njihove ciljne labele u $O(1)$. Dalje se izvršava for petlja za svaku vrijednost ciljne labele. Zove se funkcija *traziPravilo* u $O(an^2)$, postavlja se minimalni $WRAcc$ u $O(1)$ te se dalje ide u *while* petlju koja se izvršava dok god *traziPravilo* vraća pravilo. Zbog načina pronalaženja pravila uz smanjivanja težine važnosti primjera u svakoj iteraciji, *while* petlja će se izvršiti konstantan broj puta

(dok ne pretraži cijeli prostor primjera). Broj primjera i broj jednostavnih uvjeta nije povezan s brojem izvršavanja te petlje. Nadalje, u svakoj se iteraciji dodaje pravilo u listu u vremenu $O(1)$, pokriju elementi funkcijom *težinskoPokrivanje* (Algoritam 4) koje iterira for petljom po svim primjerima skupa u vremenu $O(n)$ te su sve operacije unutar petlje u $O(1)$. Nakon pokrivanja zove *traziPravilo*.

Možemo zaključiti da je složenost funkcije *traziPodgrupe* u $O(c(an^2 + k(an^2 + n))) = O((an^2 + k(an^2))) = O(kan^2) = O(an^2)$.

Funkcija *main* (Algoritam 1) napravi stratificirane uzorke u $O(n)$. Slijedi *for* petlja koja u $xValidationFolds$ iteracija zove funkciju *traziPodgrupe* na svakom uzorku. Zatim se svako od r pronađenih pravila evaluira u $O(n)$ te se dodaje u listu svih pravila u $O(1)$.

Za kraj možemo zaključiti da je najgora i prosječna vremenska složenost CN2SD u $O(an^2)$.

3.2 Naivni višepogledni CN2SD

Naivni višepogledni CN2SD ima jednaku vremensku složenost za induciranje jednog pravila kao CN2SD, odnosno u $O(an^2)$. Također koristi funkciju *traziPodgrupe* od CN2SD, odnosno u $O(an^2)$. Razlika je u funkciji *main* (Algoritam 7). Radi sve što i *main* od CN2SD, samo dva puta, jednom za svaki pogled te na kraju računa kartezijev produkt pravila prvog i drugog pogleda. Iznad smo utvrdili da algoritam vraća konstantan broj pravila tako da taj dio ne mijenja složenost.

Možemo zaključiti da je najgora i prosječna vremenska složenost naivnog višepoglednog CN2SD u $O(an^2)$.

3.3 Kompleksni višepogledni CN2SD

Funkcija *nađiParPravila* u višepoglednom slučaju počinje računanjem svih početnih pravila koja se sastoje od jednog jednostavnog uvjeta, ali za dva pogleda. Tada je vremenska složenost tog koraka i dalje linearna, odnosno u $O(2an) = O(an)$. Konstrukcija pravila je u $O(an)$, međutim konstrukcija i evaluacija početnih pravila, odnosno indukcija početnih pravila, je u $O(an^2)$. Sada sortiramo dvije liste početnih pravila umjesto jedne, odnosno sortiranje je u $O(2n \cdot logn) = O(n \cdot logn)$. Sada, kod stvaranja parova pravila, umjesto koristeći kartezijev produkt skupova početnih pravila svakog pogleda koji bi imao kvadratnu složenost (svako pravilo prvog pogleda spajamo sa svakim pravilom drugog pogleda, odnosno u $O(n^2)$), da bi sačuvali linearnu složenost tog koraka (bez evaluacije), uveli smo konstantu $startingRuleCompSize$ koja

određuje koliko najboljih pravila jednog skupa će se sparivati sa svakim pravilom iz drugog skupa. Tada je složenost tog koraka, uračunajući evaluaciju, u $O(startingRuleCompSize(n^2+n^2)) = O(n^2)$. Bez ove heuristike, složenost ovog koraka, uračunajući evaluaciju, bila bi u $O(n^3)$. Ovdje smo implementirali prioritetni red kojim smo izbjegli sortiranje pravila. Svaki par pravila koji dođe na evaluaciju se ubacuje u prioritetni red ako za njega ima mjesta, odnosno na glavi se drži par s najmanjom mjerom dobrote te ako par na evaluaciji ima veću mjeru dobrote od onog para na glavi, glava se izbaci te se dodaje novi element u red na odgovarajuće mjesto.

Dalje analogno CN2SD ulazimo u *while* petlju samo što ovdje moramo prvo svakom prvom pravilu iz para dodati sve jednostavne uvjete odgovarajućeg pogleda, zatim drugom. Vanjske *while* i for petlje su jednake CN2SD, dok su unutar njih dvije zaredane jednake do na pogled for petlje koje dodaju jednostavne uvjete. Također, nakon dodavanja uvjeta u pravilo, implementirali smo funkciju specijalizacije koja smanjuje vrijeme izvršavanja evaluacije jednog pravila iz $O(n)$ u $O(p_r)$ gdje p_r označava broj primjera koje pokriva pravilo kojem se dodaje jednostavni uvjet, što je i dalje u $O(n)$, ali svejedno je vrijeme izvršavanja manje. Dalje su dodani iznad spomenuti uvjeti koji su svi u $O(1)$ tako da je složenost tih for petlji $O(an(p_r + n)) \leq O(an(n + n)) = O(2an^2) = O(an^2)$. Dalje dodani uvjeti u odnosu na CN2SD su svi u $O(1)$ tako da ne mijenjaju vremensku složenost, osim provjere reduntantnosti. Provjera reduntantnosti najgorem slučaju iterira po cijelom skupu dosad induciranih parova pravila, označimo taj skup sa RC. Za svaki par pravila u RC provjerava je li u ovom koraku zasad najbolji par pravila reduntantan s njim. Budući da je broj uvjeta u pravilu ograničen s $maxIterations$, najgore vrijeme izvođenja ovog koraka je u $O(2maxIterationsRC)$, odnosno u $O(RC)$.

Možemo zaključiti da je najgora i prosječna složenost funkcije *nađiParPravila* u $O(an^2 + n \cdot logn + an^2 + an^2 + RC)$, odnosno u $O(an^2)$.

Dalje, funkcija *traziPodgrupu* i *main* su iste osim što se spremaju parovi pravila, a ne pravilo. Tada možemo zaključiti da je najgora i prosječna vremenska složenost višepoglednog CN2SD u $O(an^2)$.

4 Eksperiment

Kako bi testirali rad višepoglednog CN2SD algoritma, mentor Mihelčić je na raspolaganje dao skup podataka u kojem su primjeri ljudi koji imaju razne stupnjeve demencije ili Alzheimerovu bolest [11]. Jedan pogled kao atribute sadržava biološka obilježja dok drugi sadržava klinička obilježja pacijenata. Svi atributi su numerički, a ciljna labela je kategorijska sa skupom vrijednosti

{CN (kognitivno normalan, eng. cognitive normal), SMC (značajna smetnja pamćenja, EMCI (rana blaga kognitivna smetnja, eng. early mild cognitive impairment), LMCI (kasna blaga kognitivna smetnja, eng. late mild cognitive impairment), eng. significant memory concern), AD (dijagnosticirana Alzheimerova bolest, eng. Alzheimer disease)}.
U svrhu kvalitetnijeg testiranja algoritma našli smo još dva višepogledna skupa podataka.
Prvi korišten skup [12] sadržava filmove kao primjere. Jedan pogled kao atribute sadržava glumce te su atributi binarni, odnosno glumi li glumac u filmu ili ne. Drugi pogled kao atribute sadržava riječi kojima su filmovi opisani, odnosno atribut je opet binarni, opisuje li riječ film ili ne. Ciljna labela je kategorijska sa skupom vrijednosti od 17 žanrova.
Drugi korišten skup [13] sadržava novinske članke kao primjere. Oba pogleda sadržavaju kao atribute riječi te su atributi numerički, odnosno koliko odre-đenih riječi se nalazi u članku. U jednom pogledu se broje riječi članaka napisanih u BBC-u, a u drugom se broje riječi članaka te iste teme napisanih u Guardian-u. Ciljna labela je kategorijska sa skupom vrijednosti tipova članaka, odnosno jesu li članci politički, sportski, poslovni, zdravstveni, tehnološki ili zabavni.

Na sva tri skupa podataka pokrenuli smo algoritam koristeći određene parametre definirane u glavnom programu:

Parametar	Vrijednost
maxIterations (maksimalni broj jednostavnih uvjeta pravila)	10
beamSize (veličina zrake)	10
minSignificance (signifikantnost podgrupe - Fischerov test)	9.24
gamma (baza težinskog pokrivanja)	0.6
minIntersectionSize (najmanji broj primjera pokriven podgrupom)	10
startingRuleCompSize (broj najboljih početnih pravila jednog pogleda)	50
significanceBinDistThreshold (prag za p-vrijednost)	0.01

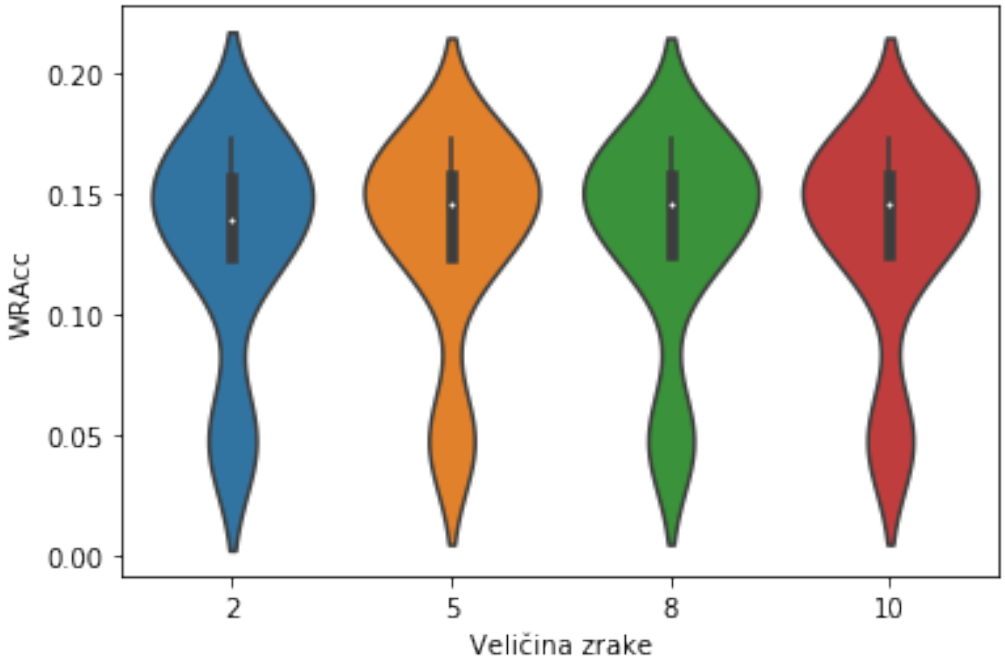
4.1 Prikaz podgrupa

Za sve navedene skupove podataka, pokrenuli smo originalni CN2SD, naivni višepogledni CN2SD te kompleksni višepogledni CN2SD na istima. Za prikaz rezultata korišteni su violinski dijagrami (eng. violin plot). Ovakvi grafovi su slični standardnim kutijastim dijagramima sa dodanim prikazom gustoće

distribucije elemenata.

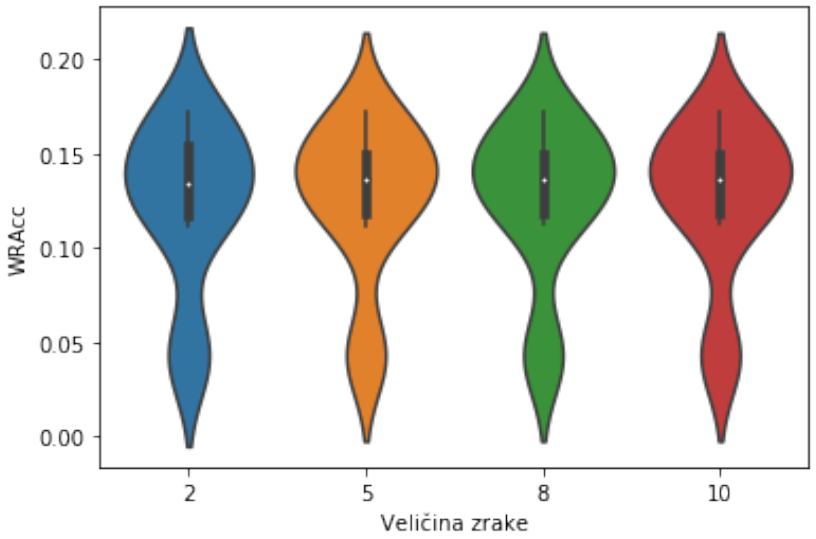
4.1.1 CN2SD

Za potrebe testiranja originalnog CN2SD algoritma spojili smo poglede višepoglednih skupova podataka u jedan te pokrenuli CN2SD na njima.



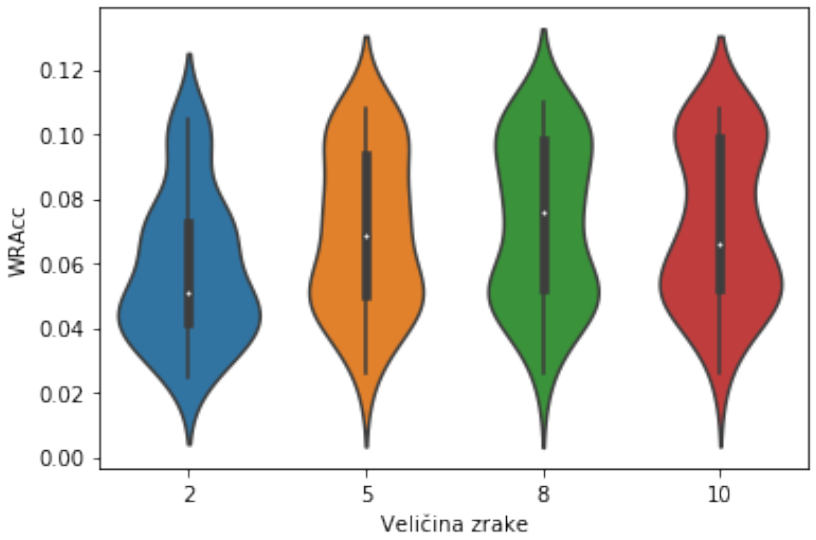
Slika 1: Alzheimer skup podataka na CN2SD

CN2SD je inducirao dobre podgrupe s visokim *WRAcc* vrijednostima. Medijan je osjetno bolji na prijelazu veličine zrake s dva na pet. Medijan *WRAcc*-a je na otprilike 0.14 te je najveća gustoća na otprilike 0.15. Dalje nema nekog znatnijeg poboljšanja povećanjem veličine zrake.



