SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4850

POSTUPCI ODABIRA ZNAČAJKI

Josip Renić

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 6. ožujka 2017.

ZAVRŠNI ZADATAK br. 4850

Pristupnik: **Josip Renić (0036486723)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Postupci odabira značajki**

Opis zadatka:

Postupci odabira značajki koriste se kako bi se smanjila dimenzionalnost ulaznog skupa podataka i tako omogućilo postupcima za izgradnju modela da što učinkovitije pronađu rješenje problema.
Postupci odabira značajki u slučaju pretpostavke o međusobnoj neuvjetovanosti pojedinih značajki mogu se podijeliti na filterske postupke, postupke omotača, ugrađene postupke i hibridne postupke.
Potrebno je proučiti danas dostupne postupke za odabir značajki (npr. symmetrical uncertainty, ReliefF, ChiSquare, omotač Naive Bayes) te implementirati u programskom jeziku Java najmanje tri takva postupka.
Osim implementacije samih postupaka za odabir značajki, potrebno je podržati i učitavanje podataka te prikaz rezultata analize kroz sučelje implementiranog programa.
Procjenu uspješnosti primjene implementiranih postupaka odabira značajki potrebno je provesti na temelju više otvorenih skupova podataka iz repozitorija UC Irvine Machine Learning (UCI).
Rezultate treba usporediti s rezultatima već postojećih implementacija u programskim paketima Weka, RapidMiner ili KEEL.
Za vrednovanje uspješnosti odabranog podskupa značajki mogu se koristi već postojeće implementacije algoritama za izgradnju modela.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:
Doc. dr. sc. Alan Jović

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

Dielovođa:
Doc. dr. sc. Tomislav Hrkać

SADRŽAJ

Sadržaj

1. Uvod

U današnjem vremenu svakodnevno se generiraju rekordne količine podataka. Tu nam se javlja potreba da iskoristimo te novonastale izvore informacija. Metode i alate za izvlačenje znanja iz samih podataka nam daje strojno učenje pod čiju disciplinu spadaju i postupci odabira značajki. Gradeći model podataka koje promatramo prvo pitanje jest koje značajke izabrati za opis tog modela? Na to nam pitanje odgovor mogu dati samo eksperti iz dotičnih područja. No što ako mi nismo eksperti tog područja? Kako ćemo onda vrednovati kvalitetu odabranih značajki?
Postoji li mogućnost automatskog odabira korisnih i izbacivanja štetnih značajki za naš model? Upravo taj problem rješavaju postupci odabira značajki. Oni se koriste kako bi smanjili dimenzionalnost ulaznog skupa podataka i tako poboljšali učinkovitost postupaka strojnog učenja za izgradnju modela.

U ovom radu pokušat ćemo dati odgovore na pitanja: koji postupci odabira značajki postoje, kako ih dijelimo i zašto ih koristiti. Pokazat ćemo, objasniti i usporediti nekoliko implementacija postupaka odabira značajki s već gotovim implementacijama. Uzet ćemo u obzir da čitatelj razumije osnovne koncepte iz područja strojnog učenja poput klasifikacije i izgradnje modela itd. i osnovne koncepte iz statistike poput pojma distribucije, *p*-vrijednosti, itd.

2. Izvori podataka

Razmatrat ćemo podatke s vrlo velikim brojem primjera i varijabli. Oni će nam biti interesantni u okviru problema izbora značajki koji se opisuje ovim radom. Opisat ćemo nekoliko područja i izvora takvih podataka:

Medicinski i biološki podatci

Jedan od možda i najboljih primjera važnosti postupaka odabira značajki jest obrada gena tj. pronalazak biomarkera iz genetskih zapisa. Pronalaskom najvažnijih značajki, iz tisuća značajki koje sadrži genetski zapis, možemo znatno ubrzati proces pronalaska interesantnih gena za naš problem.

Problem detekcije i klasifikacije srčane aritmije iz EKG zapisa

Iz jednog EKG zapisa možemo izvući stotine različitih značajki. Primjer možemo vidjeti na skupu podataka *Arrhythmia Data Set* (https://archive.ics.uci.edu/ml/datasets/Arrhythmia) koji sadrži 279 različitih atributa/značajki.

Podatci iz sociologije i društvenih znanosti

Sociološka istraživanja koja koriste podatke iz društvenih mreža koji najčešće imaju jako veliki broj značajki. Jedan od interesantnih problema koje možemo ispitati je ocjena kvalitete blogova, čiji podatkovni skup *BlogFeedback Data Set (*https://archive.ics.uci.edu/ml/datasets/BlogFeedback*)* sadrži 281 značajku.

Jedno od zanimljivih pitanja je povezanost razvijenosti zajednice s razinom zločina koje opisuje skup podataka *Communities and Crime Data Set (*https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime*)*, koji sadrži 128 značajki.

Obrada teksta

Dobar primjer obrade teksta je klasifikacija elektroničke pošte na interesantnu i beskorisnu. Kao značajke svake pošte možemo uzeti sve riječi koje su se pojavile. No nije nam svaka riječ interesantna za našu klasifikaciju, zbog čeka moramo odabrati samo one riječi/značajke koje najviše pridonose klasifikaciji.

Obrada slike

Za klasifikaciju slike teoretski možemo uzeti svaki piksel kao jednu značajku slike, što ne bi imalo smisla. Primjer jednog takvog skupa podataka iz kojeg mi sami trebamo pronaći interesantne značajke slike jest *CMU Face Images Data Set (*https://archive.ics.uci.edu/ml/datasets/CMU+Face+Images*).*

Meteorološki podatci

Svakodnevno dobivamo podatke iz tisuća meteoroloških postaja, balona, senzora itd. Primjer jednog takvog skupa podataka je *Greenhouse Gas Observing Network Data Set* koji sadrži 5232 značajke.
*(*https://archive.ics.uci.edu/ml/datasets/Greenhouse+Gas+Observing+Network*)*

Za samu implementaciju koristi ćemo podatke iz *UC Irvine Machine Learning Repository* (https://archive.ics.uci.edu/ml/datasets.html) i podatke koji dolazi uz programski alat *Weka* Prvenstveno ćemo se fokusirati klasifikacijske probleme i na podatke s kategorijskim (nominalnim) značajkama.

3. Kategorizacija postupaka

Postoje četiri opće klase postupaka za odabir značajki: filtarski postupci, postupci omotača, ugrađeni postupci i hibridni postupci[1]. U ovom radu obradit ćemo neke primjere filtarskih postupaka, no važno je objasniti i ostale.

