Implementacija i evaluacija algoritama mašinskog učenja za filtriranje neželjene elektronske pošte

*Master rad*

Mentor:  dr Filip Marić  
    Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:  dr Mladen Nikolić  
    Univerzitet u Beogradu, Matematički fakultet

    dr Jelena Graovac  
    Univerzitet u Beogradu, Matematički fakultet

Datum odbrane:  ______________________
Rezime: Slanje neželjene elektronske pošte (eng. spam e-mail) značajno je intenzivirano tokom poslednjih 25 godina. Mane u protokolima za slanje i ubrzan razvoj elektronskog biznisa i finansijskih transakcija direktno utiču na uvećanje pretnji putem elektronske pošte. Neželjena elektronska pošta danas predstavlja jedan od glavnih problema na internetu donoseći gubitke finansija i vremena kompanijama i individualnim korisnicima.

Većina postojećih istraživanja na polju suzbijanja neželjene e-pošte fokusira se na dizajn protokola, metode autentifikacije i tehnike filtriranja. Među pristupima koji su razvijeni najvažnija je tehnika filtriranja. U poslednje vreme mašinsko učenje za klasifikaciju neželjene pošte je veoma važno istraživačko pitanje. Da bi se napravio dobar filter neželjene elektronske pošte, analiziraju se reči koje se pojavljuju u tekstu poruke, njihova učestanost, elektronska adresa pošaljoca, kao i mnogi drugi podaci. U procesu učenja učestvuju dobromamerni korisnici koji mogu klasifikovati određene poruke kao spam i ukoliko se te iste poruke pošalju na druge adrese u okviru istog mejl servera filter neželjene elektronske pošte će ih sam klasifikovati kao spam. Eфikasan sistem filtriranja e-pošte zahteva preciznost (nizak stepen lažno pozitivnih i lažno negativnih instanci), mogućnost samorazvijanja (sistem koji imaju mogućnost da se adaptiraju na nove neželjene poruke) i visoke performanse (detekcija novih neželjenih poruka mora se detektovati brzo).

Cilj rada je implementacija i evaluacija nekih algoritama mašinskog učenja (Bajesovo filtriranje, metoda neuronskih mreža, metoda podržavajućih vektora i slično) za klasifikaciju i filtriranje neželjene pošte i njihova evaluacija na nekoliko korpusa elektronske pošte.

Ključne reči: klasifikacija tekstualnih dokumenata, filtriranje neželjene elektronske pošte, algoritmi mašinskog učenja, Naivni Bajesov algoritam, veštacke neuronske mreže (ANN), metod podržavajućih vektora (SVM), k najbližih suseda (kNN)
Sadržaj

1 Uvod 1
1.1 E-mail komunikacija 2
1.2 Problem neželjene elektronske pošte 3

2 Klasifikacija 6
2.1 Klasifikacija tekstualnih dokumenata 6
2.2 Mere kvaliteta i tehničke evaluacije klasifikatora 7
   2.2.1 Unakrsna validacija 10

3 Algoritmi filtriranja elektronske pošte 11
3.1 Bajesovo filtriranje 11
   3.1.1 Matematička osnova 11
   3.1.2 Kombinovanje individualnih verovatnoća 13
   3.1.3 Rad sa retkim rečima 14
   3.1.4 Druge heuristike 14
   3.1.5 Primena 15
3.2 K najbližih suseda 17
   3.2.1 Euklidska metrika 19
   3.2.2 Pretraga prostora rešenja 20
   3.2.3 Udaljenost instanci 20
   3.2.4 Sum u podacima za učenje 21
3.3 Metod podržavajućih vektora (SVM) 22
   3.3.1 Linearno razdvojive klase 22
   3.3.2 Linearno nerazdvojive klase 23
   3.3.3 Upotreba kernela 25
3.4 Vestačke neuronske mreže 26
   3.4.1 Perceptron 26
   3.4.2 Višeslojni perceptron 28
   3.4.3 Karakteristike vestačkih neuronskih mreža 29
3.5 Tehnika maksimalne entropije 30

4 Podaci 31
4.1 Ling spam korpus 31
4.2 PU1, PU2, PU3 i PUA spam korupsi 31
4.3 Enron spam korpus 32

5 Implementacija 33
5.1 Biblioteka SVMlight 35
5.2 Kombinovanje klasifikatora 36

6 Eksperimentalni rezultati 37
6.1 PU1 korpus 37
   6.1.1 Naivni Bajesov klasifikator 37
   6.1.2 K najbližih suseda 38
   6.1.3 Metod podržavajućih vektora 39
   6.1.4 Perceptron 40
   6.1.5 Kombinovanje klasifikatora 40
6.2 PU2 korpus ................................................. 42
6.3 PU3 korpus ................................................. 44
6.4 PUA korpus ................................................ 46
6.5 Ling spam korpus ........................................... 48
6.6 Enron spam korpus ......................................... 50
  6.6.1 enron1 .................................................. 50
  6.6.2 enron2 .................................................. 52
  6.6.3 enron3 .................................................. 54
  6.6.4 enron4 .................................................. 56
  6.6.5 enron5 .................................................. 58
  6.6.6 enron6 .................................................. 60