Slika 2: Testni Alzheimer skup podataka na CN2SD

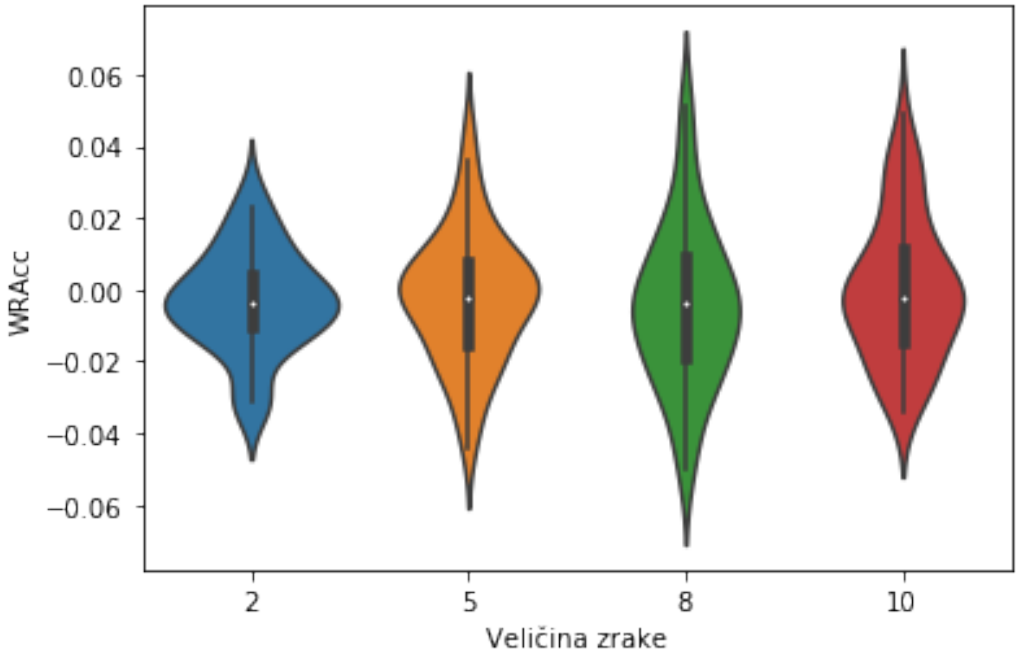
CN2SD pokazuje dobro prediktivno svojstvo na ovom skupu podataka. Medijan *WRAcc*-a je blizu onom cijelog skupa podataka (0.14). Najveća gustoća distribucije podgrupa je opet oko 0.15.



Slika 3: Skup podataka članaka na CN2SD

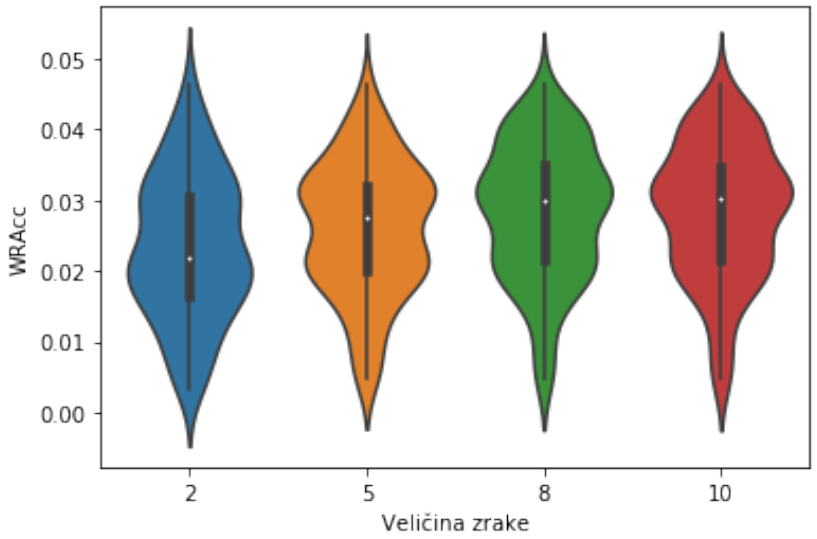
CN2SD je inducirao relativno dobre podgrupe s relativno visokim *WRAcc* vrijednostima. Medijan osjetno raste s prijelaza veličine zrake dva na pet

te pet na osam, no na prijelazu osam na deset znatno padne što se može dogoditi. Najbolji medijan na veličini zrake osam je malo manji od 0.08, a najveća gustoća distribucije podgrupa je na malo manje od 0.06.



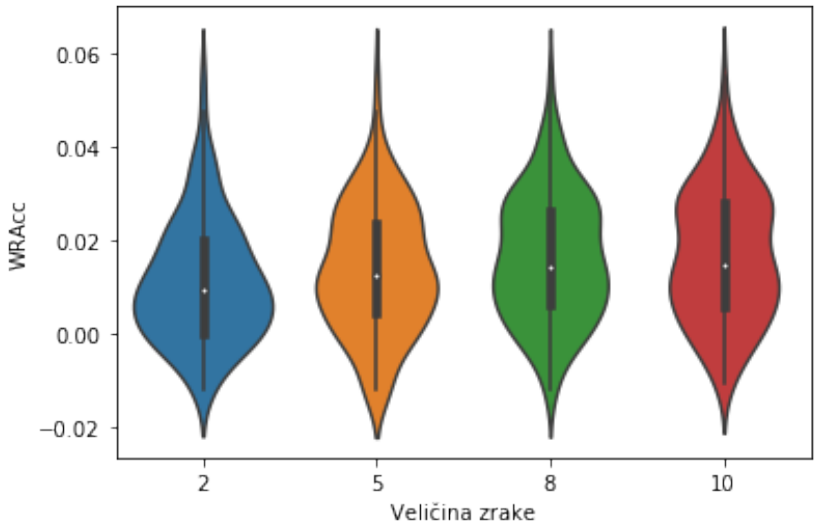
Slika 4: Testni skup podataka članaka na CN2SD

CN2SD pokazuje loše prediktivno svojstvo na ovom skupu podataka. Medijan *WRAcc*-a se vrti oko nule na svim veličinama zraka. Najveća gustoća distribucije podgrupa je također oko nule. Povećanjem veličine zrake se neznatno poboljšava gustoća distribucije podgrupa.



Slika 5: Skup podataka filmova na CN2SD

CN2SD je inducirao loše podgrupe s malim *WRAcc* vrijednostima. Medijan raste povećanjem veličine zrake, a vrijednost mu raste od malo više od 0.02 do oko 0.03. Najveća gustoća distribucije podgrupa također raste od oko 0.02 do oko 0.03.

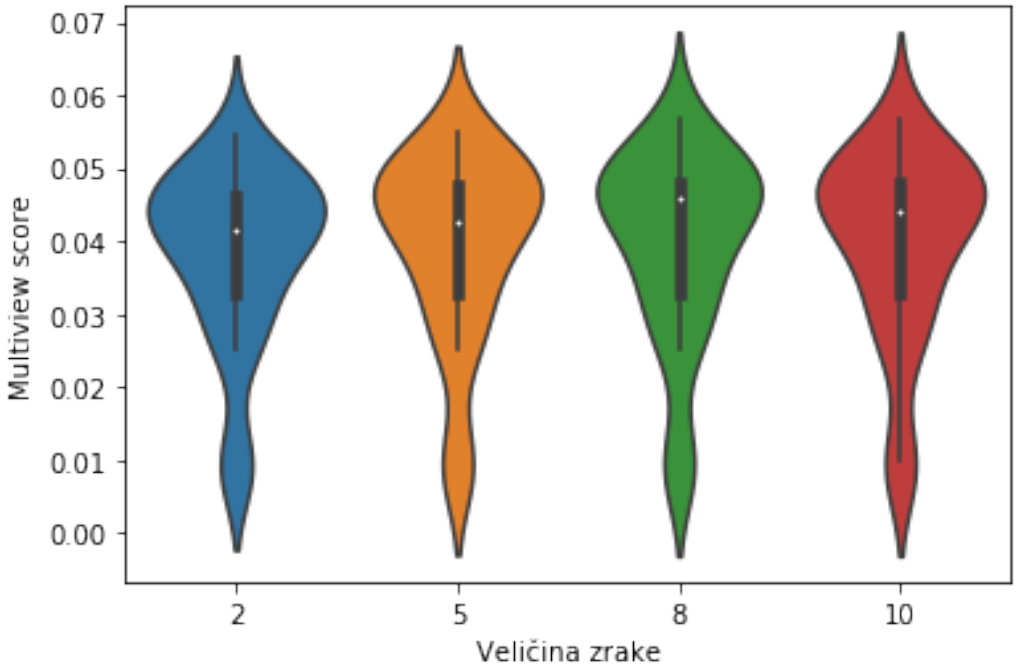


Slika 6: Testni skup podataka filmova na CN2SD

CN2SD pokazuje relativno dobro prediktivno svojstvo uzeći u obzir da su i inducirane podgrupe na tom skupu podataka imale malu vrijednost

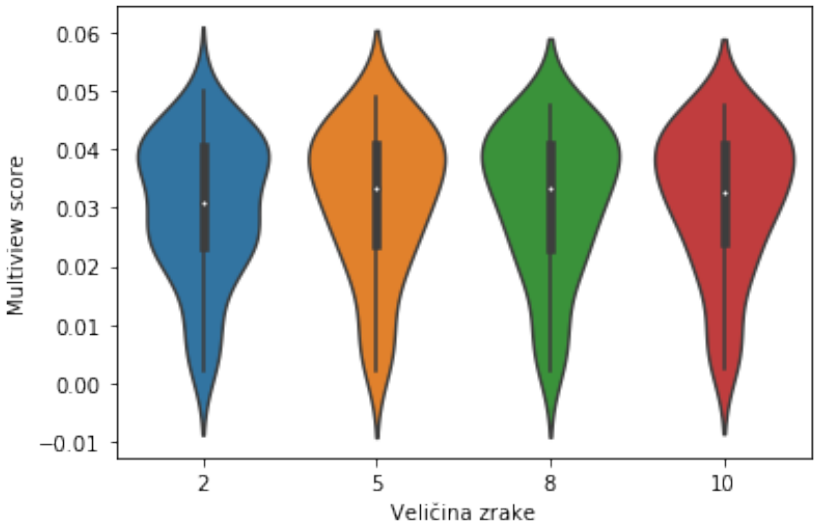
WRAcc-a. Medijan raste na prijelazu veličine zrake s dva na pet, kasnije raste neznatno malo. Vrijednost medijana je oko 0.01, kao što i najveća gustoća distribucije podgrupa.

4.1.2 Naivni višepogledni CN2SD



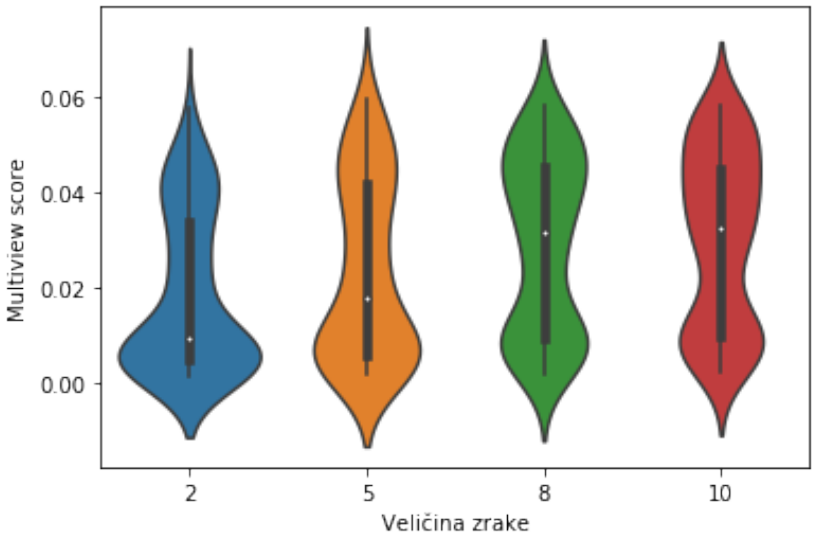
Slika 7: Alzheimer skup podataka na naivnom višepoglednom CN2SD

Naivni višepogledni CN2SD inducira relativno dobre podgrupe s relativno dobrim *Multiview score* vrijednostima. Medijan raste osjetno na prijelazu veličine zrake s pet na osam te pada s osam na deset. Vrijednost medijana je oko 0.045, kao i najveća gustoća distribucije podgrupa.



Slika 8: Testni Alzheimer skup podataka na naivnom višepoglednom CN2SD

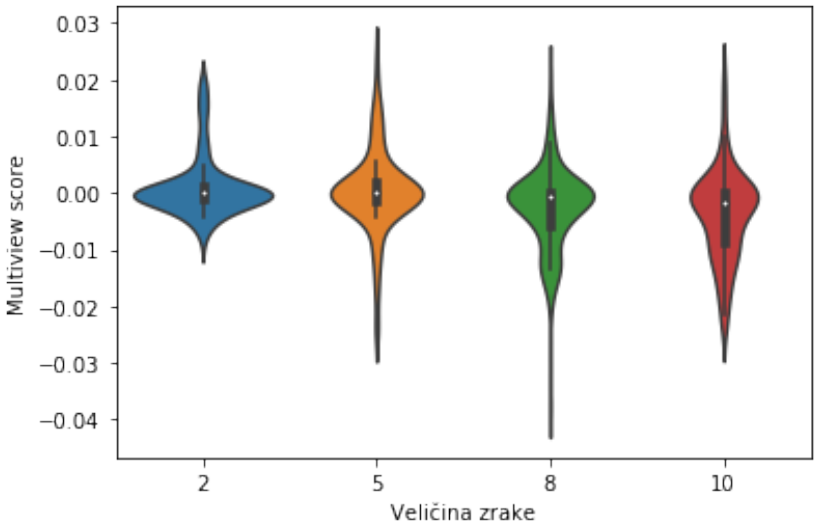
Naivni višepogledni CN2SD pokazuje dobro prediktivno svojstvo budući da je medijan za oko 0.01 manji od medijana na cijelom skupu podataka. Neznatno raste i pada, vrijednost mu je oko 0.03, a najveća gustoća distribucije podgrupa oko 0.04.



Slika 9: Skup podataka članaka na naivnom višepoglednom CN2SD

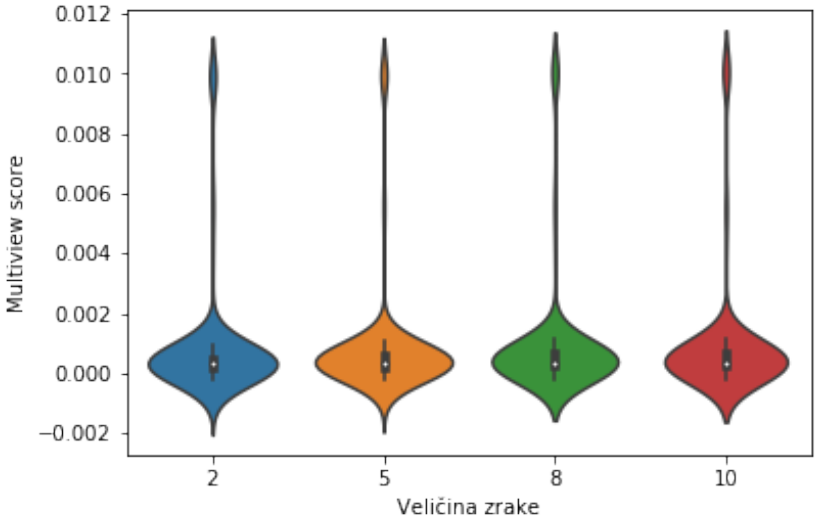
Naivni višepogledni CN2SD inducira loše podgrupe na veličini zrake dva, dok povećanjem zrake podgrupe postaju reativno dobre s najvećom gustoćom

distribucije oko 0.045. Medijan znatno raste na prijelazu veličine zrake s dva na pet, a još više s pet na osam. Vrijednost medijana dosegne oko 0.035.