Postupci omotača odabiru podskup korisnih značajki tako da podskup minimizira pogrešku korištenog algoritma za izradu modela. Izračun optimalnog podskupa je izrazito računski zahtjevan, budući da trebamo za svaki potencijalni podskup izgraditi novi model. S obzirom da optimiziramo performanse algoritma, postoji opasnost od pojave pristranosti prema korištenom algoritmu. Iz tog razloga, nakon što je nađen optimalni podskup značajki, model gradimo drugim algoritmom i testiramo ga nad drugim skupom ulaznih podataka.

Filtarski postupci primjenjuju mjere, najčešće statističke, kako bi bodovali svaku značajku prema njenoj korisnosti. Ovi postupci su neovisni o algoritmu za izgradnju modela podataka. Time smo izbjegli pristranost prema algoritmu tj. filtarski postupci ne odabiru skup značajki koji minimizira pogrešku algoritma već skup najbolje rangiranih značajki. Najčešće se koriste kao korak prije izgradnje modela za odabir najkorisnijih značajki. Mjere koje se koriste trebaju se moći brzo izračunati, a značajke najčešće rangiraju prema nekoj mjeri korisnosti.

Ugrađeni postupci obavljaju odabir značajki tijekom izvedbe algoritma za izradu modela. Najčešće postupci koriste metode regularizacije odnosno koriste penalizaciju značajki kako bi umanjili pogrešku prevelike prilagodbe modela podatcima [8].

Hibridni postupci su kombinacija filtarskih postupaka i postupaka omotača. Prvi korak je primjena filtarskog postupka kako bi se smanjio broj značajki ili dobilo više podskupova korisnih značajki. Zatim se koriste postupci omotača kako bi se izabrale najbolje iz pod skupova dobivenih filtarskim postupkom.

4. Implementacija postupaka

U sklopu implementacije algoritama, razvijen je jednostavan program s grafičkim sučeljem za provođenje postupaka odabira značajki o kojem će biti više riječi u petom poglavlju. Implementirana su tri algoritma: *symmetrical uncertainty*, *ReliefF* i *ChiSquare*. Algoritmi su ispitivani isključivo nad nominalnim podatcima za postupke klasifikacije. Dobivene vrijednosti implementiranih postupaka uspoređivane su s vrijednosti implementacija istih postupaka u gotovim alatima *Weka* i *RapidMiner*. Postupci *Symmetrical uncertainty* i *ReliefF* su implementirani u alatu *Weka* dok je *ChiSquare* postupak implementiran u alatu *RapidMiner*. Kada se u nastavku rada budemo referencirali na postupke iz ovih alata nakon imena postupka pisti će *_W* za postupke iz alata *Weka* i *_RM* za postupke iz alata *RapidMiner.* Skupovi podataka nad kojima ispitujemo implementacije algoritama dobiveni su iz skupa podataka koji dolazi sa standardnom instalacijom alata *Weka*, a svi su podatci spremljeni u .arff formatu. Za ispitivanje koristit ćemo 3 skupa podataka: „weather.nominal“ s 4 značajke i 14 primjera, „vote“ s 16 značajki i 435 primjera te „audiology“ s 70 značajki i 226 primjera, s time da ćemo promatrati samo 15 najboljih značajki skupa „audiology“.

4.1 Rangiranje značajki

Sva tri implementirana algoritma možemo svrstati u filtarske postupke odabira značajki. Ova tri algoritma pripadaju skupini postupaka koji pridružuju težine i rangiraju svaku pojedinu značajku prema njenoj važnosti za klasifikacijski problem. Značajku smatramo dobrom ako je veća od zadanog praga kojeg korisnik sam bira. Potrebno je biti oprezan pri zadavanju praga jer svaki algoritam drugačije rangira istu značajku što znači da za isti prag dobivamo drugačiji podskup dobrih značajki. Tom problemu možemo doskočiti birajući prvih N dobrih značajki no opet treba biti oprezan jer se N-ta i N+1 značajka mogu neznatno razlikovati i tako možemo izbaciti korisne značajke. Dolazimo do zaključka da je za korištenje postupaka odabira značajki potrebno razumjeti kako i što rade tj. što znače težine koje postupak pridijeli svakoj značajki. Implementacije postupaka neće raditi nikakvu selekciju nad značajkama, već će ih rangirati i ostaviti korisniku zadatak odabira željenog skupa značajki.

4. 2 Symmetrical uncertainty

Mjera *Symmetrical uncertainty* korisnosti značajke za izgradnju klasifikacijskog modela temelji se na mjeri korelacije između same značajke i klase modela. Ako uzmemo kao mjeru korisnosti značajke korelaciju između značajke i klase općenito možemo reći da je značajka korisna ako poznavanje značajke pridonosi klasifikaciji ulaznog primjera. Što znači da je značajka dobra ako je njezina korelacija s klasom visoka. Želji bismo izbjeći korelacije među značajkama, dakle pretpostavljamo međusobnu nezavisnost značajki, što u praksi ne mora vrijediti te u tom slučaju potrebno je koristiti napredniju verziju algoritma koja će niže biti opisana. Ideja je vrednovati koliko nam značajka donosi informacije o pripadnosti ulaza pojedinoj klasi. Za opis ove mjere potrebno je poznavanje pojmova iz teorije informacija poput entropije i uvjetne entropije. Entropiju varijable *X* definiramo :

$$H(X) = -\sum_j P(x_i)\ \log_2\big(P(x_i)\big) \tag{1}$$

Uvjetnu entropiju varijable *X* nakon poznavanja vrijednosti varijable *Y* definiramo:

$$H(X|Y) = -\sum_j P\big(y_j\big) \sum_i P\big(x_i|y_j\big) \log_2\Big(P\big(x_i|y_j\big)\Big) \tag{2}$$

Vjerojatnost $P(y_j)$ nazivamo apriornom vjerojatnošću odnosno vjerojatnošću pojavljivanja *j*-te vrijednosti klase. Vjerojatnost $P(x_i|y_j)$ nazivamo uvjetnom vjerojatnošću, odnosno vjerojatnošću pojavljivanja *i*-te vrijednosti atributa X uz poznavanje da je se na ulazu pojavila *j*-ta vrijednost atributa *Y*. Količinu dodatne informacije koju Y donosi o X vrijednosti definiramo:

$$IG(X|Y) = H(X) - H(X|Y) \tag{3}$$

No ova mjera informacije nam nije poželjna jer postoji pristranost prema značajkama s većim brojem vrijednosti. Kako bi izbjegli tu pristranost i postigli normalizaciju težina da bi ih mogli i rangirati, uvodimo mjeru *symmetrical uncertainty* koju definiramo:

$$SU(X,Y) = 2\left[\frac{IG(X|Y)}{H(X) + H(Y)}\right] \tag{4}$$

Ovaj postupak ćemo pokazati na jednostavnom primjeru, kako bi se intuitivnije shvatilo što on radi. Pretpostavimo da gradimo klasifikacijski model i imamo dvije značajke A i B koje poprimaju vrijednosti {T, F} i klasu C koja poprima vrijednosti {yes, no}. Ako dobijemo 3 skupa podataka od 4 primjera, navodeći vrijednosti A,B zatim klase C, s vrijednostima:

1) (F,T,no), (F,T,no), (T,T,yes) , (T,T,yes)
2) (F,T,no), (F,T,no), (T,T,yes) , (T,F,yes)
3) (T,T,no), (F,T,no), (T,T,yes) , (T,F,yes)

Nad prvim skupom dobili smo rezultat SU(A,C) = 1 i SU(B,C) = 0. To možemo zaključiti na prvi pogled uvidjevši da informacija koju značajka A donosi u potpunosti opisuje klasu C tj. korelacija je maksimalna. Dok nam informacija o značajci B nikako ne opisuje klasu C tj. korelacija ne postoji.

Nad drugim skupom dobili smo rezultat SU(A,C) = 1 i SU(B,C) = 0.344. Informacija značajke A još uvijek u potpunosti opisuje klasu C, dok informacija značajke B djelomično opisuje klasu C, što znači da nam je još uvijek dovoljna značajka A da iz podataka u potpunosti odredimo klasu C iako postoji korelacija između značajke B i klase C.

Nad trećim skupom dobili smo rezultat SU(A,C) = 0.344 i SU(B,C) = 0.344. Informacija značajke A i B u jednakoj mjeri opisuje klasu C što znači da su te značajke podjednako korisne.

Implementacija

Implementacija samog algoritma provodi se u dva koraka: izračunavanje SU za svaki par značajka-klasa i rangiranje dobivenih vrijednosti. Izračun SU(značajka, klasa) se provodi prema formuli (4). Prvi korak izračuna SU jest izračun entropije značajke i entropije klase prema formuli (1). Implementaciju izračuna entropije možemo vidjeti na slici 1.

```
public static double entropy(Attribute attr) {
    double entropy = 0;
    double sum = 0;

    double n = attr.getSize();

    for(Integer row : attr.getLabelsCounter().values()) {
        double p = row/n;
        if(p > 0) {
            entropy -= p*Math.log(p);
            sum +=p;
        }
    }

    //nemamo logaritam po bazi 2 u Math
    return (entropy + Math.log(sum)) / (sum*Math.log(2));
}
```

Slika 1.Izračun entropije zadanog atributa

Idući korak je izračun informacijskog doprinosa prema formuli (3) za što nam treba uvjetna entropija koju računamo po formuli (2). Za dobivanje uvjetne entropije prvo gradimo kontingencijsku tablicu. Na slici 2 možemo vidjeti primjer jedne takve tablice gdje je značajka dominantna ruka a klasa spol ispitanika.

Handed-ness / Gender	Right handed	Left handed	Total
Male	43	9	52
Female	44	4	48
Total	87	13	100

Slika 2 Kontingencijska tablica [10]

Prema formuli (2) potrebno je izračunati $P(y_j)$ što lagano dobivamo dijeleći sumu pojavljivanja y_j vrijednosti s ukupnim brojem primjera. U primjeru s slike 2 $P(Male) = 52/100$, gdje je $y_j = Male$ u našem slučaju. Zatim je potrebno izračunati uvjetnu vjerojatnost $P(x_i|y_j)$ koju dobivamo tako da svaku vrijednost unutar kontingencijske tablice dijelimo s ukupnim brojem pojavljivanje vrijednosti klase y_j. U primjeru s slike 2 $P(Right\ handed|Male) = 43/52$, gdje je $y_j = Male$ a $x_i = Right\ handed$. Nakon što smo dobili sve potrebne brojeve težinu značajke dobivamo prema formuli (4).

Usporedba s gotovom implementacijom

Tablica 1. Postupak symmetrical uncertainty nad skupom podatak weather.nominal

Značajke	SU_W	SU
outlook	0,196	0,19601
humidity	0,1565	0,15651
windy	0,05	0,04999
temperature	0,0234	0,02341

Iz tablice 1 možemo vidjeti da naša implementacije daje identične rezultate kao i programski alat *Weka*. Razlike u zadnjoj decimali nastaju samo zbog Wekinog ograničenja prikaza decimalnog rezultata.

Tablica 2. Postupak symmetrical uncertainty nad skupom podataka vote

Značajke	SU_W	SU
physician-fee-freeze	0,728786	0,78037
adoption-of-the-budget-resolution	0,432598	0,45862
el-salvador-aid	0,4105911	0,43910
education-spending	0,3499392	0,41593
crime	0,3224629	0,36148
aid-to-nicaraguan-contras	0,3182305	0,35058
mx-missile	0,2911317	0,31889
superfund-right-to-sue	0,2162762	0,24724
duty-free-exports	0,2068379	0,24397
anti-satellite-test-ban	0,1952164	0,20956
religious-groups-in-schools	0,1475674	0,15210
handicapped-infants	0,1241247	0,13011
synfuels-corporation-cutback	0,1056645	0,11918
export-administration-act-south-africa	0,0639225	0,20468
immigration	0,0050042	0,00518
water-project-cost-sharing	0,0000119	0,00013

Iz tablice 2 možemo vidjeti da naša implementacije daje rezultate s vrijednosti prosječnog odstupanja od 0.032. Razlika dolazi zbog razlike u postupanju s nepoznatim vrijednostima. Naša implementacija postupka SU nije posebno prilagođena za rad s nepoznatim vrijednostima. Pri izradi kontigencijskih tablica nepoznate vrijednosti preskačemo što nije dobra strategija no to nije problematika rada. Najbolji primjer te razlike se vidi na značajci „export-administration-act-south-africa“ čiji postotak nepoznatih vrijednosti iznosi 24%.

Tablica 3. Postupak symmetrical uncertainty nad skupom podataka audiology

Značajke	SU_W	SU
age_gt_60	0,41095	0,41095
tymp	0,39863	0,39863
history_noise	0,26764	0,26764
air	0,24321	0,24321
o_ar_c	0,23653	0,2625
airBoneGap	0,22536	0,22536
ar_u	0,21236	0,21632
ar_c	0,19691	0,20786
speech	0,19087	0,20344
bone	0,14876	0,35451
o_ar_u	0,14284	0,14508
notch_at_4k	0,11376	0,11376
history_fluctuating	0,09942	0,09942
boneAbnormal	0,08885	0,08885
history_dizziness	0,08855	0,08855

Iz tablice 3 možemo vidjeti da naša implementacije daje rezultate s vrijednosti prosječnog odstupanja od 0.017. U ovom primjeru razlika je manja jer prosječno po svakoj značajki fali po svega par posto vrijednosti. Trebamo istaknuti veliko odstupanje na značajci „bone“ čiji postotak nepoznatih vrijednosti iznosi 33%.