7 Zaključak ................................................. 62

Bibliografija ................................................ 63

A Dodatak .................................................. 65
  A.1 FileUtil.h ................................................ 65
  A.2 Instance.h ............................................... 66
  A.3 IzdvajanjeOdlika.h ..................................... 68
  A.4 Izuzetak.h ............................................... 70
  A.5 Klasifikator.h ......................................... 71
  A.6 Klasifikator1od2.h ....................................... 73
  A.7 Klasifikator2od3.h ....................................... 74
  A.8 KlasifikatorKNajblizihSuseda.h ...................... 75
  A.9 NaivniBajesovKlasifikator.h ......................... 77
  A.10 PerceptronKlasifikator.h ............................. 79
  A.11 SVMKlasifikator.h ..................................... 80
  A.12 main.cpp ............................................... 83
Spiskovi

Algoritmi

1 Trening algoritam za Bajesov klasifikator .................................................. 16
2 Klasifikacija zasnovana na Bajesovom klasifikatoru ..................................... 16
3 K najближih suseda ......................................................................................... 18
4 Treniranje perceptrona .................................................................................... 27
5 Klasifikacija perceptron pravilom ................................................................. 27

Tabele

1 Korporativni naspram korisničkih e-mail naloga u periodu od 2011. do 2015. .................. 2
2 Korporativna primljena i poslata e-pošta jednog korisnika po danu u periodu od 2011. do 2015. .......... 3
3 Raspodjela spam poruka .................................................................................. 4
4 Matrica konfuzije za problem 2 klase (binarna klasifikacija) ......................... 8
5 Raspodjela poruka u korpusima ..................................................................... 31
6 Parametri programa ...................................................................................... 34
7 Matrica konfuzije PU1 korpusa prilikom klasifikacije Naivnim Bajesovim klasifikatorom za $\lambda = 5$ .......... 37
8 Testiranje performansi klasifikatora PU1 korpusa Naivnim Bajesovim klasifikatorom .......... 37
9 Matrica konfuzije PU1 korpusa prilikom klasifikacije algoritmom K najближих suseda za $k = 50$ i $l = 30$ ................... 38
10 Testiranje performansi klasifikatora PU1 korpusa algoritmom K najближих suseda .......... 38
11 Matrica konfuzije PU1 korpusa prilikom klasifikacije Metodom podržavajućih vektori bez meke margini ................................................................. 39
12 Matrica konfuzije PU1 korpusa prilikom klasifikacije Metodom podržavajućih vektori sa mekom marginom .............................. 39
13 Testiranje performansi klasifikatora PU1 korpusa Metodom podržavajućih vektori sa i bez meke margini ................ 39
14 Matrica konfuzije PU1 korpusa prilikom klasifikacije korišćenjem perceptrona .............. 40
15 Testiranje performansi klasifikatora PU1 korpusa korišćenjem perceptrona .......... 40
16 Testiranje performansi kombinovanih klasifikatora nad PU1 korpusom ................. 40
17 Rezultati testiranja implementiranih klasifikatora nad PU2 korpusom .................. 42
18 Rezultati testiranja implementiranih klasifikatora nad PU3 korpusom .................. 44
19 Rezultati testiranja implementiranih klasifikatora nad PUA korpusom .................. 46
20 Rezultati testiranja implementiranih klasifikatora nad Ling spam korpusom ............... 48
21 Rezultati testiranja implementiranih klasifikatora nad enron1 korpusom .................. 50
22 Rezultati testiranja implementiranih klasifikatora nad enron2 korpusom ................. 52
23 Rezultati testiranja implementiranih klasifikatora nad enron3 korpusom ................. 54
24 Rezultati testiranja implementiranih klasifikatora nad enron4 korpusom .................. 56
25 Rezultati testiranja implementiranih klasifikatora nad enron6 korpusom .................. 58
26 Rezultati testiranja implementiranih klasifikatora nad enron6 korpusom ................. 60
Slike

1. Primer jednog dela zaglavlja poruke
2. Primer jednog dela tela poruke
3. Primer koji ilustruje algoritam k najbližih suseda za k = 1, k = 2, k = 3
4. Primer koji ilustruje optimalnu hiper-ravan sa maksimalnom marginom koja razdvaja podatke za trening u 2 klase
5. Preslikavanje u višedimenzionalni prostor u kome je skup podataka linearno razdvojen
6. Primer koji ilustruje dve linearno nerazdvojive klase
7. Perceptron kao neuron
8. Struktura višeslojnog perceptrona
9. Primer kodirane poruke iz PU1 korpusa
10. Dijagram klasa 1. deo
11. Dijagram klasa 2. deo
12. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU1 korpusom
13. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU1 korpusom
14. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU2 korpusom
15. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU2 korpusom
16. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU3 korpusom
17. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU3 korpusom
18. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PUA korpusom
19. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PUA korpusom
20. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad Ling spam korpusom
21. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad Ling spam korpusom
22. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron1 korpusom
23. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron1 korpusom
24. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron2 korpusom
25. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron2 korpusom
26. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron3 korpusom
27. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron3 korpusom
28. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron4 korpusom
29. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron4 korpusom
30. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron5 korpusom
31. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron5 korpusom
32. Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron6 korpusom
33. Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron6 korpusom
1 | Uvod