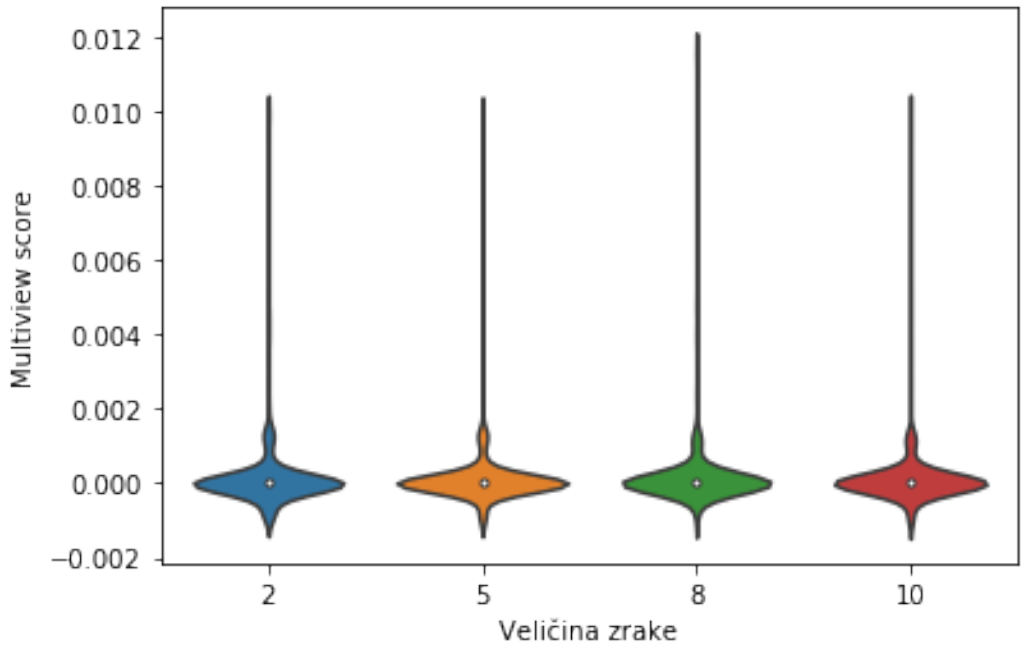
Slika 10: Testni skup podataka članaka na naivnom višepoglednom CN2SD

Naivni višepogledni CN2SD pokazuje loše prediktivno svojstvo, distribucija podgrupa ne postaje ništa bolja povećanjem veličine zrake. Vrijednost medijana je oko nule, kao i najveća gustoća distribucije podgrupa.



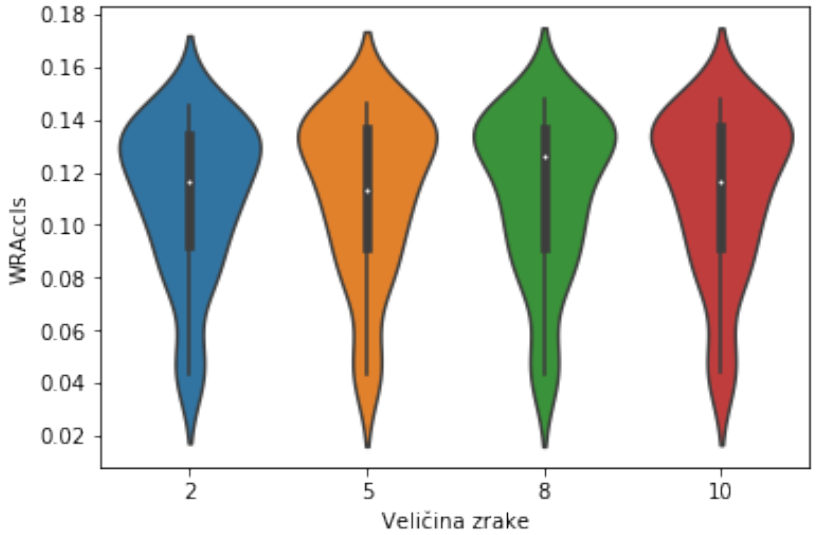
Slika 11: Skup podataka filmova na naivnom višepoglednom CN2SD

Naivni višepogledni CN2SD inducira jako loše podgrupe s niskom vrijednosti *Multiview score*-a koje ne postaju bolje povećanjem veličine zrake. Vrijednost medijana je oko nule, kao i najveća gustoća distribucije podgrupa.

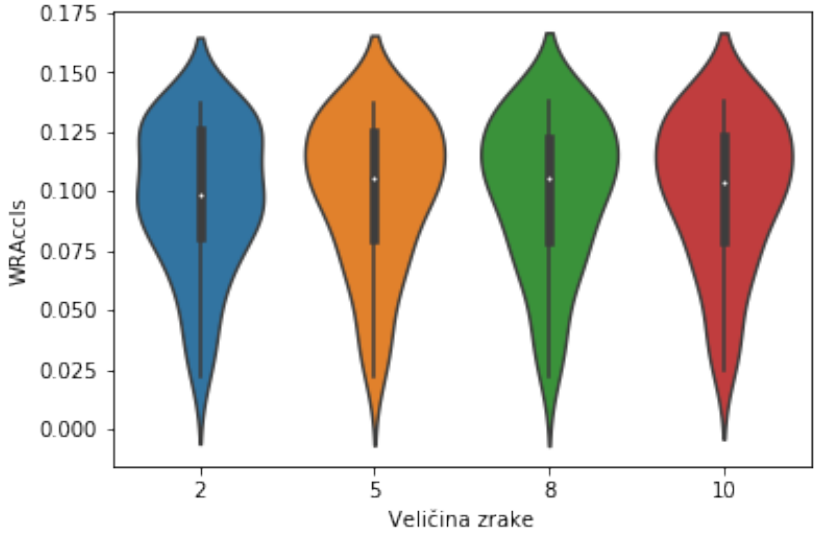


Slika 12: Testni skup podataka filmova na naivnom višepoglednom CN2SD

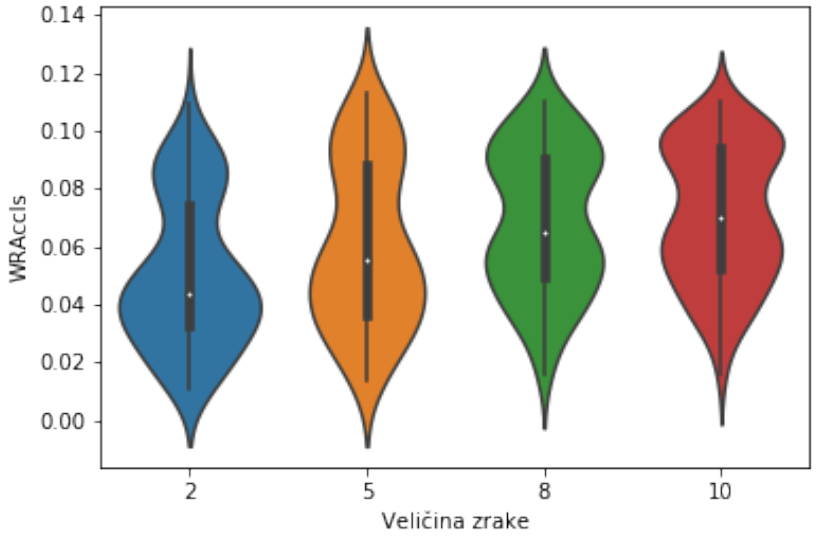
Slično cijelom skupu podataka, inducirane podgrupe imaju nisku vrijednost *Multiview score*-a, vrijednost medijana je oko nule, kao i najveća gustoća distribucije podgrupa.



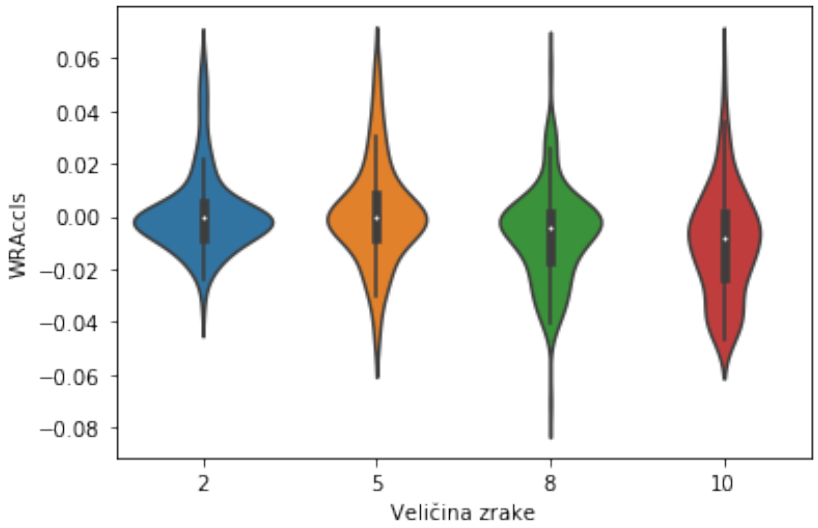
Slika 13: Alzheimer skup podataka na naivnom višepoglednom CN2SD - ocjena WRAccIs



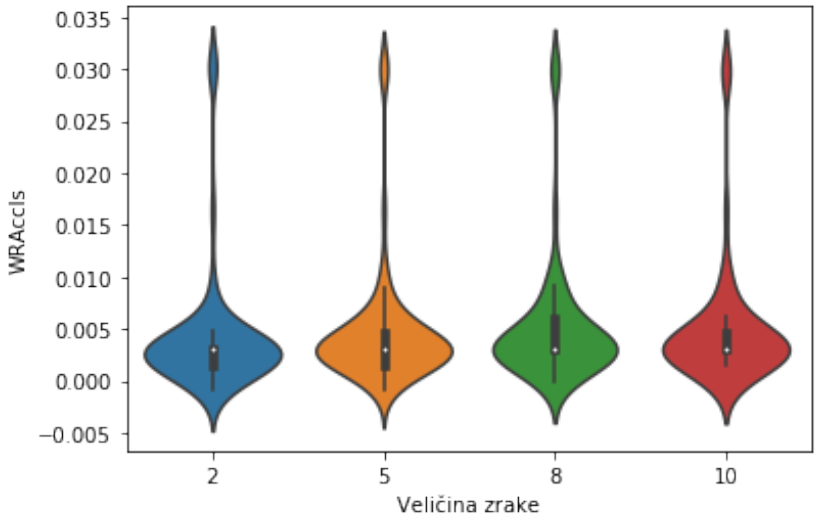
Slika 14: Testni Alzheimer skup podataka na naivnom višepoglednom CN2SD - ocjena WRAccIs



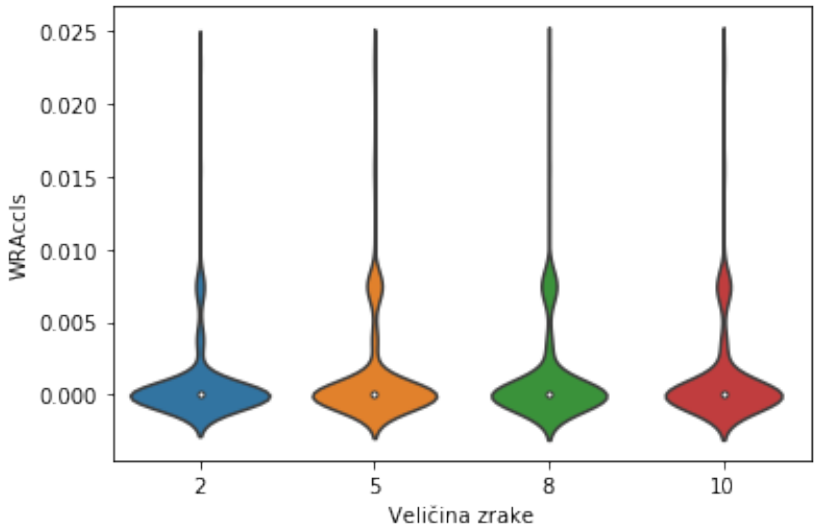
Slika 15: Skup podataka članaka na naivnom višepoglednom CN2SD - ocjena WRAccIs



Slika 16: Testni skup podataka članaka na naivnom višepoglednom CN2SD - ocjena WRAccIs



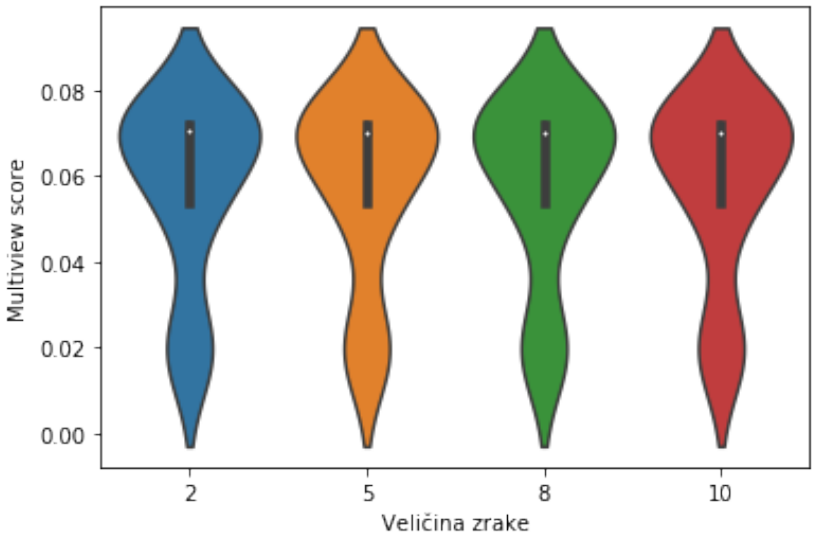
Slika 17: Skup podataka filmova na naivnom višepoglednom CN2SD - ocjena WRAccIs



Slika 18: Testni skup podataka filmova na naivnom višepoglednom CN2SD - ocjena WRAccIs

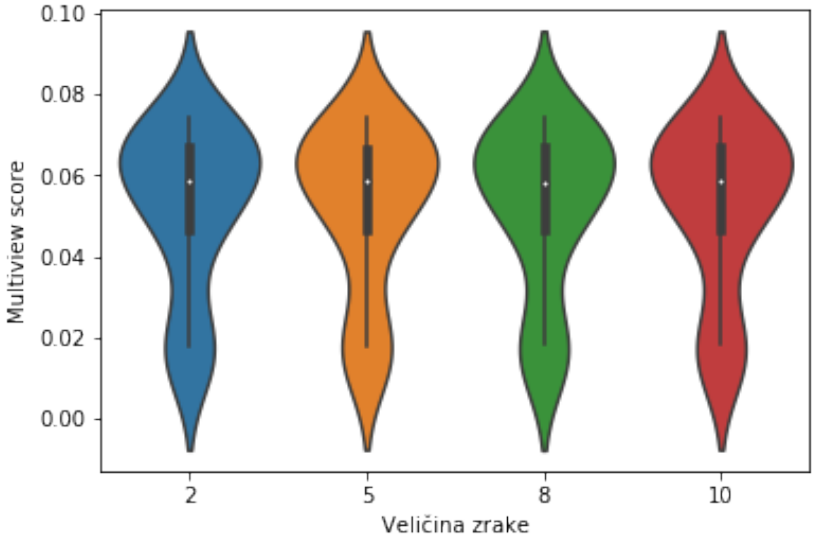
Možemo primijetiti da distribucija podgrupa na dijagramima na kojima se gleda odnos *WRAccIs*-a i veličine zrake odgovaraju onima iznad na kojima se gleda odnos *Multiview score*-a i veličine zrake te su objašnjenja grafova slična.