Nadogradnja

Prema radu [4], potrebno je istražiti i međusobnu korelaciju između značajki. Sve dok se značajke smatraju relevantnima za predikciju klase one će se odabrati bez obzira na postojanje jake korelacije između parova značajki. Takve značajke

dovode do redundancije među značajkama. Redundantne značajke ne pridonose razumijevanju, brzini i preciznosti učenja modela i takve bi također trebalo eliminirati [5,6]. Prema tome, kada nismo sigurni u pretpostavku međusobne nezavisnosti značajki, ne bismo trebali koristiti algoritme koji značajkama pridružuju težine, već algoritme koji daju kao izlaz podskup bitnih značajki. Primjer takvog algoritma bi bio **CFS** (engl. *Correlation-based Feature Selection*) algoritam [6]. Također algoritam koji nadograđuje opisano znanje o mjeri SU značajki **FCBF** (engl. *Fast Correlation Based Filter*), koji je prikazan na slici 3.

```
Ulaz: S(f1, f2, ..., fN, C) // skup podataka za treniranje
      δ                     // zadana granica
Izlaz: S_best               // podskup optimalnih značajki
1.  početak
2.    za i=1 do N radi:
3.      izračunaj SU_i,c za f_i;
4.      ako (SU_i,c > δ):
5.        dodaj f_i u S'_lista;
6.    završi;
7.    poredaj S'_lista silaznim redoslijedom prema SU_i,c vrijednosti;
8.    f_p = dohvatiPrviElement(S'_lista);
9.    započni
10.     f_q = dohvatiIdućiElement(S'_lista, f_p);
11.     ako (f_q <> NULL)
12.       započni
13.         f'_q = f_q;
14.         ako (SU_p,q ≥ SU_q,c)
15.           ukloni f_q iz S'_lista;
16.           f_q = dohvatiIdućiElement(S'_lista, f'_q);
17.         inače f_q = dohvatiIdućiElement(S'_lista, f_q);
18.       završi kada je (f_q == NULL);
19.     f_q = dohvatiIdućiElement(S'_lista, f_p);
20.   završi kada je (f_p == NULL);
21. S_best = S'_lista;
22. kraj;
```

Slika 3.Pseudokod algoritma FCBF [4]

Algoritam ima dva koraka. U prvom koraku se računaju težine svake značajke s obzirom na zadanu klasu i izbacuju one manje od zadanog praga (linije 2-7). U drugom koraku, izbacuju se redundantne značajke, ostavljajući podskup predominantnih značajki. Za korelaciju između značajki *fi* (*fi* $\in S$) i klase *C* kažemo da je predominantna ako ne postoji značajka za koju vrijedi $SU_{j,i} > SU_{i,c}$,

Ako postoji takva značajka, f_j nazivamo redundantnim vršnjakom f_i značajke i smještamo je u skup značajki Sp_i tj. u skup Sp_i^+ ako vrijedi $SU_{j,c} > SU_{i,c}$,a inače u Sp_i^-.

Značajka je predominantna za klasu ako je njena korelacija predominantna prema klasi ili može postati uklanjanjem redundantnih vršnjaka. Još je potrebno definirati heuristike za pronalazak predominantnih značajki kako bismo smanjili složenost algoritma.

1. heuristika: krećemo od značajke s najvišom vrijednošću $SU_{i,c,}$ jer je ona uvijek predominantna značajka.

2. heuristika: ako je Sp_i^+ prazan skup f_i tretiramo kao predominantnu značajku, a sve značajke iz skupa Sp_i^- možemo ukloniti.

3. heuristika: ako Sp_i^+ nije prazan skup, prvo obrađujemo sve značajke iz Sp_i^+, a ako barem jedna značajka postane predominantna uklanjamo f_i iz skupa korisnih značajki.

Ovim postupkom rješavamo problem dodavanja redundantnih značajki u konačni skup korisnih značajki. Za detaljnije objašnjenje heuristika i samog algoritma pogledati rad Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution [4]

4.3 ChiSquare

Postupak *Chi Square* (hrv. Hi-kvadrat) koristimo za testiranje statističke nezavisnosti značajke i klase za potrebe klasifikacije. Mjera hi-kvadrat dolazi nam iz statistike, za razliku od mjere SU koja se oslanja na znanje iz teorije informacija. Za provođenje statističkog testa hi-kvadrat, potrebno je znati distribuciju hi-kvadrat sa $(r-1)(c-1)$ stupnjeva slobode, gdje r predstavlja broj mogućih vrijednosti značajke, a c broj mogućih vrijednosti klase. Statistika hi-kvadrat definirana je formulom :

$$\chi^2 = \sum_i \frac{(o_i - e_i)^2}{e_i} \qquad (5).$$

Za računanje ove statistike gradimo kontingencijsku tablicu, gdje e_i predstavlja očitanu frekvenciju i-te kombinacije značajke i klase iz tablice, a o_i predstavlja očekivanu frekvenciju koju računamo kao umnožak vjerojatnosti pojavljivanja vrijednosti klase i vjerojatnosti pojavljivanja vrijednosti značajke za i-tu kombinaciju iz tablice.

Uz poznavanje distribucije potrebno je i odrediti nivo značajnosti uz koji provodimo test tj. za koji odbacujemo hipotezu nezavisnosti varijabli za dobivenu p-vrijednost. Statistika hi-kvadrat za potpuno nezavisne varijable daje rezultat 0, dok s povećanjem zavisnosti raste i broj koji statistika daje. Za više informacija o statističkoj pozadini testa hi-kvadrat pogledati [7].

Trenutačno nama to nije interesantno, jer mi želimo pridijeliti težine našim značajkama kako bi ih mogli rangirati. Ideja je raditi obrnuto od samog testa hi-kvadrat. Mi želimo dobiti kao najbolju značajku onu čija je zavisnost s klasom najveća tj. statistika hi-kvadrat za nju daje najveći broj. Takvim rangiranjem lagano možemo prepoznati značajke koje nisu korisne, jer će njihova težina biti blizu 0.