Elektronska pošta (e-mail) je mrežni servis koji omogućava slanje i primanje poruka raznovrsnog sadržaja. Ime predstavlja analogiju tradicionalnoj pošti, pri čemu poštansko sanduče zamenjuju serveri na kojima se pošta čuva dok je korisnik ne preuzme. Programi za rad sa elektronskom poštom se zasnivaju na uređivaču teksta za sastavljanje poruka. U oktobru 1971. (mada ima izvora koji tvrde da se to dogodilo jula 1970.) stručnjak za računarstvo Rej Tomlinson je napisao prvi program za razmerno poruka između dva računara. Pored programa za razmenu e-pošte, Rej Tomlinson je odgovoran za masovnu upotrebnu znaku @ (et, poznato i kao ludo a ili majmunsko a). Smisljavajući kako da razvrstaju primioci poruka odlučio je da njihova imena i imena računara na kojima se nalaze njihovi nalozi razdvojiti nekim znakom interpunkcije. Kako je na svojoj tastaturi imao samo 12 takvih znakova na raspolaganju, odlučio se za onaj koji se nikada ne koristi u pisanju poruka. Taj princip adresiranja koristi se i danas. Svaka poruka elektronske pošte se sastoji iz tela i zaglavlja. Zaglavlje nosi sledeće podatke:

- adresu i ime pošaljioca
- adresu primaoca
- adresu servera preko kojeg je poruka poslata kao i servera koji su prosleđivali poruku na njenom putu do odredišta
- datum slanja
- ime programa koji je korišćen za slanje poruke
- prioritet

Svaka stavka zaglavlja se sastoji od imena stavke i vrednosti stavke, razdvojenih dvotačkom.

Slika 1: Primer jednog dela zaglavlja poruke

Telo poruke se može sastojati iz više delova u zavisnosti od toga da li se sa tekstim poruke šalju i datoteke. Ukoliko se šalju onda se u samom zaglavlju poruke to može označiti pomoću stavke sledećeg oblika:

---

1
Slika 2: Primer jednog dela tela poruke

Korišćenje elektronske pošte je ugroženo od strane četiri pojava: bombardovanja porukama, spamom, pokušajima preuzimanja ličnih podataka i prenošenjem virusa. Na internetu termin *spam* označava nepoželjnu, besciljnu (untargeted) elektronsku poštu. U osnovi, to je slanje komercijalnih poruka, najčešće reklama i mrežnih marketing šema[1] na stotine hiljada pa i miliona adresa korisnika širom mreže, bez njihovog odobrenja. Blaži oblik spama predstavlja TDEM (*Targeted Direct Email Marketing*) ili direktni marketing putem ciljnih grupa. Bez obzira u koju vas grupu budu stavili nemilosrdni spameri, jedno je sigurno — vaše e-mail sanduče svakoga dana biće bombardovano desetinama beskorisnih poruka.

1.1 E-mail komunikacija


U 2011. godini 25% naloga elektronske pošte na internetu činili su korporativni e-mail nalozi. U naredne četiri godine očekuje se brži tempo rasta korporativnih e-mail naloga zbog sve pristupačnijih e-mail servisa koji su bazirani na računarstvu u oblaku. Mnoge korporacije koriste e-mail servise računarstva u oblaku kao način za proširivanje usluga elektronske pošte radnicima koji možda nisu imali pristup elektronskoj pošti u prošlosti.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Broj e-mail naloga na internetu u milionima</td>
<td>3146</td>
<td>3375</td>
<td>3606</td>
<td>3843</td>
</tr>
<tr>
<td>Broj korporativnih e-mail naloga u milionima</td>
<td>788</td>
<td>850</td>
<td>918</td>
<td>991</td>
</tr>
<tr>
<td>Broj korporativnih e-mail naloga u %</td>
<td>25%</td>
<td>25%</td>
<td>25%</td>
<td>26%</td>
</tr>
<tr>
<td>Broj korisničkih e-mail naloga u milionima</td>
<td>2358</td>
<td>2525</td>
<td>2688</td>
<td>2852</td>
</tr>
<tr>
<td>Broj korisničkih e-mail naloga u %</td>
<td>75%</td>
<td>75%</td>
<td>75%</td>
<td>74%</td>
</tr>
</tbody>
</table>

Tabela 1: Korporativni naspram korisničkih e-mail naloga u periodu od 2011. do 2015.

Problem neželjene elektronske pošte

Prosečni korporativni e-mail korisnik prini oko 105 e-mail poruka dnevno. Uprkos spam filterima, otprilike 19% elektronske pošte koja je dostavljena u prijemno sanduče je neželjena pošta. Brzina rasta poslatih i primljenih e-mail poruka postepeno usporava zbog ubrzanog rasta drugih vidova komunikacije poput instant poruka i socijalnih mreža. Prosečni godišnji rast instant poruka iznosi oko 11% i očekuje se da broj naloga u 2015. godini premaši cifru od 3.8 milijardi korisnika. Socijalne mreže takođe beleže ubrzan rast kako kod potrošačkih tako i kod korporativnih korisnika. U 2011. ukupan broj naloga na socijalnim mrežama iznosio je oko 2.4 milijarde dok se u 2015. očekuje da taj broj naraste do 3.9 milijardi [24].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prosečan broj primljenih i poslatih e-mailova jednog korisnika po danu</td>
<td>105</td>
<td>110</td>
<td>115</td>
<td>120</td>
</tr>
<tr>
<td>Prosečan broj primljenih e-mailova</td>
<td>72</td>
<td>75</td>
<td>78</td>
<td>81</td>
</tr>
<tr>
<td>Prosečan broj legitimnih e-mailova</td>
<td>58</td>
<td>62</td>
<td>65</td>
<td>68</td>
</tr>
<tr>
<td>Prosečan broj neželjene pošte</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>Prosečan broj poslatih e-mailova</td>
<td>33</td>
<td>35</td>
<td>37</td>
<td>39</td>
</tr>
</tbody>
</table>