4.1.3 Kompleksni višepogledni CN2SD



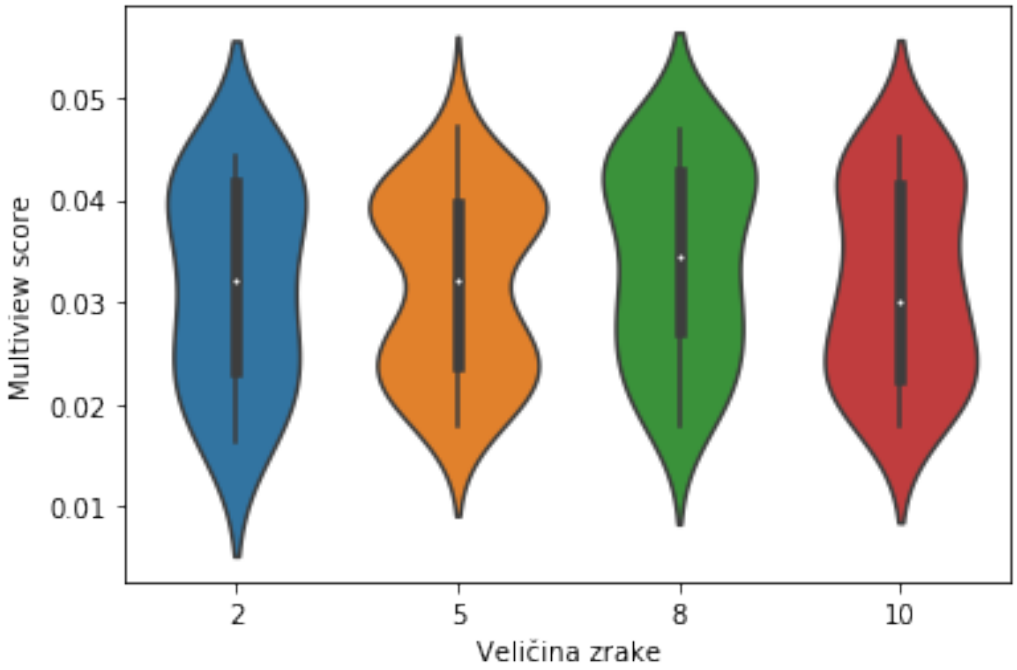
Slika 19: Alzheimer skup podataka na kompleksnom višepoglednom CN2SD

Kompleksni višepogledni CN2SD inducira jako dobre podgrupe s visokim vrijednostima *Multiview score*-a. Medijan ne raste povećanjem veličine zrake te mu je vrijednost oko 0.07, kao što je i najveća gustoća distribucije podgrupa.



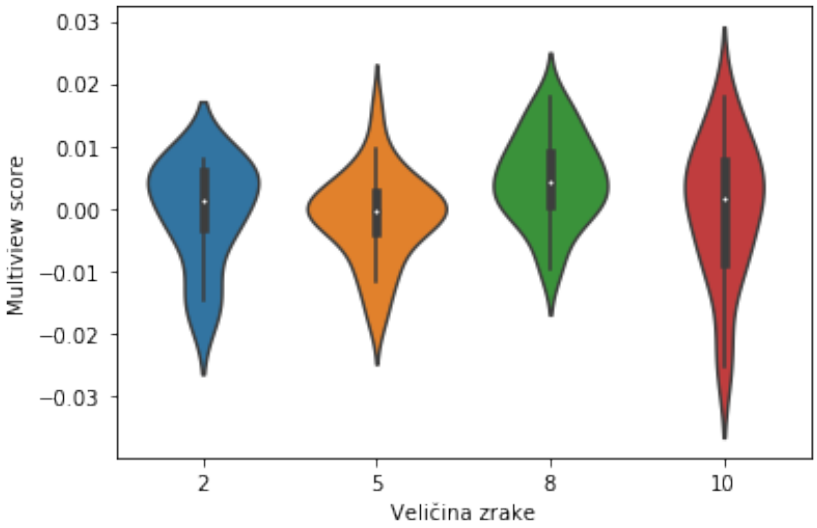
Slika 20: Testni Alzheimer skup podataka na na kompleksnom višepoglednom CN2SD

Kompleksni višepogledni CN2SD pokazuje dobro prediktivno svojstvo sa medijanom vrijednosti oko 0.06 koji se ne mijenja povećanjem veličine zrake. Najveća gustoća distribucije podgrupa je na oko 0.065.



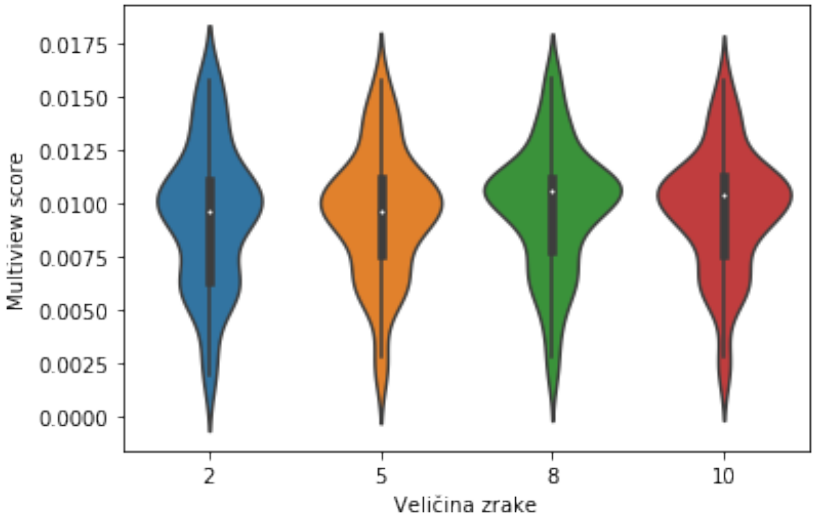
Slika 21: Skup podataka članaka na kompleksnom višepoglednom CN2SD

Kompleksni višepogledni CN2SD inducira relativno dobre podgrupe. Medijan ima neznačajan rast kroz prve tri veličine zrake, dok na veličini zrake 10 ima pad vrijednosti. Distribucija podgrupa je također bolja kod zrake veličine osam.



Slika 22: Testni skup podataka članaka na kompleksnom višepoglednom CN2SD

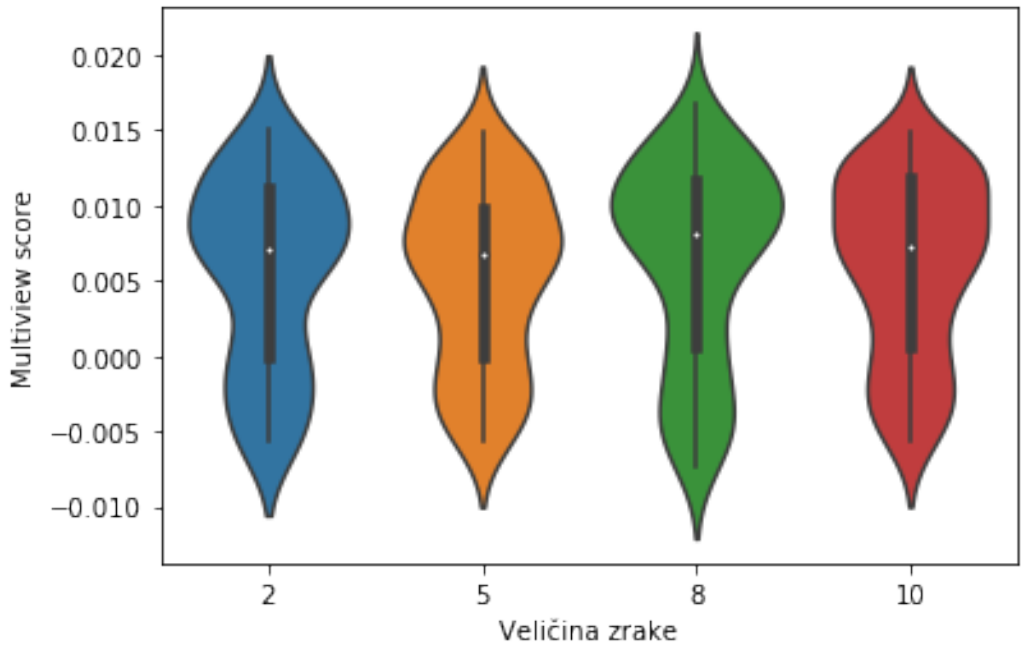
Vidljiva je promjena gustoće na svakoj veličini zrake, no opet su medijan i gustoća najbolji na veličini zrake osam. Prediktivnost je jako loša budući da je medijan i najveća gustoća distribucije podgrupa na oko 0.005.



Slika 23: Skup podataka filmova na kompleksnom višepoglednom CN2SD

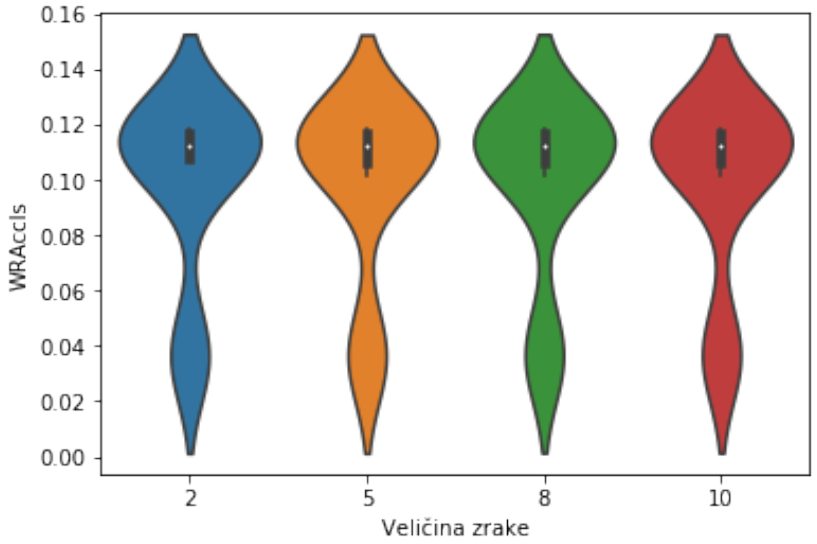
Kompleksni višepogledni CN2SD inducira relativno loše podgrupe sa neznačajnim rastom medijana kroz veličina zrake. Vidi se mala promjena u gustoći, slična

kod veličine zrake osam i deset, te je medijan kao i najveća gustoća distibucije podgrupa oko 0.01.



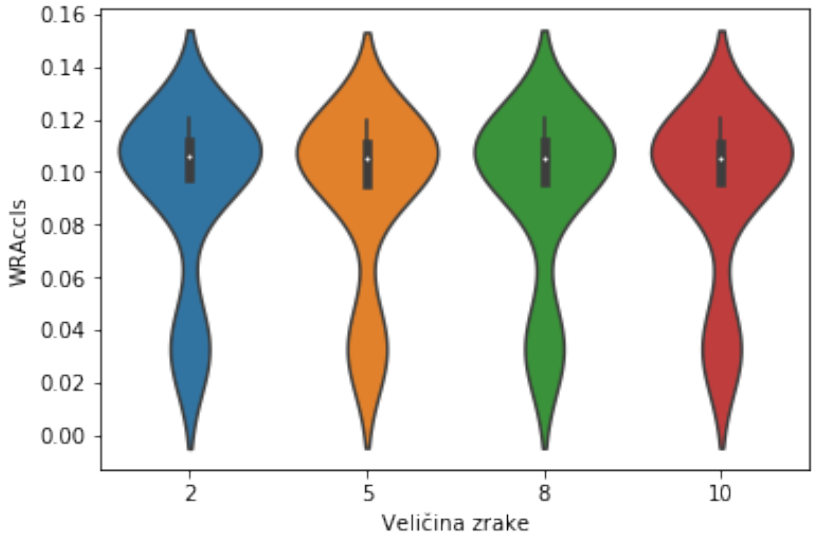
Slika 24: Testni skup podataka filmova na kompleksnom višepoglednom CN2SD

Slično cijelom skupu podataka, inducirane podgrupe imaju nisku vrijednost *Multiview score*-a, vrijednost medijana je oko 0.08. kao i najveća gustoća distribucije podgrupa. Kod zrake osam je najbolji medijan i distribucija podgrupa.



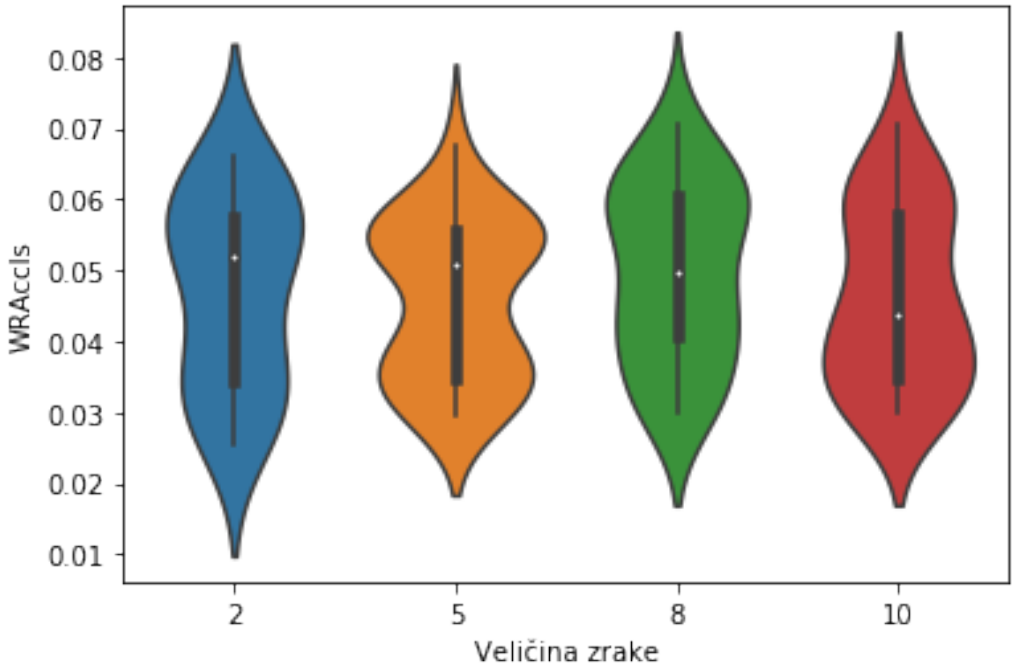
Slika 25: Alzheimer skup podataka na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

Kompleksni višepogledni CN2SD inducira jako dobre podgrupe s visokim vrijednostima *WRAccIs*-a. Medijan ne raste povećanjem veličine zrake te mu je vrijednost oko 0.105, kao što je i najveća gustoća distribucije podgrupa.



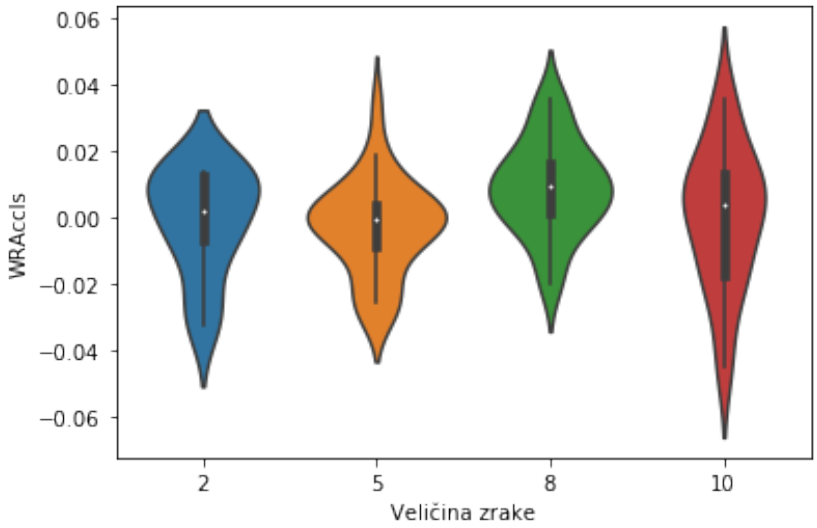
Slika 26: Testni Alzheimer skup podataka na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

Kompleksni višepogledni CN2SD pokazuje dobro prediktivno svojstvo sa medijanom vrijednosti oko 0.105 koji se ne mijenja povećanjem veličine zrake. Najveća gustoća distribucije podgrupa je na oko 0.011.