Korištenje postupka ćemo ilustrirati nad dvije značajke („temperature“, „outlook“) iz „weather.nominal“ skupa podataka. Prvi korak je izgradnja kontingencijskih tablica:

Tablica 4. Kontingencijska tablica značajki

temperature\play	*yes*	*no*	*outlook\play*	*yes*	*no*
Hot	2	2	*Sunny*	2	3
Mild	4	2	*Overcast*	4	0
Cool	3	1	*Rainy*	3	2

Iz tablice 4 možemo dobiti očitane frekvencije kombinacija npr. kombinacije(*Hot*, yes) frekvencija je 2. Očekivane frekvencije dobivamo formulom:

$$o_i = \frac{N(X_i) * N(Y_i)}{N} \tag{6}$$

Očekivana frekvencija kombinacije (*Hot, yes*) jednaka je $\frac{N(Hot)*N(yes)}{N(ukupno)}$ tj. $\frac{4*9}{14} = 2.57$

Računajući očekivane frekvencije svake kombinacije dobivamo tablicu 5:

Tablica 5. Kontingencijska tablica očekivanih frekvencija značajki

temperature\play	yes	No	outlook\play	yes	No
Hot	2.57	1.43	Sunny	3.21	1.78
Mild	3.86	2.14	Overcast	2.57	1.43
Cool	2.57	1.43	Rainy	1.92	0.71

Uvrštavajući dobivene vrijednosti u formulu (5) dobivamo:

$$ChiSquare(temperature) = 0.569 \; i \; ChiSquare(outlook) = 3.547$$

Težina značajke „temperature“ nam govori da je značajka nezavisna prema klasi prema tome tu značajku ne bi koristili za izgradnju modela. Iz tablice 5 možemo zaključiti da postoji zavisnost između značajke „outlook“ i klase zbog većeg odstupanja očekivanih od očitanih frekvencija.

Implementacija

Implementacija ovog postupka svodi se na izgradnju potrebnih kontingencijskih tablica i izračuna formule (5). Na slici 4 možemo vidjeti kod implementacije. Za izračun se koristi kontingencijska tablica i marginalne tablice svakog atributa dok se očekivana frekvencija izračunava u hodu.

```
public static double chiSquareScore(Attribute a1, Attribute a2) {
    double chiScore = 0.0;

    int[] table = ContingencyTable.generateTable(a1, a2);

    int[] boundaryA1 = ContingencyTable.boundaryTable(a1);

    int[] boundaryA2 = ContingencyTable.boundaryTable(a2);

    double N = (double)IntStream.of(table).sum();
    int M,P;
    double exp;

    for(int i = 0, nRow = a1.getLabelsSize(), nCol = a2.getLabelsSize(); i < nRow; i++) {
        for(int j = 0; j < nCol; j++) {
            M = boundaryA1[i];
            P = boundaryA2[j];

            exp = M*P/N;

            chiScore += Math.pow((exp-table[i*nCol+j]), 2)/exp;
        }
    }

    return chiScore;
}
```

Slika 4. Implementacija postupka ChiSquare

Usporedba s gotovom implementacijom

Tablica 6. Postupak ChiSquare nad skupom podataka weather.nominal

Značajke	CS_RM	CS
outlook	3,547	3,54667
humidity	2,8	2,80000
windy	0,933	0,93333
temperature	0,57	0,57037

Iz tablice 6 možemo vidjeti da naša implementacije daje identične rezultate kao i programski alat *Rapid Miner*.

Tablica 7. Postupak ChiSquare nad skupom podataka vote

Značajke	CS_RM	CS
physician-fee-freeze	363,04	353,55
adoption-of-the-budget-resolution	237,93	232,17
el-salvador-aid	220,06	211,39
education-spending	206,13	193,52
aid-to-nicaraguan-contras	189,58	176,19
mx-missile	171,88	158,80
crime	163,31	157,55
superfund-right-to-sue	126,65	120,79
duty-free-exports	117,81	111,83
anti-satellite-test-ban	114,65	111,31
religious-groups-in-schools	80,16	76,52
handicapped-infants	72,09	69,52
export-administration-act-south-africa	60,55	60,75
synfuels-corporation-cutback	59,23	57,29
immigration	3,05	3,06
water-project-cost-sharing	0,21	5,32

Iz tablice 7 možemo vidjeti da u rezultatima implementacije dolazi do razlike između vrijednosti koje daje alat *Rapid Miner*. Zanimljivo je primijetiti kako za postupak *ChiSquare* nema značajne razlike na značajci „export-administration-act-south-africa“ čiji postotak nepoznatih vrijednosti iznosi 24%, dok nad značajkom „water-project-cost-sharing“ dolazi do signifikantnog odstupanja gledajući omjer vrijednosti iako postotak nepoznatih vrijednosti iznosi 11%, što je znatno manje od prethodno komentirane značajke. Takvo ponašanje nam govori da ovaj postupak nije otporan na nepoznate vrijednosti i da je potrebno oprezno obrađivati rezultate koje isti daje.

Tablica 8 Postupak ChiSquare nad skupom podataka audiology

Značajke	CS_RM	CS
tymp	792	792
bser	337	224
air	254	254
bone	247	174
o_ar_c	246	175
history_heredity	226	226
with_nerve_signs	226	226
wave_V_delayed	226	226
late_wave_poor	226	226
middle_wave_poor	226	226
speech	214	180

age_gt_60	212	180
history_fluctuating	196	196
ar_c	192	146
airBoneGap	188	188

Iz tablice 8 možemo vidjeti da naša implementacije ima velika odstupanja u radu sa značajkama s visokim postotkom nepoznatih vrijednosti npr. „bone“ čiji postotak nepoznatih vrijednosti iznosi 33%. Zanimljivo je primijetiti kako postupak SU daje potpuno drugačiji poredak značajki. Treba primijetiti kako se ovdje na drugoj poziciji pojavila značajka „bser“ čiji postotak nepoznatih vrijednosti iznosi 98% dok je ta ista u prošlom postupku završila na predzadnjem mjestu. Iz toga možemo zaključiti da je SU puno robusniji na nepoznate vrijednosti od postupka Chi Square.

4.4 ReliefF

ReliefF je postupak odabira značajki korišten za izradu klasifikacijskog modela[11]. Postupak ReliefF je nadogradnja postojećeg postupka Relief postupka [9] koji je ograničen na klasifikacijske (binarne) probleme s dvije klase. ReliefF omogućuje korištenje postupka nad klasifikacijskim problemima s više klasa, a uz to podržava rad i s nepotpunim podatcima. Postoji nadogradnja ReliefF postupka RReliefF [2] za rješavanje regresijskih problema.

Ideja postupka ReliefF je predvidjeti značajnost značajki prema tome kako se dobro njihove vrijednosti razlikuju između primjera s istih i različitih klasa koje su „blizu“ jedna drugoj. Pseudokod postupka možemo vidjeti na slici 5.