1.2 Problem neželjene elektronske pošte

U posljednje vreme neželjena pošta postala je veliki problem na internetu i predstavlja gubljenje vremena, prostora za skladištenje i protoka informacija. Problem spama se godinama uvećava. Po skorašnjim statistikama oko 15 milijardi mejlova dnevno su neženjena pošta i koštaju internet korisnike oko 355 miliona dolara godišnje. Spameri su počeli da koriste nekoliko trikova kako bi zaobišli metode filtriranja korišćenjem različitih adresa slanja i različitih karaktere kojim započinju i završavaju naslov poruke [14]. Tradicionalni sistemi za otkrivanje neželjene pošte koji koriste sisteme bazirane na rečima su lako poraženi jer su spameri pronašli nove načine za reprezentovanje reči. Na primer reč hipoteka se može zapisati kao h-i-p-o-t-e-k-a ili kao h i p o t e k a i biće neregistrovana od strane ovakvih sistema. Zbog toga bi se svaka detektovana modifikovana reč morala dodavati u bazu. Fundamentalna potreba bilo kog filtera neželjene pošte je da nikada ne označi dobru poruku kao neželjenu čak i ako to znači neotkrivanje nekoliko neželjenih poruka.

Azija je odgovorna za 55,5 % spamova u svetu u 2013. godini, prati je Severna Amerika sa 19 %, dok je na trećem mestu Istočna Evropa sa 13,3 %. Srbija se nalazi na 27. mestu globalne liste zemalja koje su izvori spama sa 0,47 %, dok su vodeće zemlje Kina sa 23 % i Sjedinjene Američke Države sa 18 %, pokazuje istraživanje kompanije Kaspersky lab. Udeo Istočne Evrope se skoro udvostručio u odnosu na prethodnu godinu, stavljajući ovu oblast na treće mesto sa 13,3 %, Zapadna Evropa ostaje na četvrtom mestu uprkos smanjenju od 2,4 %, dok je udeo Latinske Amerike na petom mestu sa trostrukim padom u poređenju sa 2012. godinom.

Internet je prepun ljudi koji žele da nas prevare, a jedan od najpopularnijih načina je takozvana *Nigerijska prevara*[^1]. U pitanju je e-mail u kojem piše da ste navodno dobili ogroman novac, iako ni sami ne znate osobu koja vam je to saopštila. Prvi talas ovih prevara potekao je iz Nigerije, pa je po tome i dobila ime. Prevara počinje tako što vas nepoznate osobe kontaktiraju putem e-maila nudeći ogromnu sumu novca koji je ste navodno dobili ili nasledili. Kako biste dobili nasledstvo ili novac koji žele da vam pošalju, od vas će tražiti da platite razne takse ili da im date informacije o vašim računima. U početku je to mala količina novca, kako bi sve delovalo uverljivo, ali ukoliko nasednete na ovo, sigurno će izmisliti još nešto što mora da se plati kako bi dobili svoju nagradu.

<table>
<thead>
<tr>
<th>Kategorija</th>
<th>Udeo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proizvodi</td>
<td>25 %</td>
</tr>
<tr>
<td>Finansijski</td>
<td>20 %</td>
</tr>
<tr>
<td>Za odrasle</td>
<td>19 %</td>
</tr>
<tr>
<td>Prevara</td>
<td>9 %</td>
</tr>
<tr>
<td>Zdravlje</td>
<td>7 %</td>
</tr>
<tr>
<td>Internet</td>
<td>7 %</td>
</tr>
<tr>
<td>Opuštanje</td>
<td>6 %</td>
</tr>
<tr>
<td>Duhovni</td>
<td>4 %</td>
</tr>
<tr>
<td>Ostali</td>
<td>3 %</td>
</tr>
</tbody>
</table>

**Tabela 3:** Raspodela spam poruka

[^1]: Više o *Nigerijskoj prevari* možete pročitati na sajtu [MUP-a](http://www.mup-a.com).

Jedan od novijih vidova spama jeste spam putem internet pretraživača. Sajtovi koji pretražuju internet, kao što su Google, Yahoo, Microsoft-ov Bing, se danas susreću sa novim poteškoćama. Kako bi neki pretraživač mogao da zna koji je sajt popularniji, kvalitetniji, koristi se poznati *PageRank* algoritam za rangiranje i njegove modifikacije. Sam algoritam se oslanja na broj linkova koji pokazuje na određenu internet stranicu, smatrajući stranicu na koju pokazuje veći broj linkova popularnijom, što je u prvim godinama interneta zaista i važilo. Poznavajući tu metodologiju, spameri su se dosetili i

Saveti za zaštitu od neželjenih poruka:

1. Ne otvarajte neželjene poruke, izbrišite ih bez prethodnog otvaranja. Otvaranjem poruke obaveštavate pošiljaoca da vaša e-mail adresa postoji.