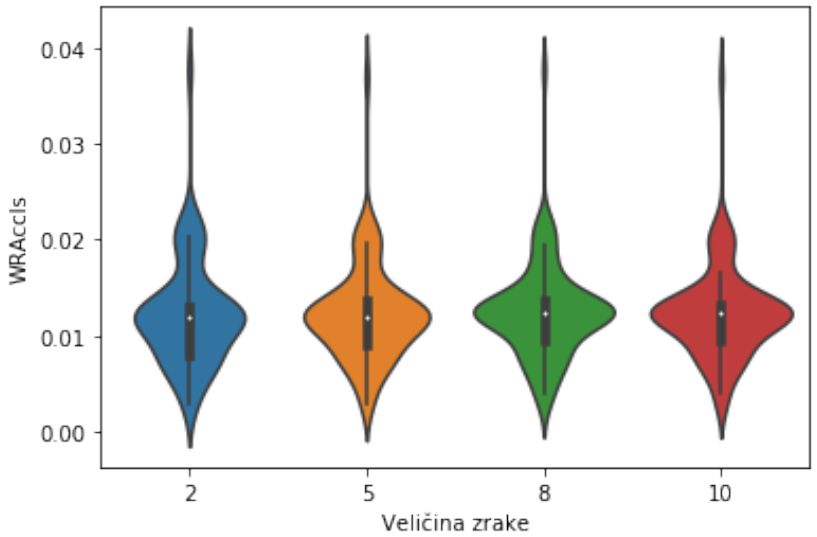
Slika 27: Skup podataka članaka na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

Kompleksni višepogledni CN2SD inducira relativno dobre podgrupe. Medijan ima neznačajan rast kroz prve tri veličine zrake, dok na veličini zrake 10 ima nekarakterističan pad vrijednosti. Distribucija podgrupa je također bolja kod zrake veličine osam.



Slika 28: Testni skup podataka članaka na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

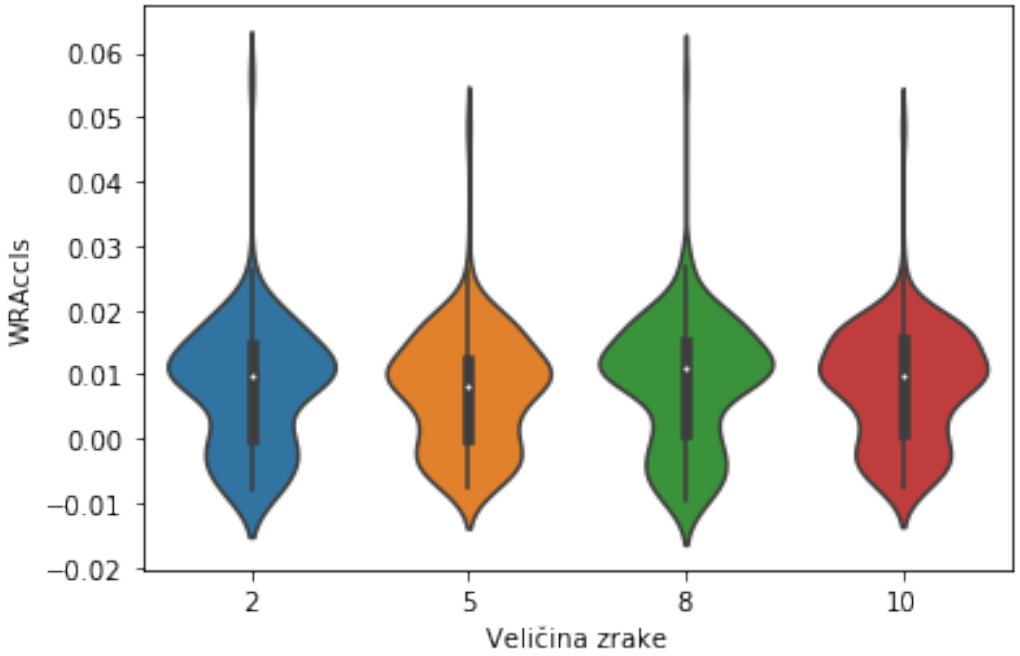
Vidljiva je promjena gustoće na svakoj veličini zrake, no opet su medijan i gustoća najbolji na veličini zrake osam. Prediktivnost je jako loša budući da je medijan i najveća gustoća distribucije podgrupa na oko 0.01.



Slika 29: Skup podataka filmova na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

Ovaj graf se razlikuje od odgovarajućeg s ocjenom *Multiview score*. Pod-

grupe su koncentrirane oko 0.013 s medijanom također oko 0.013. No, ovdje imamo mali uzorak podgrupa s višim *WRAccIs*-om, oko 0.04.



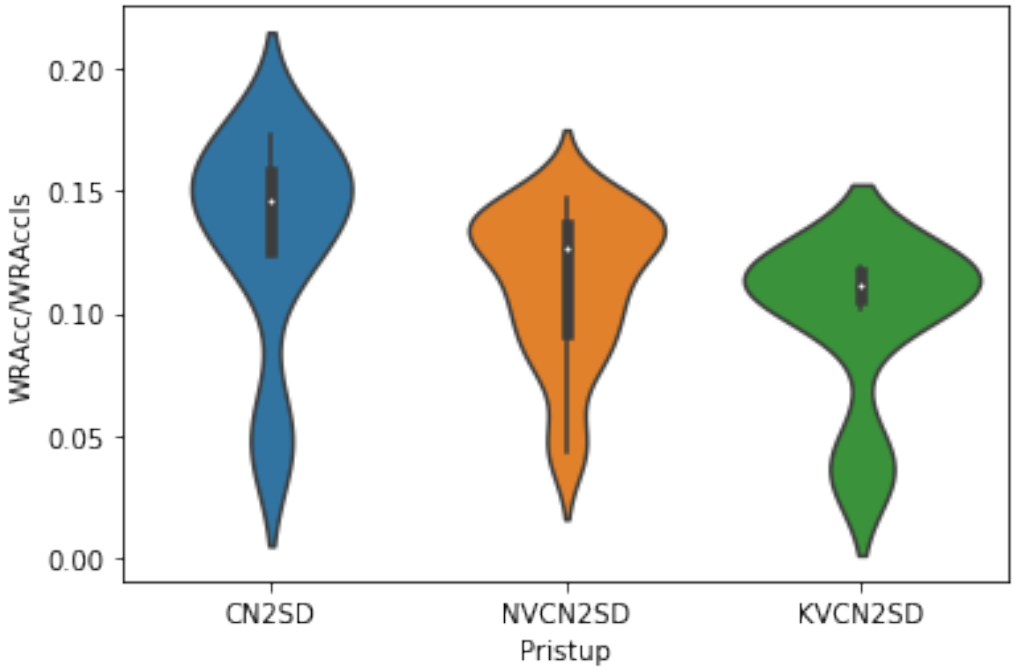
Slika 30: Testni skup podataka filmova na kompleksnom višepoglednom CN2SD - ocjena WRAccIs

Ovaj graf slično kao na cijelom skupu podataka ima medijan i najveću gustoću distribucije podgrupa oko 0.01 te poneke neobično visoke vrijednosti *WRAccIs*-a oko 0.06.

4.2 Usporedba pristupa koristeći zraku veličine osam

Budući da je vidljivo iz prethodnog poglavlja da su najbolji rezultati kad je veličina zrake jednaka 8, za navedenu veličinu zrake ćemo proučiti tri pristupa traženju podgrupa. Također ćemo provesti statistički jednostrani Mann-Whitney U test između pristupa koristeći funkciju *mannwhitneyu* scipy biblioteke.

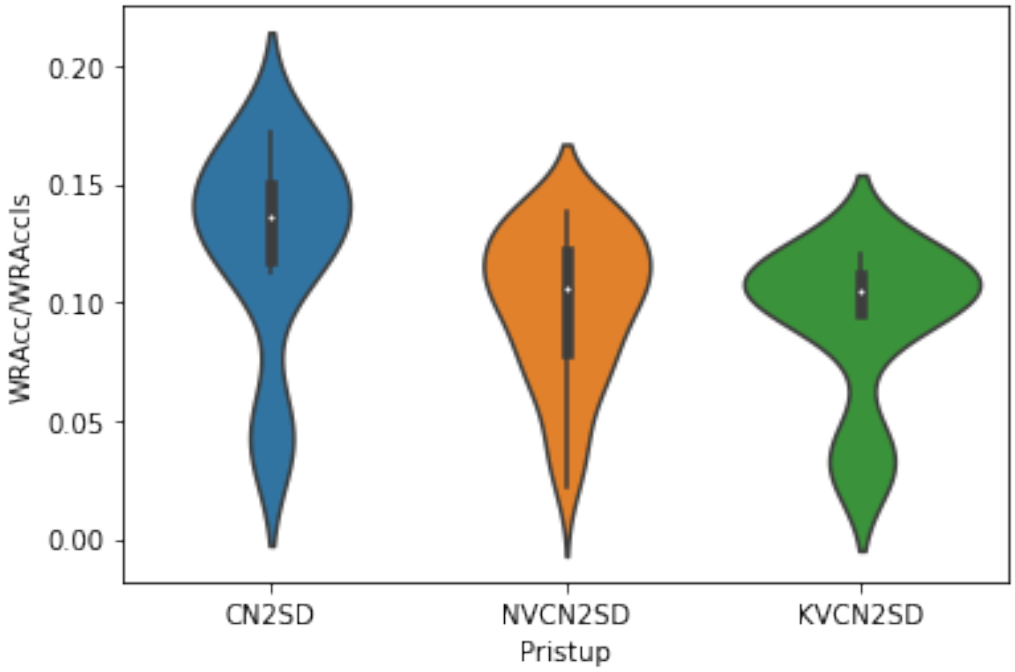
4.2.1 Skup podataka Alzheimer



Slika 31: Alzheimer skup podataka

Lako je vidljivo da originalni CN2SD ima najbolji medijan te distribuciju podgrupa kada se gleda *WRAcc/WRAccIs*. Zatim slijedi naivni višepogledni CN2SD pristup sa malo manjim medijanom te malo lošijom distribucijom podgrupa. Logično je da naivni pristup ima bolju distribuciju podgrupa od kompleksnog kada se gleda *WRAccIs* budući da nema nikakvih ograničenja na *Jaccard* indeks, koji govori o podudarnosti pravila u paru. Višepogledni pristup nam daje malo lošiju distribuciju i medijan od naivnog pristupa u ovom slučaju.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2209.0, pvalue=6.118828503267118e-10)
MannWhitneyUOneSided(CN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=1632.0, pvalue=0.013369141596421483)
MannWhitneyUOneSided(NVCN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=788.0, pvalue=0.9997020091335667)

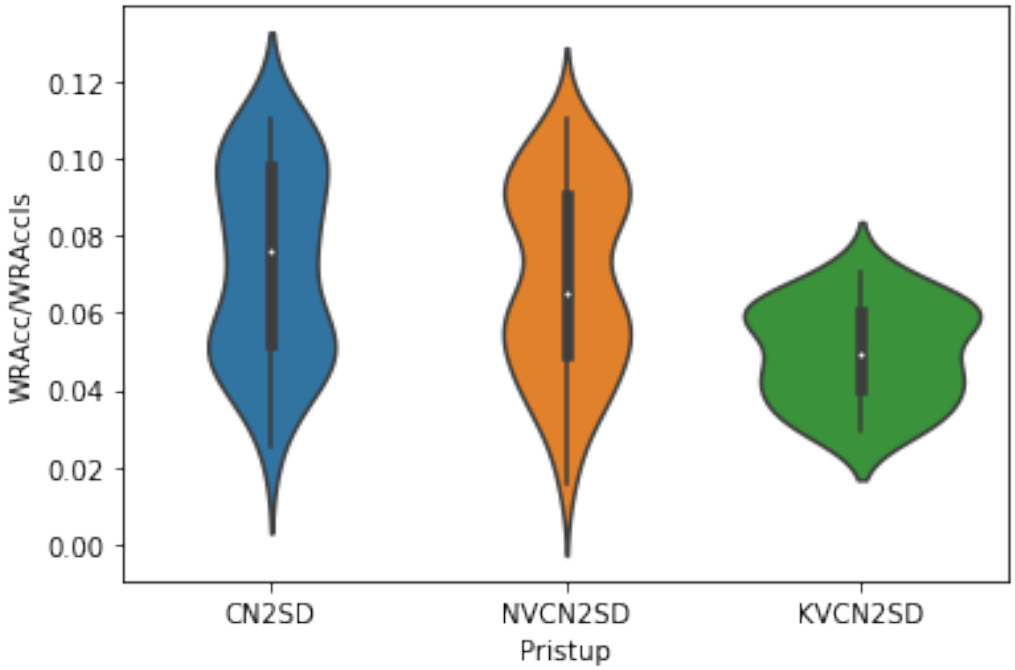


Slika 32: Testni Alzheimer skup podataka

Slijedi dobra prediktivnost budući da ocjena na testnom skupu ne odskače od ocjene na cijelom skupu podataka te možemo povući zaključak isti onom na cijelom skupu podataka. Možemo vidjeti da kompleksni pristup ima bolju prediktivnost budući da se izjednačio s naivnim.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2224.5, pvalue=3.188473089298856e-10)
MannWhitneyUOneSided(CN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=1623.0, pvalue=0.015578891876722196)
MannWhitneyUOneSided(NVCN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=702.0, pvalue=0.9999695102902699)

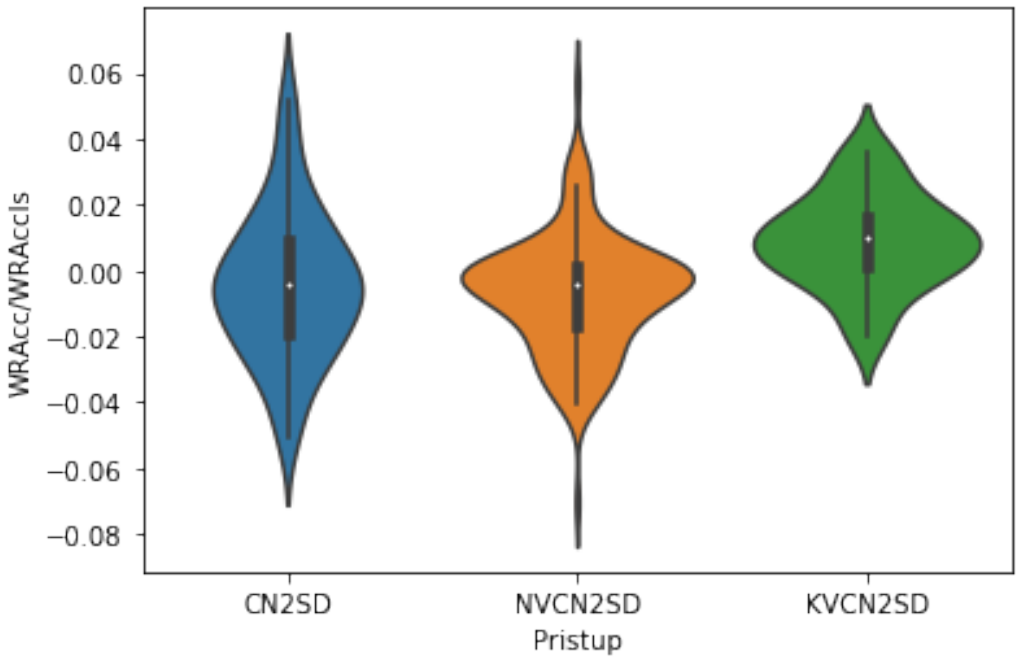
4.2.2 Skup podataka članaka



Slika 33: Skup podataka članaka

Opet slično Alzheimer skupu, najbolji je CN2SD pristup, zatim naivni te na kraju kompleksni pristup. No, ponavljam, naivan pristup nema ograničenje na *Jadcard indeks*, odnosno ne daje dosta podudarne parove pravila.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=15459.0, pvalue=5.132128456241944e-22)
MannWhitneyUOneSided(CN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=12320.0, pvalue=1.095190394954519e-06)
MannWhitneyUOneSided(NVCN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=2751.0, pvalue=1.0)