Postupak ReliefF

Ulaz: skup podataka za treniranje

Izlaz: polje W procjene kvalitete značajki

1. postavi sve težine $W[A] = 0.0$;
2. za i=1 do m radi:
3. slučajno odaberi primjer R_i;
4. pronađi k najbližih pogodaka H_j;
5. za svaku klasu $C \neq class(R_i)$ radi
6. za klasu C pronađi k najbližih promašaja $M_j(C)$;
7. za A=1 do a radi:
8. $W[A] = W[A] - \sum_{j=1}^{k} \frac{diff(A,R_i,H_j)}{m*k} +$
9. $\sum_{C \neq class(R_i)} [\frac{P(C)}{1-P(class(R_i))} * \sum_{j=1}^{k} diff(A, R_i, M_j(C))]/(m * k)$;
10. kraj;

Slika 5. Pseudokod postupka ReliefF [2]

Rezultat provođenja postupka su težine pridružene svakoj značajci. Težine mogu biti pozitivne i negative. Postupak korekcije težina značajki se ponavlja m puta, gdje je m korisnički definiran parametar za koji je poželjno postaviti što većeg no sam taj parametar pridonosi složenosti postupa pa s njim treba oprezno postupati. Svakim korakom slučajno odabiremo primjerak ulaza R_i za koji tražimo njemu najbliže pogotke (najbliže primjerke iste klase) i promašaje (najbliže primjerke suprotne klase) metodom k-najbližih susjeda. Dozvoljavanjem korištenja podatkovnih skupova s više od dvije različite klase moramo tražiti najbliže promašaje za svaku moguću različitu klasu. Razliku između dva primjera I_1 i I_2 računamo formulom $diff(A, I_1, I_2)$ čiju definiciju možemo vidjeti na slici 6, gdje je A promatrana značajka (atribut). Doprinos pogodaka i promašaja računamo kao sumu razlika po svim atributima, osim same klase.

$$diff(A, I_1, I_2) = \begin{cases} 0; vrijednost(A, I_1) = vrijednost(A, I_2) \\ 1; inače \end{cases}$$

Slika 6. Funkcija diff [2]

Pri izračunu korekcije najbližih promašaja, moramo normalizirati svaku sumu razlika koju dobijemo kako bismo izbjegli privrženost vrijednostima klase koje poprima veći broj primjera. Važno je istaknuti zašto ispred sume po najbližim pogodcima stoji minus a ispred sume po najbližim promašajima plus. Ako primjerci R_i i H_j iste klase imaju drugačije vrijednosti tj. funkcija *diff* vraća pozitivan broj za značajku A, to znači da značajka A dijeli ova dva primjerka iako su oni iste klase što nam nije pogodno za predikciju klase pomoću značajke A. Međutim, ako primjerci R_i i M (iz skupa promašaja) imaju različite vrijednosti to nam govori da značajka A uspješno radi razliku između primjeraka različitih klasa što nam je poželjno svojstvo značajke. Složenost ovog algoritma je $O(m * N * A)$ gdje je m parametar zadan od strane korisnika, N broj primjeraka nad kojima provodimo postupak a A je broj značajki [2].

Implementacija

Za implementaciju istaknuo bih par važnih koraka: pronalazak *k*-najbližih pogodaka, pronalazak *k*-najbližih promašaja i korekcija težina atributa. Također potrebno je objasniti i algoritam *k*-najbližih susjeda (engl. *k-nearest neighbors algorithm – kNN algorithm*). Izračun *k*-najbližih pogodaka i promašaja svodi se na implementaciju istog algoritma uz različito zadanu traženu vrijednost klase.

```
private static List<KNNValue> findKNN(List<Instance> instances, Instance ins, String classLabel) {
    int classIndex = Instance.size - 1;

    List<KNNValue> values = new ArrayList<>();

    for (int j = 0, n = instances.size(); j < n; j++) {
        Instance i = instances.get(j);

        if(i == ins) continue;
        if(! i.getValueAt(classIndex).equals(classLabel)) continue;

        values.add(new KNNValue(distance(i, ins), j));
    }

    values.sort((a,b) -> a.getValue().compareTo(b.getValue()));

    return values;
}
```

Slika 7. Implementacija algoritma kNN

Implementaciju samog algoritma vidimo na slici 7. Algoritam se svodi na pronalazak i izračun udaljenosti primjerka s istom zadanom vrijednošću klase od zadanog primjerka. Funkcija *distance* računa udaljenost između dva primjerka sumirajući za svaki atribut rezultate funkcije *diff* (slika 6). Uzlaznim sortiranjem očitanih susjeda dobivamo rang listu najbližih susjeda iz koje odabiremo do *k* primjeraka pri daljnjim izračunima. Ne ograničavanjem broja različitih vrijednosti klase potrebno je izračunati najbliže promašaje za svaku različitu klasu.

```
// randomly select an instance Ri
Instance Ri = instances.get(randomIns);

// HITS
List<KNNValue> nearestHits = findKNN(instances, Ri, Ri.getValueAt(Instance.size-1));


List<List<KNNValue>> misses = new ArrayList<>();

// MISSES
for(int j = 0, n = classAtr.getLabels().size(); j < n; j++) {
    String label = classAtr.getLabels().get(j);
    if(label.equals(Ri.getValueAt(Instance.size-1))) continue;

    List<KNNValue> nearestMisses = findKNN(instances, Ri, label);
    misses.add(nearestMisses);
}
```

Slika 8. Implementacija pronalaska susjeda

Nakon pronalaska *k*-najbližih pogodaka i promašaja došli smo do sedme linije pseoudokoda (slika 8) tj. idući korak je dorada težina svih atributa prema izračunatim susjedima.

```
for(int attrNo = 0, n = data.getAttributesNumber()-1; attrNo < n; attrNo++) {
    int K = nearestHits.size();
    if(K >= K_MAX) K = K_MAX;

    double sumNearestHitsDiff = 0.0;
    for(int f = 0; f < K; f++) {
        sumNearestHitsDiff += diff(Ri, instances.get(nearestHits.get(f).getIndex()), attrNo);
    }

    if(K > 0.0) {
        sumNearestHitsDiff /= (K*instancesNum);
    }
```