2. Nikada ne odgovarajte na spam poruke bez obzira što se od vas traži jer u suprotnom na ovaj način obaveštavate pošiljaoca o validnosti vaše e-mail adrese.

3. Ne kupujte proizvode koji se reklamiraju putem spam poruka bez obzira koliko povoljni bili. Ukoliko kupite proizvod očekujte još više reklamnih poruka u vašem poštanskom sandučetu.


Prema podacima FBI, u 2013. godini je ukupni gubitak od svih internet prevara bio skoro 782 miliona dolara [9]. Broj prijava koje je ova agencija prikupila u 2013. godini iznosio je 262.813, a najviše su bili pogođeni ljudi srednjih godina. Ovo i nije tako neuobičajeno, jer generalno gledano, ta generacija je odrastala bez interneta, pa većina ne zna kakve prevare sve postoje.
2 Klasifikacija

Zadatak klasifikacije je da objektima dodeli jednu od nekoliko predefinisanih kategorija. Ulazni podatak u proces klasifikacije je skup slogova koji se nazivaju instance. Slogovi su oblika $(x, y)$ gde je $x$ vektor vrednosti nekih atributa a, $y$ atribut koji određuje oznaku klase. Ulazni podaci se dele na podatke za učenje i testiranje. Na osnovu podataka za učenje pronalazi se klasifikator koji svaki vektor vrednosti nekih atributa $x$ preslikava u neku predefinisanu kategoriju $y$. Cilj je da klasifikacija bude što preciznija, a određivanje tačnosti modela se postiže na osnovu podataka za testiranje korišćenjem neke od mera evaluacije.

Klasifikacija nekog objekta se zasniva na pronalaženju sličnosti sa unapred određenim objektima koji su pripadnici različitih klasa, pri čemu se sličnost dva objekta određuje analizom njihovih karakteristika. Pri klasifikaciji se svaki objekt svrstava u neku od klasa sa određenom verovatnoćom. Zadatak je da se na osnovu karakteristika objekata čija klasifikacija je unapred poznata, napravi model na osnovu koga će se vršiti klasifikacija novih objekata. U problemu klasifikacija, broj klasa je unapred poznat i ograničen.

**Definicija:** Klasifikacija je proces učenja **ciljne funkcije** $f$ koja predefinisane klase $y$ pridružuje vektoru vrednosti atributa $x$ [27].

Ciljna funkcija je takođe poznata i kao **klasifikacioni model**. Klasifikacioni model je koristan za sledeće namene.

**Deskriptivno modelovanje.** Predstavlja glavne karakteristike podataka. U suštini rezimira podatke i omogućava nam da proučavamo najvažnije aspekte podataka bez uziimanja u obzir veličine skupa podataka.

**Prediktivno modelovanje.** Ima specifičan cilj da nam omogući da predvidimo vrednosti nekih ciljanih karakteristik objekta na bazi posmatranja vrednosti drugih karakteristika objekta.

2.1 Klasifikacija tekstualnih dokumenata

Automatska klasifikacija tekstualnih dokumenata je važna istraživačka tema još od prihvatanja digitalizacije dokumenata. Intuitivno, klasifikacija teksta je zadatak klasifikovanja dokumenta u jednu ili više predefinisanih kategorija. Formalnije, ako je $d_i$ dokument iz skupa dokumenata $D$ i $\{c_1, c_2, ..., c_n\}$ je skup svih kategorija, tada je klasifikacija teksta dodeljivanje jedne ili više kategorija $c_j$ dokumentu $d_i$. 
2.2 Mere kvaliteta i tehnike evaluacije klasifikatora

Definicija: Kategorizacija teksta je zadatak dodeljivanja Boolean vrednosti svakom paru \( \langle d_i, c_j \rangle \in D \times C \), gde je \( D \) domen dokumenata i \( C = \{ c_1, ..., c_{|C|} \} \) je skup predefinisanih kategorija [25].

Vrednost T (True) dodeljena paru \( \langle d_i, c_j \rangle \) ukazuje na odluku da je dokument \( d_i \) pod kategorijom \( c_j \), dok vrednost F (False) ukazuje na odluku da dokument \( d_i \) nije pod kategorijom \( c_j \). Zadatak je aproksimirati ciljnu funkciju \( f : D \times C \to \{ T, F \} \).

Različite zavisnosti se mogu javiti pri klasifikaciji teksta u zavisnosti od primene. Na primer za dato celobojno \( k \), tačno \( k \) elemenata iz \( C \) treba dodeliti svakom \( d_i \in D \). Slučaj u kome tačno jedna kategorija mora biti dodeljena svakom dokumentu \( d_i \in D \) se često naziva jednoznačna (eng. single-label) klasifikacija, dok se slučaj u kome se svakom dokumentu \( d_i \in D \) dodeljuje više od jedne kategorije naziva višeznačna (eng. multi-label) klasifikacija. Specijalan slučaj jednoznačne klasifikacije je binarna klasifikacija u kojoj se svakom dokumentu \( d_i \in D \) dodeljuje i kategorija \( c_j \) ili njen komplement \( \overline{c_j} \).