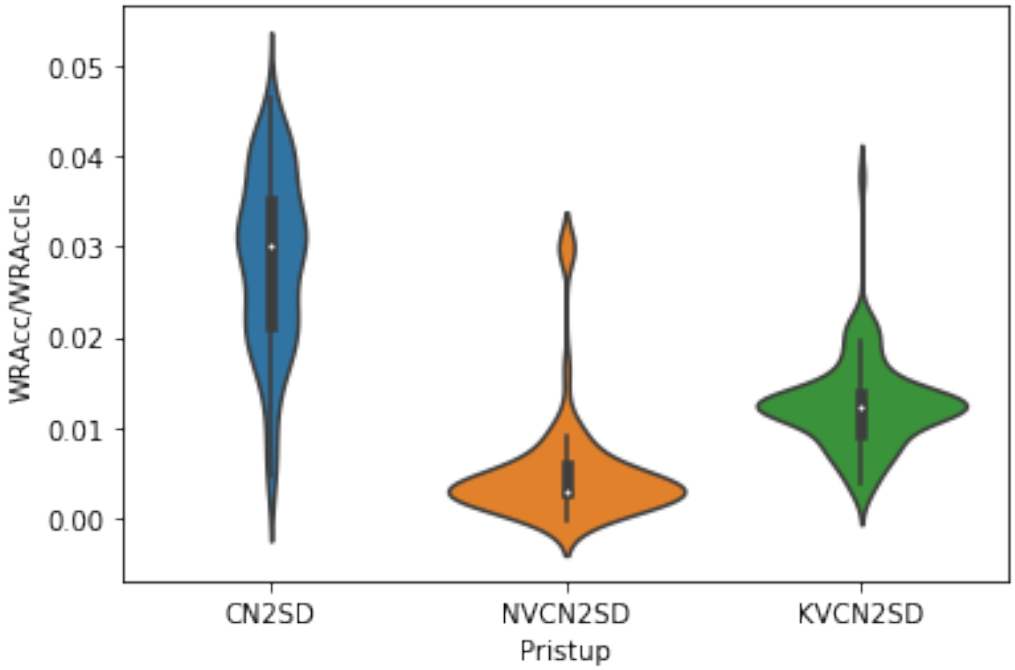


Slika 34: Testni skup podataka članaka

Slijedi loša prediktivnost na ovom skupu podataka što se vidi iz toga da je medijan *WRAcc/WRaccIs*-a na testnom skupu oko nule, no zanimljivo je da je na ovom skupu kompleksni pristup vidljivo bolji od originalnog i naivnog CN2SD.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=15339.0, pvalue=3.0077012736583384e-21)
MannWhitneyUOneSided(KVCN2SD, CN2SD)=
MannwhitneyuResult(statistic=15494.0, pvalue=3.04378199778813e-22)
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=17684.0, pvalue=5.740561606502885e-39)

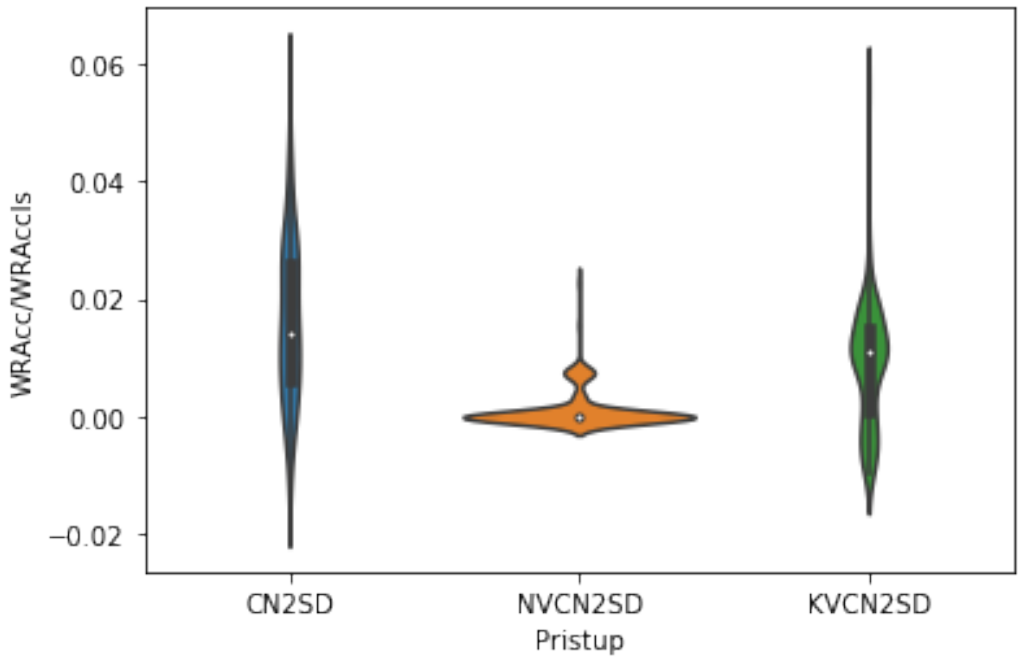
4.2.3 Skup podataka filmova



Slika 35: Skup podataka filmova

CN2SD ovdje daje puno bolje podgrupe od višepoglednih pristupa. Naivan pristup inducira jako loše podgrupe s medijanom oko nule, ali opet je zanimljivo da je kompleksni pristup bolji s medijanom oko 0.01 te boljom distribucijom podgrupa.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=74528.0, pvalue=9.629373594509977e-87)
MannWhitneyUOneSided(CN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=61461.0, pvalue=3.3306561675883997e-37)
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=70811.0, pvalue=1.723776491470655e-70)



Slika 36: Testni skup podataka filmova

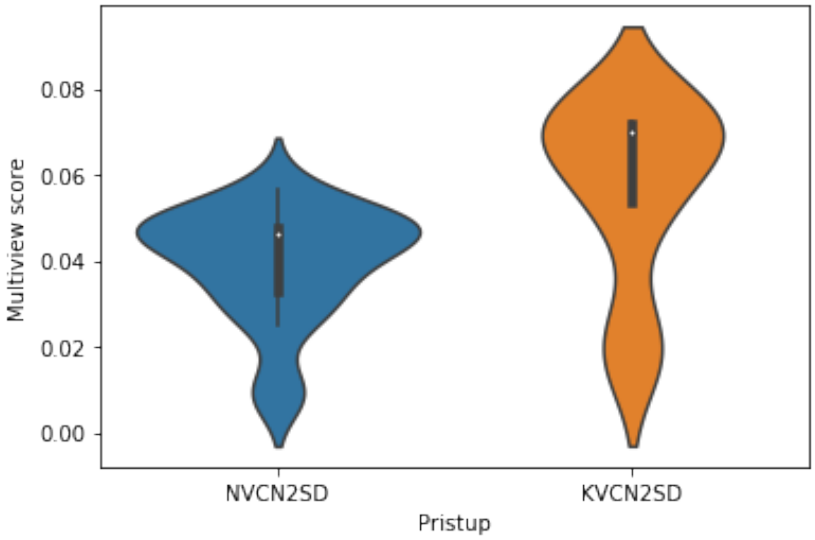
Ovdje je još manja razlika između CN2SD i kompleksnog CN2SD. Kompleksni pristup je i dalje bolji od naivnog te je prediktivnost dobra budući da je distribucija *WRAcc/WRAccIs* na testnom skupu slična onom na cijelom skupu podataka.

MannWhitneyUOneSided(CN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=69785.0, pvalue=6.749222741358724e-70)
MannWhitneyUOneSided(CN2SD, KVCN2SD)=
MannwhitneyuResult(statistic=55334.0, pvalue=2.660185578459048e-21)
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=62381.0, pvalue=3.925483959517429e-42)

4.3 Usporedba višepoglednih pristupa koristeći zraku veličine osam

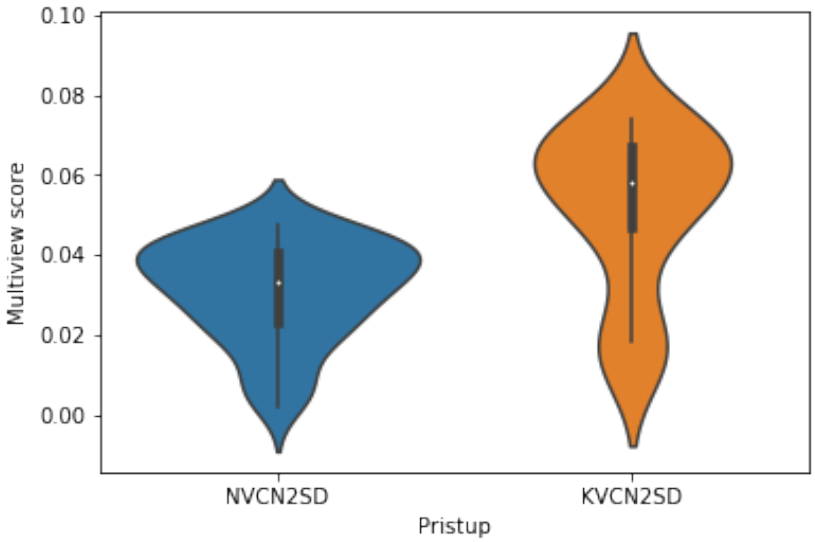
U ovom potpoglavlju ćemo usporediti *Multiview score* i *Jaccard indeks* višepoglednih pristupa. Iskoristili smo veličinu zrake osam budući da daje najbolje rezultate. Provesti ćemo statistički jednostrani Mann-Whitney U test između višepoglednih pristupa koristeći funkciju *mannwhitneyu* scipy biblioteke.

4.3.1 Skup podataka Alzheimer



Slika 37: Alzheimer skup podataka

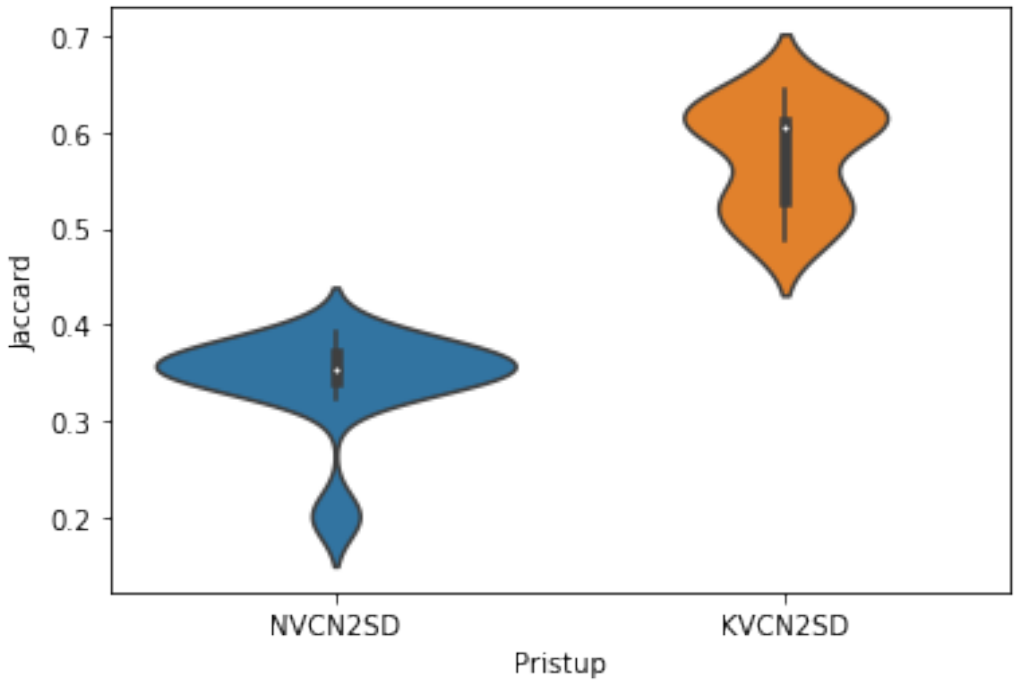
Kompleksni višepogledni CN2SD daje znatno bolju distribuciju podgrupa uz medijan 0.07 dok je medijan naivnog pristupa oko 0.045.
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2356.0, pvalue=8.272208599082776e-13)



Slika 38: Testni Alzheimer skup podataka

Kompleksni višepogledni CN2SD daje znatno bolju distribuciju podgrupa uz medijan 0.06 dok je medijan naivnog pristupa oko 0.035. Prediktivnost je dobra budući da distribucije podgrupa ne odskaču previše od one na cijelom skupu podataka.

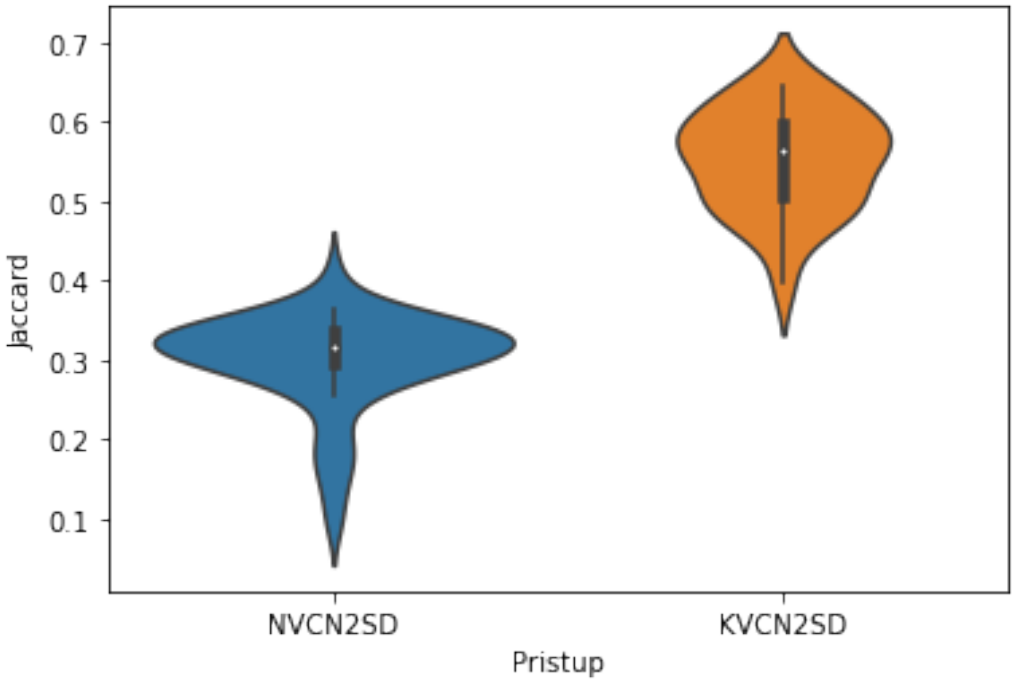
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2362.0, pvalue=6.190951287858802e-13)



Slika 39: Alzheimer skup podataka (Jaccard)