Slika 9. Doprinos k-najbližih pogodaka

Izračun doprinosa *k*-najbližih susjeda vidimo na slici 9. Potrebno je istaknuti da u nekim slučajevima ne moramo imati najbliže susjede pa se izračun doprinosa preskače ili možemo imati više od *k* najbližih susjeda pa odabiremo prvih *K_MAX* što je u našem slučaju 10. Izračun doprinosa promašaja za svaku različitu klasu možemo vidjeti na slici 10. I u ovom slučaju potrebno je raditi korekciju ako ne postoje najbliži

promašaji ili ako ih postoji više od *K_MAX*. Također potrebno je istaknuti da se ukupni doprinos računa kao suma doprinosa svake različite klase korigirane izračunatim faktorom *p*.

```
    double sumPerClass = 0.0;

    for(int j = 0, c = 0; j < classAtr.getLabels().size(); j++) {
        String label = classAtr.getLabels().get(j);
        if(label.equals(Ri.getValueAt(Instance.size-1))) continue;

        double sumNearestMiss = 0.0;

        List<KNNValue> missesForClass = misses.get(c++);

        K = missesForClass.size();
        if(K >= K_MAX) K = K_MAX;

        for(int f = 0; f < K; f++) {
            sumNearestMiss += diff(Ri, instances.get(missesForClass.get(f).getIndex()), attrNo);
        }

        double p = ((FeatureSelectionMethods.prob(label,data))/(1-FeatureSelectionMethods.prob(Ri.getValueAt(Instance.size-1),data)));
        sumNearestMiss *= p;

        if(K > 0) {
            sumNearestMiss /= (K*instancesNum);
        }

        sumPerClass += sumNearestMiss;
    }

    weights[attrNo] = weights[attrNo] - sumNearestHitsDiff + sumPerClass;
}
```

Slika 10. Doprinos k najbližih promašaja

Usporedba s gotovom implementacijom

Implementaciju postupka *ReliefF* uspoređivao sam s implementacijom alata *Weka*.

Tablica 9. Postupak ReliefF nad skupom podataka weather.nominal

Značajke	RelifF_W	RelifF
outlook	0,13571	0,13571
humidity	0,05476	0,05476
windy	-0.00238	-0.00238
temperature	-0.08333	-0.08333

Iz tablice 9 možemo vidjeti da naša implementacija daje identične rezultate za podatke bez nepoznatih vrijednosti.

Tablica 10. Postupak ReliefF *nad skupom podataka vote*

Značajke	RelifF_W	RelifF
physician-fee-freeze	0,6734	0,677011
crime	0,3	0,297356
synfuels-corporation-cutback	0,2756	0,283678
adoption-of-the-budget-resolution	0,2548	0,249425

el-salvador-aid	0,1971	0,196437
duty-free-exports	0,1898	0,198161
immigration	0,1879	0,186092
education-spending	0,1695	0,162414
water-project-cost-sharing	0,1228	0,124368
superfund-right-to-sue	0,111	0,114713
handicapped-infants	0,1087	0,103563
religious-groups-in-schools	0,1075	0,108391
mx-missile	0,0994	0,093563
aid-to-nicaraguan-contras	0,0686	0,069885
anti-satellite-test-ban	0,0599	0,059770
export-administration-act-south-africa	0,0511	0,052529

Iz tablice 10 možemo vidjeti da dolazi do neznatnog odstupanja između rezultata. Zanimljivo je primijeti kako za postupak *ReliefF* nema značajne razlike na značajci „export-administration-act-south-africa“ čiji postotak nepoznatih vrijednosti iznosi 24%, dok nad značajkom „duty-free-exports“ dolazi do odstupanja koje dovodi do razlike u rangiranju samih značajki, iako značajka sadrži tek 6% nepoznatih vrijednosti. Uzrok takvih razlika može biti u drugačijem tretiranju nepoznatih vrijednosti pri izračunu korekcijskog faktora *p*. Još jedna zanimljivost na koju treba obratiti pažnju jest utjecaj nepoznatih vrijednosti na pronalazak najbližih susjeda. Više nam nepoznate vrijednosti ne utječu samo na rezultat same značajke kao u prethodnim postupcima nego i na pronalazak najbližih susjeda.

Tablica 11. Postupak ReliefF nad skupom podataka audiology

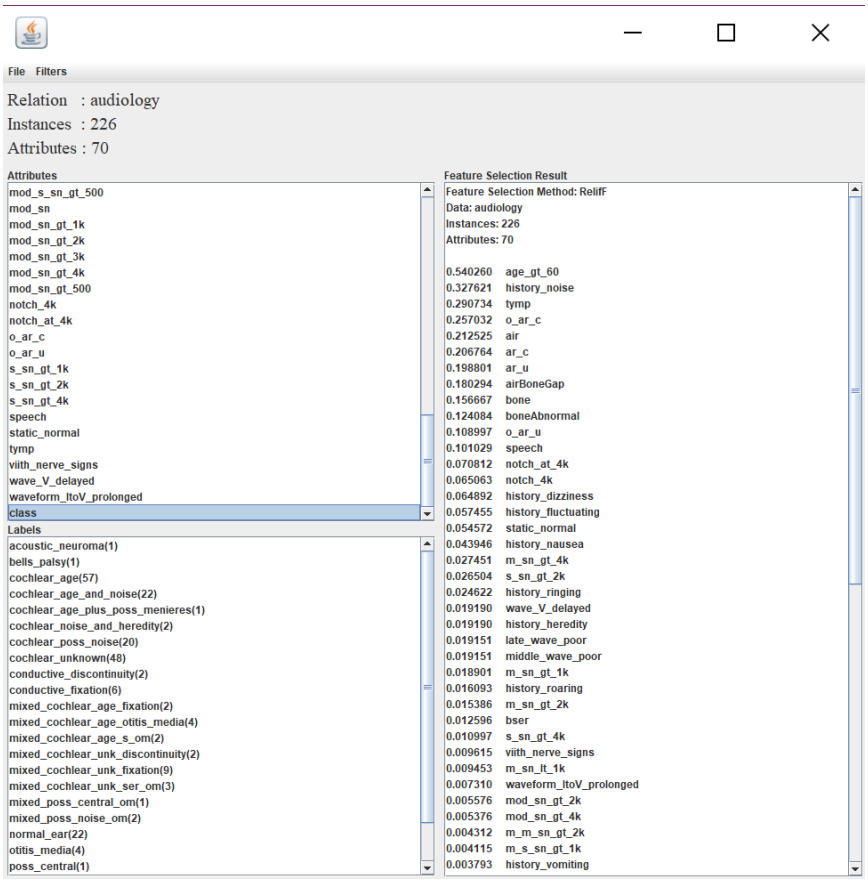
Značajke	RelifF_W	RelifF
age_gt_60	0,5402	0,540260
history_noise	0,3278	0,327621
tymp	0,2903	0,290734
o_ar_c	0,2572	0,257032
air	0,2173	0,212525
ar_c	0,2096	0,206764
ar_u	0,1971	0,198801
airBoneGap	0,1802	0,180294
bone	0,1563	0,156667
boneAbnormal	0.1220	0,124084
o_ar_u	0,1041	0,108997
speech	0,0970	0,101029
notch_at_4k	0,0705	0,070812
notch _4k	0,0700	0,065063
history_dizziness	0,0652	0,064892

Iz tablice 11 možemo vidjeti da naša implementacije gotovo pa i nema odstupanja čak niti sa značajkama s visokim postotkom nepoznatih vrijednosti npr. „bone“ čiji postotak nepoznatih vrijednosti iznosi 33%. Među najboljih 15 se ne pojavljuje značajka „bser“, za razliku od *ChiSquare* postupka, čiji postotak nepoznatih vrijednosti iznosi 98%. Iz toga možemo zaključiti da je *ReliefF* također robusniji na nepoznate vrijednosti od postupka *Chi Square*.