Iz teoretske tačke gledišta binarni slučaj (jednoznačna klasifikacija takođe) je opštiji od višeznačne klasifikacije jer se algoritam binarne klasifikacije može upotretiti i za višeznačnu klasifikaciju. Nepohodno je samo transformisati problem višeznačne klasifikacije tako da se kategorije \( \{ c_1, c_2, ..., c_n \} \) podele u \( |C| \) nezavisnih problema binarne klasifikacije kao \( \{ c_j, \overline{c_j} \} \), za \( i = 1, ..., |C| \) [25]. Međutim potrebno je da su kategorije stohastički nezavisne jedna od druge.

2.2 Mere kvaliteta i tehnike evaluacije klasifikatora

Ocena kvaliteta klasifikatora teksta se sprovodi eksperimentalno, a ne analitički, jer analitička procena zahteva poznavanje formalne specifikacije problema koji sistem pokušava da reši. Da bi što preciznije mogli da uporedimo različite algoritme klasifikacije, najbitnije je koristiti iste skupove podataka sa istom podelom dokumenata za učenje i testiranje. Ukoliko se klasifikator dobro ponaša na trening skupu podataka, a loše na test skupu podataka u pitanju je problem preprilagodeniog modela (eng. overfitting problem). Taj problem nastaje jer se formira model koji savršeno opisuje trening podatke, uključujući i specifičnosti nebitne za cilj učenja.

Postupak evaluacije klasifikatora predstavlja poređenje unapred poznate klase sa onom koju je predložio klasifikator. Na taj način se dobijaju ispravno i neispravno klasifikovani podaci. Na osnovu tih informacija formira se matrica konfuzije.

Tabela [4] prikazuje matricu konfuzije za problem binarne klasifikacije. Svaki unos \( f_{ij} \) u ovoj tabeli označava broj zapisa iz klase \( i \) koji su predviđeni u klasi \( j \). Na primer, \( f_{01} \)
je broj zapisa iz klase 0 pogrešno predviđenih kao klasa 1. Na osnovu unosa iz matrice konfuzije ukupan broj predviđanja koje je napravio model je \((f_{11} + f_{00})\) i ukupan broj neispravnih predviđanja je \((f_{10} + f_{01})\).

\[
\begin{array}{c|cc|c}
\text{Stvarna klasa} & \text{Klasa } = 1 & \text{Klasa } = 0 & \text{Ukupno} \\
\hline
\text{Klasa } = 1 & f_{11} & f_{10} & f_{11} + f_{10} \\
\text{Klasa } = 0 & f_{01} & f_{00} & f_{01} + f_{00} \\
\text{Ukupno} & f_{11} + f_{01} & f_{10} + f_{00} & N
\end{array}
\]

**Tabela 4:** Matrica konfuzije za problem 2 klase (binarne klasifikacije)

Vrednosti matrice konfuzije obrazložene na primeru klasifikacije elektronske pošte:

1. \(f_{11}\) – vrednost „stvarno pozitivni“ (eng. true positive) predstavlja broj dokumenata koji su zaista spamovi a koje je klasifikator prepoznao kao spamove
2. \(f_{01}\) – vrednost „lažno pozitivni“ (eng. false positive) predstavlja broj dokumenata koji nisu spamovi a koje je klasifikator klasifikovao u grupu spamova
3. \(f_{00}\) – vrednost „stvarno negativni“ (eng. true negative) predstavlja broj dokumenata koji nisu spamovi a klasiﬁkator ih je klasifikovao kao legitimne poruke
4. \(f_{10}\) – vrednost „lažno negativni“ (eng. false negative) predstavlja broj dokumenata koji su spamovi a koji su svrstani u grupu legitimnih poruka

Iako matrica konfuzije pruža informacije neophodne za određivanje koliko dobro klasifikacioni model funkcioniše, svođenje ovih informacija na jedan konkretan broj omogućava mnogo jednostavnije poređenje performansi različitih modela.

**Tačnost** (eng. accuracy) klasifikatora je mera koja nam daje procenat uspešno klasifikovanih dokumenata. Tačnost definišemo na sledeći način:

\[
\text{Tačnost} = \frac{\text{Broj ispravnih predviđanja}}{\text{Ukupan broj predviđanja}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}
\]

Za mnoge primere klasifikacije tačnost je korisna mera, međutim postoje slučajevi kada nam tačnost ne može dati informacije koje tražimo. To su uglavnom scenariji u kojima je jedna klasa značajno manja od druge. Tada je moguće dobiti visoku tačnost svrstavnjem svih instanci u veću grupu.
Ekvivalentno performanse modela se mogu izraziti u obliku stope greške, koja je data na sledeći način:

\[ \text{Stopa greške} = \frac{\text{Broj pogrešnih predviđanja}}{\text{Ukupan broj predviđanja}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}} = \frac{FN + FP}{TP + FN + FP + TN} \]

Standardne mere kvaliteta klasifikatora su preciznost i odziv. To su numeričke vrednosti koje predstavljaju mere očekivanja korisnika za klasifikaciju da će klasifikator ispravno klasifikovati slučajno odabran dokument.