Gledanjem *Jaccard indeks*-a se vidi još bolja prednost kompleksnog pristupa. Vrijednost medijana je kod višepoglednog indeksa oko 0.6 dok je kod naivnog oko 0.35 što je ogromna razlika. Parovi pravila koji čine podgrupu su znatno više podudarni kod kompleksnog pristupa.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2601.0, pvalue=1.6500512708381197e-18)

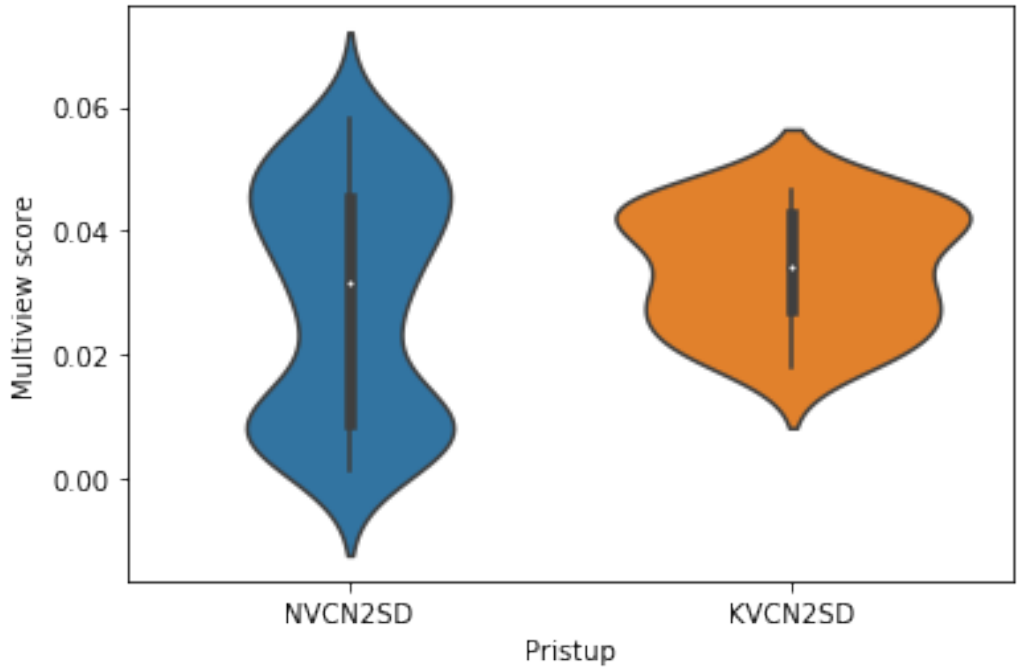


Slika 40: Testni Alzheimer skup podataka(Jaccard)

Malo lošija, no slična distribucija onoj na cijelom setu podataka što govori da je prediktivnost na Alzheimer skupu dobra.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=2600.0, pvalue=1.7506418769150495e-18)

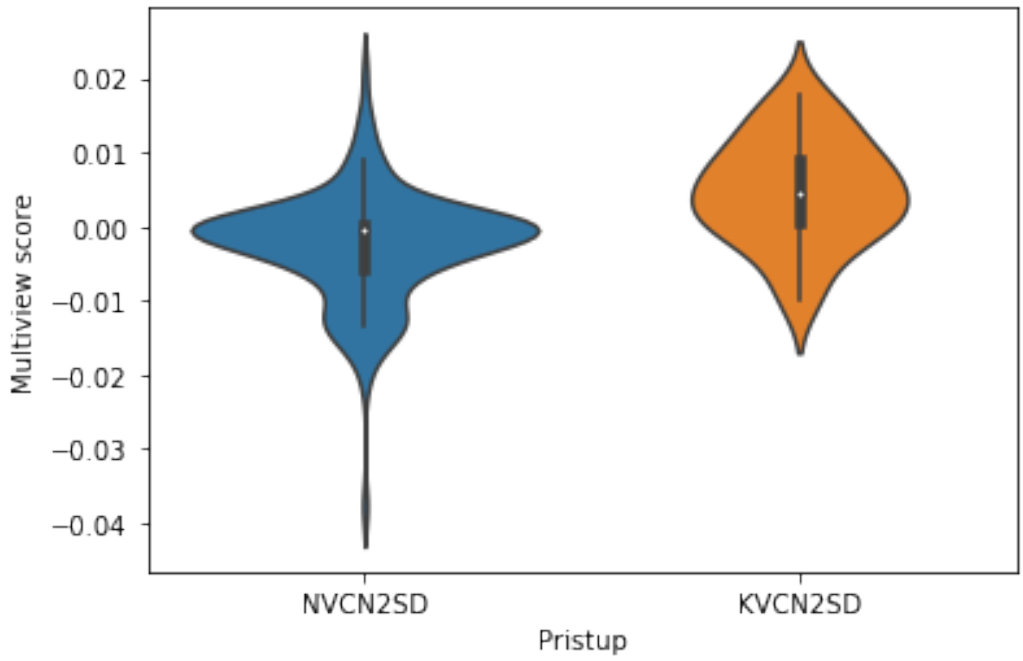
4.3.2 Skup podataka članaka



Slika 41: Skup podataka članaka

Na ovom skupu podataka medijan *Multiview score*-a je skoro isti s vrijednošću oko 0.033. Distribucija podgrupa je zanimljiva. Kompleksan pristup ima veću gustoću podgrupa oko medijana dok naivan pristup ima nekoliko boljih podgrupa od bilo koje u kompleksnom pristupu, ali također nekoliko lošijih.

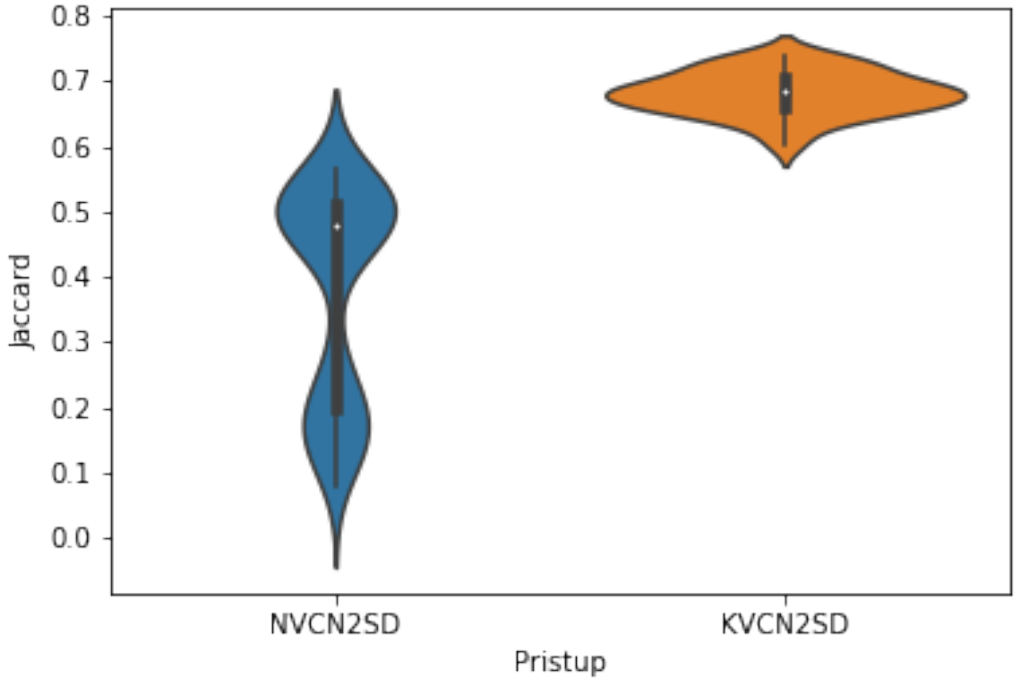
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=16820.0, pvalue=8.831113457751166e-32)



Slika 42: Testni skup podataka članaka

Na testnom skupu imamo sličan slučaj kao na cijelom skupu podataka. Ovdje se malo bolje vidi da je medijan kompleksnog pristupa veći te je i distribucija bolja. Prediktivnost je loša budući da je najviše podgrupa koncentrirano oko nule na testnom skupu kod naivnog pristupa. Kod kompleksnog pristupa je medijan ipak malo veći s boljom distribucijom.

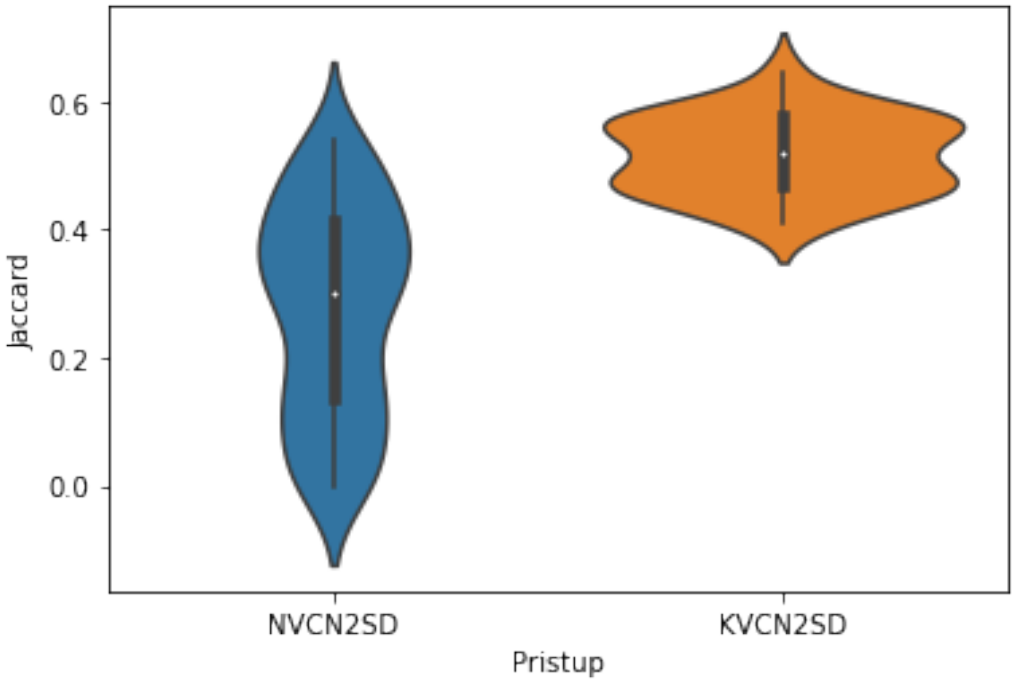
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=17670.0, pvalue=7.620138978776857e-39)



Slika 43: Skup podataka članaka (Jaccard)

Slično kao i na Alzheimer skupu podataka, medijan *Jaccard indeks* je znatno veći (oko 0.7) kod kompleksnog, nego kod naivnog pristupa (oko 0.5). Također, gustoća kod medijakna komplesnog pristupa je znatno veća nego kod naivnog. Na naivnom se vidi da postoji nekolicina podgrupa s *Jaccard indeks*-om 0.2.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=18496.0, pvalue=2.055749017822558e-46)

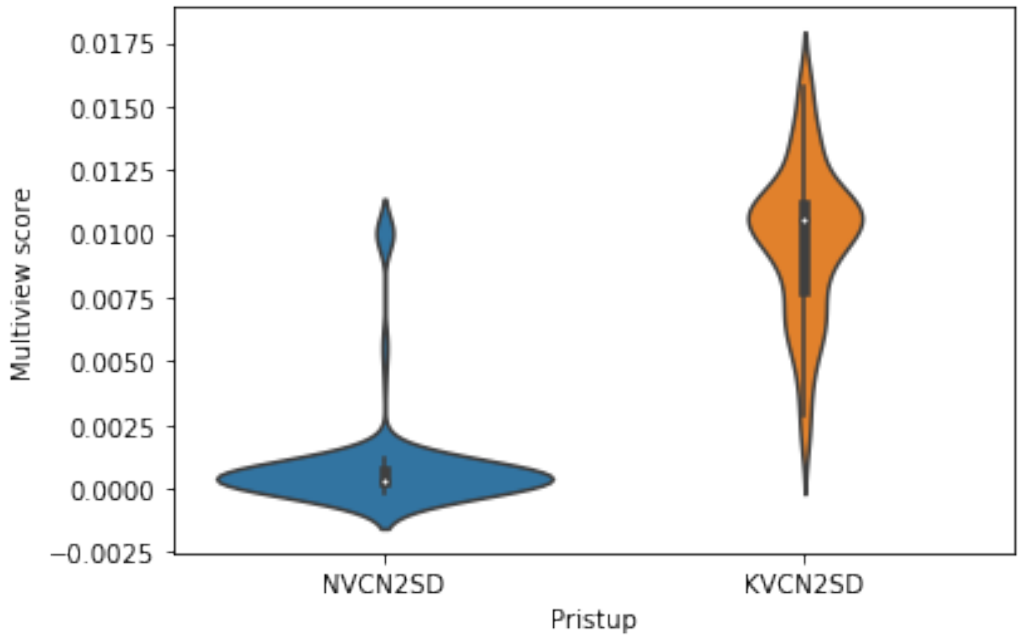


Slika 44: Testni skup podataka članaka (Jaccard)

Na testnom skupu medijani znatno padnu, kod kompleksnog pristupa s 0.7 na 0.5, a kod naivnog pristupa s 0.5 na 0.3 što govori da je prediktivnost relativno dobra.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=18219.0, pvalue=8.457002478138397e-44)

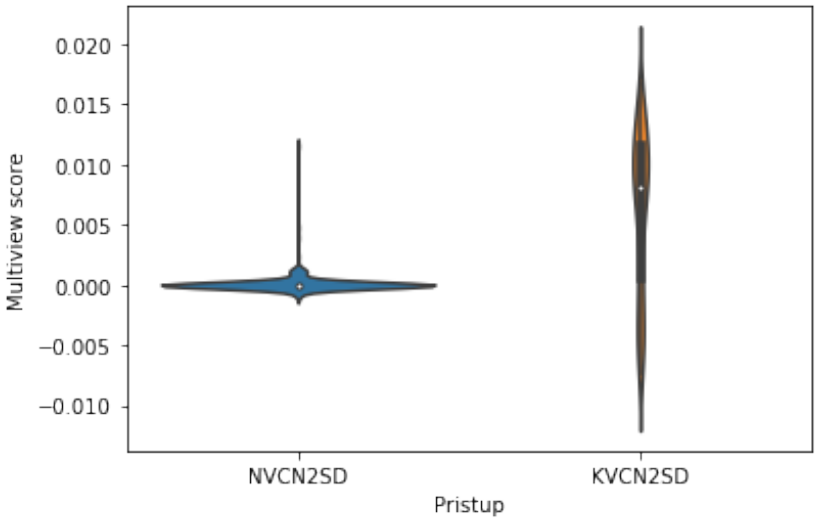
4.3.3 Skup podataka filmova



Slika 45: Skup podataka filmova

Na ovom skupu podataka vidimo veliku razliku između pristupa. Naivan pristup gotovo sve podgrupe koje inducira imaju vrijednost *Multiview score*-a oko nule, vrijednost medijana je isto oko nule, dok samo neke podgrupe imaju *Multiview score* oko 0.1 što je medijan kompleksnog pristupa.