5. Razvijeni program

Razvijeno je jednostavno sučelje za korištenje implementacija tri prethodno opisana postupka. Sučelje omogućuje učitavanje podataka isključivo u formatu *.arff* , a uz to podatci moraju biti nominalnog tipa, tj. nije podržan rad i učitavanje podataka kojima su brojevi domena vrijednosti značajka. Sučelje također daje osnovne informacije o učitanom podatkovnom skupu kao što su ime, broj primjera i broj atributa. Moguće je vidjeti koje vrijednosti pojedini atribut može poprimiti. Također, omogućeno je provođenje sva tri implementirana postupka nad podatcima. Rad programa je ispitan isključivo na prethodno opisanim skupovima podataka. Nakon odabira željenog postupka dobiva se lista značajki s pridruženim težinama, značajke su poredane silazno, izračunatim odabranim postupkom.

Izgled sučelja nakon provedenom postupka *ReliefF* može se vidjeti na slici 11. Program se pokreće izvršavanjem metode *main* razreda *App.*



Slika 11. Grafičko sučelje programa

6. Zaključak

U budućnosti možemo očekivati sve veći broj izvora visoko dimenzijskih podataka, što dovodi do sve veće potrebe za istraživanjem i osmišljavanjem novih postupaka odabira značajki. Kao odgovor na pitanje zašto koristiti postupke odabira značajki istaknuo bih tri točke:

- Poboljšanje razumijevanja pozadinskog procesa koji generira podatke
- Poboljšanje točnosti modela predikcije
- Smanjenje vremena potrebnog za izgradnju modela

Potrebno je istaknuti da korištenje nekog postupka odabira značajki ne povlači ostvarenje svih triju točaka. Kao što smo mogli vidjeti na primjerima implementacija filtarskih postupaka, svaki od njih je specifičan i zasnovan na različitim znanjima. Nekada smo u situaciji da provedbom postupka dobijemo manju točnost, ali puno bolje vrijeme izgradnje modela. Nekada nam je preskupo očitavati sve značajke ovisno o prirodi problema. Zbog svih ovih pojava, ne možemo reći da je neki postupak taj kojeg uvijek trebamo koristiti. Potrebno je razumjeti domenu problema i napraviti kompromis, ako ne uspijemo ostvariti sve tri navedene točke.

Ako se osvrnemo na poruku „No free lunch“ teorema i različita znanja na kojima su postupci zasnovani, kao odgovor na pitanje koje postupke koristiti, rekao bih da treba provesti sve i izabrati onaj koji daje najbolji rezultat za specifičnost našeg problema. Provedbom implementiranih postupaka nad istim podatcima možemo uočiti da svaki od njih daje drugačije rezultate. Zaključiti koji je podskup značajki najkorisniji možemo tek kada uparimo postupak s klasifikatorom i vrednujemo uspješnost predikcije dobivenog modela.

LITERATURA

[1] A. Jović, K. Brkić i N. Bogunović. *A review of feature selection methods with applications*. 2015

[2] M. Robnik-Šikonja i I. Kononenko. *Theoretical and Empirical Analysis of ReliefF and RReliefF*. Machine Learning Journal (2003)

[3] S. Alelyani, J. Tang i H. Liu. *Feature Selection for Classification: A Review*

[4] L. Yu i H. Liu. *Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution*. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

[5] R. Kohavi i G. John. *Wrappers for feature subset selection.* Artificial Intelligence, 1997, 273–324.

[6] M. Hall. *Correlation-based feature selectionfor discrete and numeric class machine learning*. Proceedings of the Seventeenth International Conference on Machine Learning 2000, 359–366.

[7] R. E. Walpole, R. H. Myers, S. L. Myers, K. Ye. *Probability & Statistics for Engineers & Scientists N I N T H E D I T I O N: 10.13 Test for Homogeneity* 2007, 376-382 .

[8] Jason Brownlee, *An Introduction to Feature Selection*, 6.10.2014, http://machinelearningmastery.com/an-introduction-to-feature-selection/, 1.6.2017.

[9] K. Kira i L. A. Rendell: 1992b. *A practical approach to feature selection*. Iz D.Sleeman and P.Edwards: Machine Learning: Proceedings of International Conference (ICML'92). 249–256, Morgan Kaufmann.

[10] *Contingency table*, Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Contingency_table

[11] I. Kononenko. *Estimating attributes: analysis and extensions of Relief*. Iz L. De Raedt and P.Edwards: Machine Learning: ECML-94. 171–182, Springer Verlag.

Postupci odabira značajki

Sažetak

U ovom radu opisana su tri filtarska postupka odabira značajki: symmetrical uncertainty, ReliefF i ChiSquare. Pokušali smo odgovoriti na pitanja kako, zašto i kada koristiti postupke. Za svaki postupak opisana je teorijska podloga, dan je primjer korištenje ili objašnjenje pseudokoda u svrhu shvaćanja i ispravne interpretacije dobivenih rezultata provedbom postupaka.

Osim teorijskog objašnjenja načina rada postupaka provedena je implementacija triju postupaka. Implementirani postupci su testirani nad kategorijskim podatcima i rezultati su uspoređeni s već gotovim alatima koji nude iste postupke.

Ključne riječi: postupci odabira značajki, symmetrical uncertainty, ReliefF, ChiSquare, Weka, RapidMiner, filtarski postupci.

Feature Selection Methods

Abstract

This paper describes three filter feature selection methods: symmetrical uncertainty, ReliefF, ChiSquare. We tried to answer questions like: how, why and when to use these methods. We described the theoretical background of each method and gave a usage example or explanation of the pseudocode for understanding and correct interpretation of the results obtained by performing these methods.

In addition to theoretical explanation, these three methods were also implemented. The implemented methods were tested on the categorical data and the results were compared with available tools that offer implementation of these methods.

Keywords: feature selection methods, symmetrical uncertainty, RelifF, ChiSquare, Weka, RapidMiner, filter methods.