**Preciznost** (eng. precision) je mera koja nam daje informaciju o udelu stvarno pozitivnih instanci. Od svih poruka koje su označene kao spam, koji procenat čine poruke koje su stvarno spam?

\[ \text{Preciznost} = \frac{f_{11}}{f_{11} + f_{01}} = \frac{TP}{TP + FP} \]

**Odziv** (eng. recall) je mera suprotna preciznosti. Od svih poruka koje su stvarno spam, koji procenat poruka je klasifikovan kao spam?

\[ \text{Odziv} = \frac{f_{11}}{f_{11} + f_{10}} = \frac{TP}{TP + FN} \]

Kombinovanjem preciznosti i odziva dobijena je \( f_1 \) mera koja se definiše na sledeći način:

\[ f_1 = \frac{2 \cdot \text{Preciznost} \cdot \text{Odziv}}{\text{Preciznost} + \text{Odziv}} \]
2.2.1 Unakrsna validacija

Tehnika za evaluaciju klasifikatora koja se često koristi je unakrsna validacija (eng. cross validation). Neophodno je da se skup podataka podeli na $k$ delova približno iste veličine, a zatim se svaki od $k - 1$ delova koristi kao skup za učenje a sam taj preostali deo kao skup za testiranje. Postupak se ponavlja $k$ puta tako da je svaki od delova po jednom učestvovao u ulozi testnog skupa podataka. Greška klasifikacionog modela je prosečna greška svih $k$ iteracija u postupku.

Često je slučaj da se postupak slučajnog izbora modifikuje tako da se osigura približno jednaka zastupljenost klasa u svakom od $k$ delova. Taj postupak se naziva stratifikacija i njime se obezbeđuje da je pri svakoj iteraciji zastupljenost klasa u skupu za učenje i testiranje približno jednaka zastupljenosti u inicijalnom skupu primera.

Složenost evaluacije klasifikacionog modela ovom metodom zavisi od $k$. U praksi se najčešće uzima $k = 5$ ili $k = 10$ jer se takva unakrsna validacija pokazala kao dovoljno tačna a nije previše zahtevna.
3 | Algoritmi filtriranja elektronske poštе

U ovom poglavlju predstaviću pregled osnovne teorije i ideje algoritama mašinskog učenja koji se koriste pri filtriranju elektronske poštе.

3.1 | Bajesovo filtriranje

Najpopularnija tehnika filtriranja elektronske poštе je Bajesovo filtriranje, koje je jednostavan metod koji se zasniva na verovatnoći. U kontekstu elektronske poštе to jednostavno znači da se verovatnoća da je poštа neželjena može izračunati na osnovu određenih reči ukoliko znamo verovatnoću neželjenih mejlova i koliko često se te reči pojavljuju u njima, podeljene sa verovatnoćom koliko često se reči pojavljuju u telima sve elektronske poštе [23]. Određene reči imaju određene verovatnoće pojavljivanja u neželjenoj i legitimnoj elektronskoj poštі. Većina korisnika kada vidi reč *Viagra* zna da se radi o neželjenoj poštі i retko će je videti u poštо koja nije neželjena. Filter ne zna ove verovatnoće unapred i neophodan mu je trening da bi mogao da odredi verovatnoću pojavljivanja određenih rečи u neželjenoj poštі. Za treniranje filtera neophodno je da korisnici elektronske poštе ručno obeležavaju da li je poštа neželjena ili ne. Za sve rečи svakog mejla pri testiranju filter će prilagoditi verovatnoće u svojoj bazi podataka. Na primer Bajesov filter neželjene poštе će sa veoma velikom verovatnoćom naučiti da se radi o neželjenoj poštі ukoliko se pojavljuje reč *Viagra* ali veoma malu verovatnoću da je reč o neželjenoj poštі za rečи koje su se pojavile u legitimnoj poštі. Nakon treniranja poznate su verovatnoće rečи koje se nalaze u bazi i moguće je izračunati verovatnoću da li je poštа neželjena ili ne. Ukoliko je izračunata verovatnoća svih rečи u mejlu veća od nekog praga (npr. 95%) filter će označiti ovaj mejl kao neželjen.

3.1.1 | Matematička osnova

Bajesov filter elektronske poštе koristi Bajesovu teoremу[3]

Bajesova teorema:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

gde je:

$P(A)$ — početna verovatnoćа događаja $A$

$P(B)$ — početna verovatnoćа pojavlјivanja instance $B$

$P(B|A)$ — uslovna verovatnoćа pojavlјivanja instance $B$ uz uslov ispravnosti događаja $A$

$P(A|B)$ — uslovna verovatnoćа ispravnosti događаja $A$ nakon pojavlјivanja instance $B$

koja je zanimljiva sa stanovиšта indukcije znanja jer omogućava procenu ispravnosti dogаđаja nakon posmatranja pojave novih instanci B.

---

Teorema sledi direktno iz definicije uslovne verovatnoće:

\[
P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{P(A \cap B)}{P(A)}}{P(B)} = \frac{P(B|A)P(A)}{P(B)}
\]

Kod konačno mnogo disjunktnih slučajeva \(A_i, i = 1, ..., n\) Bajesova teorema glasi:

\[
P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}
\]

Primer Bajesove teoreme:\[[12]\):

- Doktor zna da meningitis u 50% slučajeva prouzrokuje kočenje vrata.
- Prethodna (poznata) verovatnoća da bilo koji pacijent ima meningitis je \(\frac{1}{50000}\).
- Prethodna verovatnoća da bilo koji pacijent ima ukočen vrat je \(\frac{1}{20}\).