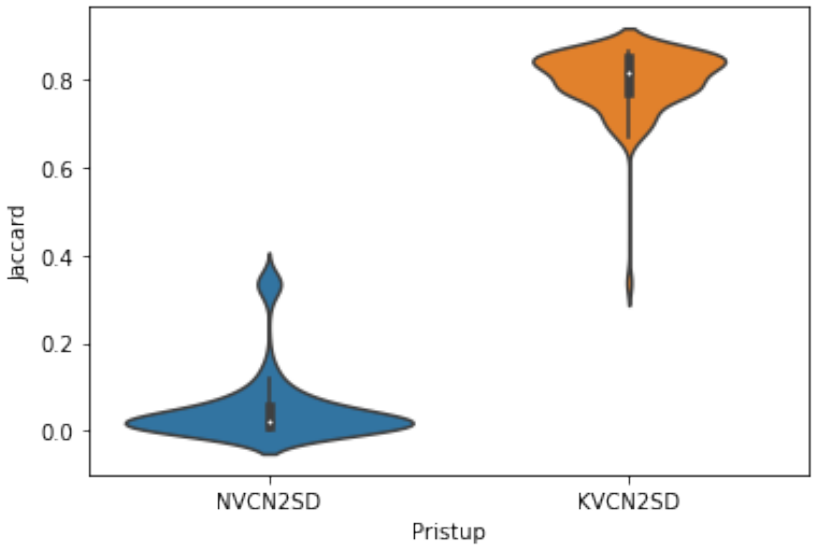
MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=74598.0, pvalue=4.804323095803454e-87)



Slika 46: Testni skup podataka filmova

Distribucija na testnom skupu slijedi distribuciju na cijelom skupu.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=63319.0, pvalue=2.9866212595130545e-45)

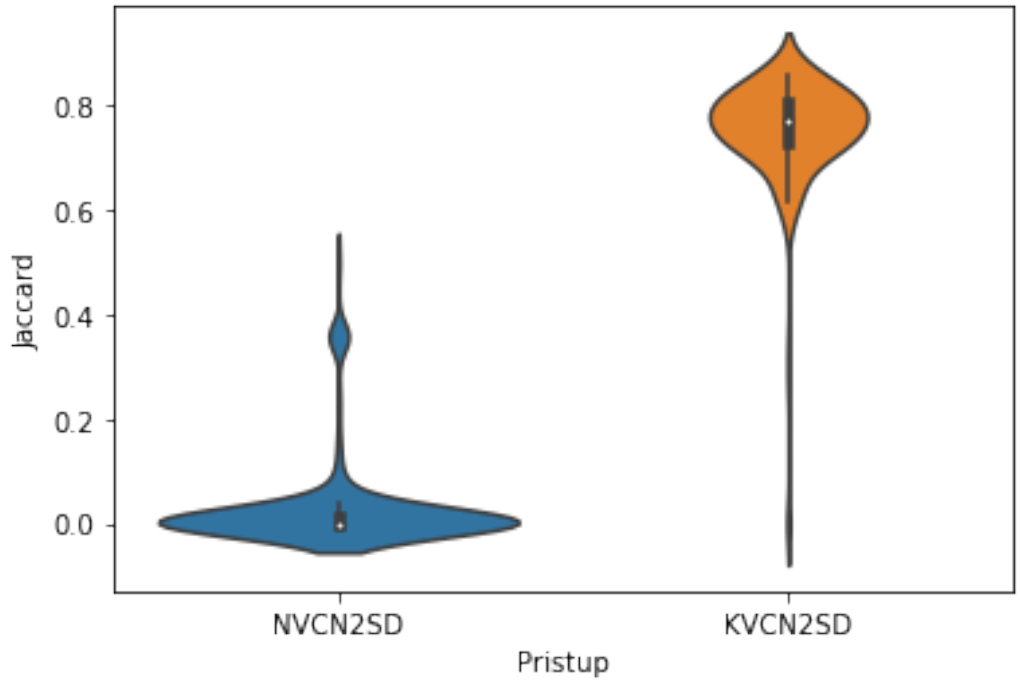


Slika 47: Skup podataka filmova (Jaccard)

Na ovom skupu podataka se vidi najveća razlika između pristupa. Naivni pristup inducira podgrupe koje se nimalo ne podudaraju ili se podudaraju u

neznačajno malo primjera. Medijan je oko nule. S druge strane kompleksni pristup inducira podgrupe s medijanom *Jaccard indeks*-a oko 0.8. To je iznimno visoka vrijednost za *Jaccard*, znači da je većina podgrupa inducirana kompleksnim pristupom jako podudarna.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=75621.0, pvalue=7.864368263110128e-92)



Slika 48: Testni skup podataka filmova (Jaccard)

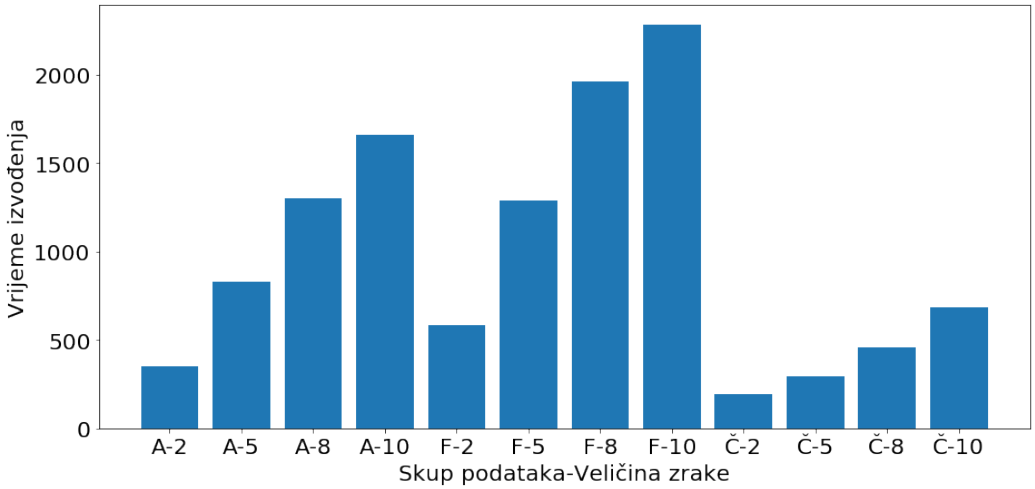
Distribucija na testnom skupu slijedi distribuciju na cijelom skupu podataka što govori da je prediktivnost dobra.

MannWhitneyUOneSided(KVCN2SD, NVCN2SD)=
MannwhitneyuResult(statistic=74706.0, pvalue=9.309135341631804e-95)

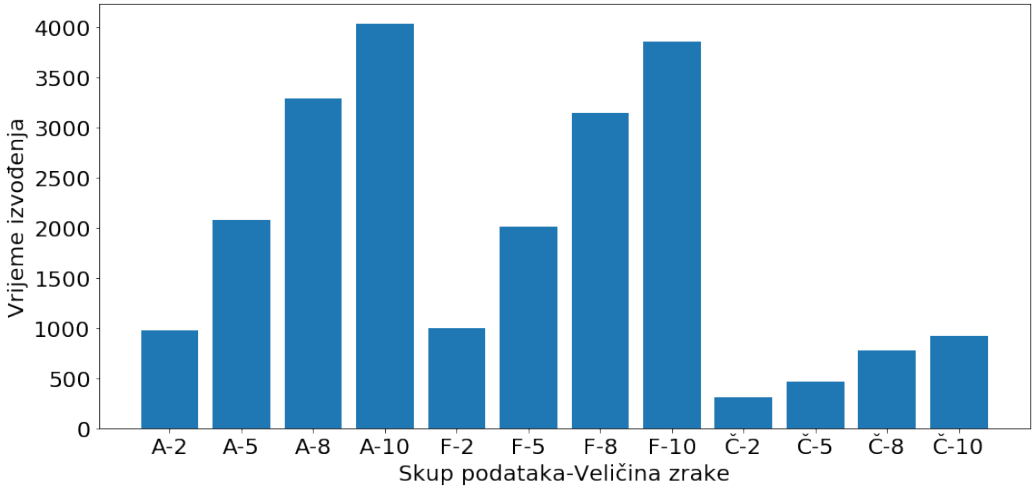
4.4 Prikaz vremena izvršavanja

Za prikaz vremena izvršavanja koristili smo stupčaste dijagrame na kojima je vrijeme izraženo u sekundama i koristimo sljedeće kratice: A - Alzheimer,

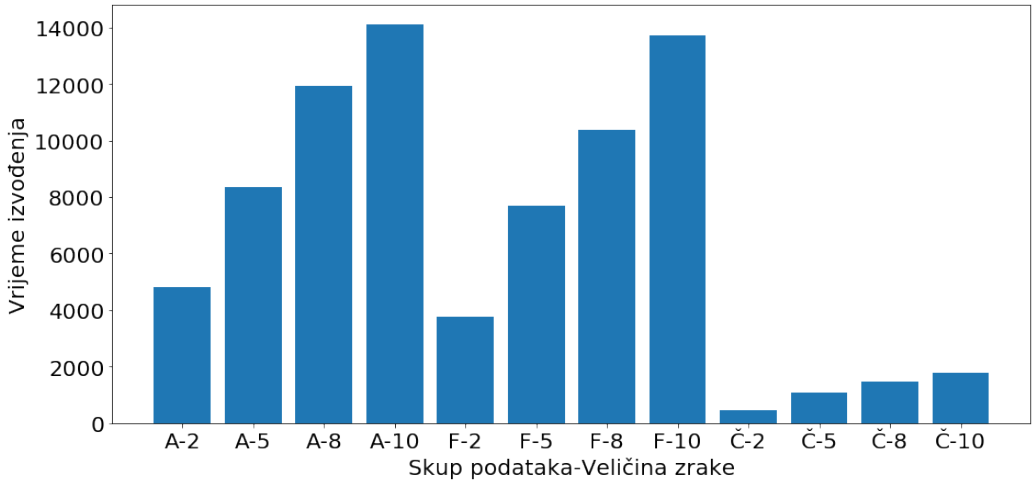
F - Filmovi, Č - Članci. Broj pokraj kratice predstavlja veličinu zrake.
Svi eksperimenti su izvedeni na istom računalu u istim uvjetima.



Slika 49: Vrijeme izvršavanja originalnog CN2SD na skupovima podataka



Slika 50: Vrijeme izvršavanja naivnog višepoglednog CN2SD na skupovima podataka



Slika 51: Vrijeme izvršavanja kompleksnog višepoglednog CN2SD na skupovima podataka

Možemo primijetiti da vrijeme izvršavanja raste linearno s povećanjem veličine zrake. Također, naivni višepogledni CN2SD izvršava se otprilike 2 puta duže od originalnog, a kompleksni višepogledni CN2SD 6 puta duže.

Literatura

[1] Carmona C.J., González P, del Jesus M.J., Naváo-Acosta M, Jiménez-Trevino L., *Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department. Soft Computing*, 2011, 15(12): 2435 – 2448

[2] Ljupčo Todorovski, Peter A. Flach, Nada Lavrač, *Predictive performance of weighted relative accuracy.* U *Proceedings of the Nineth International Workshop on Inductive Logic Programming*, str. 74 – 185, Springer, 1999.

[3] Nada Lavrač, Peter A. Flach, and Blaž Zupan, *Rule evaluation measures: A unifying view.* U *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, str. 255 – 264, Springer, 2000.

[4] D. Raj. Reddy, *Speech Understanding Systems: A Summary of Results of the Five-Year Research Effort. Department of Computer Science.* 1977.

[5] Trajkovski I, Železný F, Lavrač N, Tolar J., *Learning relational descriptions of differentially expressed gene groups.* IEEE Trans. Systems, Man, and Cybernetics, 2008., 38(1):16-25

[6] Trajkovski I, Železný F, Tolar J, Lavrač N., *Relational sub-group discovery for descriptive analysis of microarray data.*U *Proc. the 2nd International Conference on Computational Life Sciences*, rujan 2006., str. 86-96

[7] Kavšek B, Lavrač N., *Using subgroup discovery to analyze the UK traffic data.* Advances in Methodology and Statistics, 2004, 1(1): 249-264

[8] Kavšek B, Lavrač N, Bullas J C., *A case study in mining UK traffic accident data.* U Proc. International Multi-Conference on Information Society, siječanj 2002., 127-130

[9] Clark, P., Niblett, T., *The CN2 induction algorithm.* Mach Learn 3, , 1989., 261–283

[10] Lavrač N, Cestnik B, Gamberger D, Flach P., *Decision support through subgroup discovery: Three case studies and the lessons learned.* Machine Learning, 2004., 57(1/2): 115-143

[11] testDataFullBio, testDataFullClin, http://adni.loni.usc.edu/, datum zadnjeg pristupa 17.11.2021.

[12] M1.txt, M2.txt, Mact.txt, https://lig-membres.imag.fr/grimal/data.html, datum zadnjeg pristupa 17.11.2021.

[13] 3sources_global_bbc.txt, 3sources_global_guardian.txt, 3sourceLabel.txt, http://mlg.ucd.ie/datasets/3sources.html. datum zadnjeg pristupa 17.11.2021.

[14] Lavrač N., Kavšek B., Flach P., Todorovski Lj., *Subgroup discovery with CN2-SD*, Journal of Machine Learning Research 5 (2004) 153-188

[15] Helal S., *Subgroup Discovery Algorithms: A Survey and Empirical Evaluation*, Journal of computer science and technology

[16] https://en.wikipedia.org/wiki/Jaccard_index, datum zadnjeg pristupa 17.11.2021.

[17] https://www.baeldung.com/cs/beam-search, datum zadnjeg pristupa 17.11.2021.

Sažetak

U ovom diplomskom radu proučavamo CN2SD algoritam za traženje podgrupa te nudimo njegovo proširenje tako da traži podgrupe iz višepoglednih podataka.

U uvodnom dijelu objašnjavamo što je traženje podgrupa, za što se koristi te gdje leži njezin značaj u dubinskoj analizi podataka. Predstavljamo CN2SD algoritam za traženje podgrupa te zrakasto pretraživanje, odnosno algoritam pretraživanja koji CN2SD koristi. Također predstavljamo pseudokod CN2SD algoritma, kao i opis njegovog rada.

Glavni dio rada čine 2 prijedloga proširenja CN2SD algoritma za traženje podgrupa iz višepoglednih podataka, prijedlog nove ocjene dobrote podgrupa te pseudokodovi tih proširenja uz njihovo objašnjenje. Između ostalog radimo analizu vremenske složenosti CN2SD te oba proširenja. Na kraju smo proveli eksperiment na tri višepogledna skupa podataka, prikazali distribuciju podgrupa, usporedili obrađene pristupe traženju podgrupa te vremena izvršavanja algoritama.

Summary

In this graduate thesis we study CN2SD algorithm for subgroup discovery and propose its expenditure so it can be used for subgroup discovery from multiview data sets.

In the introduction of this graduate thesis we explain what is subgroup discovery, what is it used for and where lies its importance in deep learning. We present CN2SD algorithm for subgroup discovery and Beam search, a search algorithm that CN2SD uses. We also present CN2SD pseudocode, as well as explanation of CN2SD.

Main part of the thesis consists of two expanditure proposals of CN2SD algorithm for subgroup discovery from multiview data sets, proposal of new evaluation for subgroups induced from multiview data sets and pseudocodes of the two proposals with their explanation. Among other things we do time complexity analysis of CN2SD along with time complexity analysis of two expanditure proposals. In the end we conducted an experiment on three multiview data sets, showed subgroup distribution, compared elaborated approaches to subgroup discovery with corresponding algorithm execution times.

Životopis

Rođen sam 04. siječnja 1996. godine u Zaboku. Završio sam Osnovnu školu Ksaver Šandor Gjalski te Osnovnu glazbenu školu Zabok, smjer gitara, 2010. godine. Gimnaziju Antuna Gustava Matoša Zabok, smjer prirodoslovno-matematička gimnazija, završio sam 2014. godine. Preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu, Sveučilišta u Zagrebu, završavam 2019. godine te iste godine upisujem Diplomski studij Računarstva i matematike na istom fakultetu. Od hobija se bavim fitnessom, košarkom i šahom.