Ako pacijent ima ukočen vrat, koja je verovatnoća da ima i meningitis?

\[
P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.5 \cdot \frac{1}{50000}}{\frac{1}{20}} = 0.0002
\]

Pretpostavimo da poruka sadrži reč replika. Filter neželjene pošte će koristiti sledeću formulu koja se zasniva na Bajesovoj teoremi:

\[
P_r(S|W) = \frac{P_r(W|S) \cdot P_r(S)}{P_r(W|S) \cdot P_r(S) + P_r(W|H) \cdot P_r(H)}
\]

gde je:

- \(P_r(S|W)\) verovatnoća da je poruka neželjena ako znamo da se reč replika nalazi u njoj.
- \(P_r(S)\) ukupna verovatnoća da je bilo koja data poruka neželjena.
- \(P_r(W|S)\) verovatnoća da se reč replika nalazi u neželjenoj poruci.
- \(P_r(H)\) ukupna verovatnoća da je bilo koja data poruka legitimna.
- \(P_r(W|H)\) verovatnoća da se reč replika nalazi u legitimnoj poruci.

Nedavne statistike [26] pokazuju da je verovatnoća da je neka poruka neželjena 80%, pa je: \(P_r(S) = 0.8\) i \(P_r(H) = 0.2\). Međutim većina filtera neželjene pošte pravi pretpostavku da ne postoji razlog da dolazeća pošta ima veću verovatnoću da bude neželjena i pretpostavlja da je verovatnoća u oba slučaja jednaka 50%: \(P_r(S) = 0.5\) i \(P_r(H) = 0.5\). Za filtere koji koriste ovu hipotezu kaže sa da su nepristrasni jer nemaju predraste o
3.1 Bajesovo filtriranje

Ova pretpostavka dozvoljava nam pojednostavljanje polazne formule na sledeći način:

\[ P_r(S|W) = \frac{P_r(W|S)}{P_r(W|S) + P_r(W|H)} \]

Broj \( P_r(W|S) \) koji se koristi u ovoj formuli je aproksimiran frekvenciji poruka koje sadrže reč replika u porukama koje su identifikovane kao neželjene tokom faze učenja. Slično je \( P_r(W|H) \) je aproksimirano frekvenciji poruka koje sadrže reč replika u porukama koje su identifikovane kao legitimne tokom faze učenja. Da bi ove aproksimacije imale smisla neophodno je da skup poruka u fazi učenja bude dovoljno velik i reprezentativan. Takođe je preporučljivo da skup poruka u fazi učenja bude usaglašen sa 50% hipoteze o ponovnoj podeli poruka, odnosno da su skupovi podataka za neželjenu i legitimnu poštu iste veličine. Naravno, odlučivanje da li je poruka neželjena ili legitimna na osnovu samo jedne reči replika je podložno greškama, pa zbog toga Bajesov filter pokušava da napravi zaključak na osnovu više reči kombinovanjem više verovatnoća.

3.1.2 Kombinovanje individualnih verovatnoća

Većina Bajesovih algoritama za filtriranje elektronske pošte su zasnovani na formulama koje su striktno validne samo ako su reči prezentovane u poruci uslovno nezavisni događaji. Ovaj uslov nije generalno zadovoljen jer je na primer u prirodnim jezicima verovatnoća nalaženja pridava zavisnost od verovatnoće javljanja imenica u tekstu, ali je korisno kao idealizacija jer su statističke korelacije pojedinih reči obično nepoznate. Na osnovu toga može se izvesti sledeća formula iz Bajesove teoreme:

\[ p = \frac{p_1p_2\cdots p_N}{p_1p_2\cdots p_N + (1 - p_1)(1 - p_2)\cdots (1 - p_N)} \]

gde je:

\( p \) verovatnoća da je posmatrana poruka neželjena
\( p_1 \) je verovatnoća \( p(W_1|S) \) da je poruka neželjena znajući da sadrži prvu reč (npr. replika)
\( p_2 \) je verovatnoća \( p(W_2|S) \) da je poruka neželjena znajući da sadrži drugu reč (npr. satovi)
\[ \cdots \]
\( p_N \) je verovatnoća \( p(W_N|S) \) da je poruka neželjena znajući da sadrži N-tu reč (npr. kuća)

Spam filteri bazirani na ovoj formuli ponekad se nazivaju naivni Bajesovi klasifikatori. Rezultat \( p \) se tipično poredi sa pragom koji odlučuje da li je poruka neželjena ili legitimna. Ako je \( p \) niže od praga poruka se smatra legitimnom, inače se smatra neželjnom.

\[ ^4 \text{Vrednost } P_r(S|W) \text{ se u literaturi popularno naziva } spamicity \text{ ili } spaminess \text{ i predstavlja procenat pojavljivanja date reči u neželenoj pošti. Ona se računa za svaku reč koja se pojavljuje u poruci i kreće se u opsegu od 0 do 1.} \]
3.1 Bajesovo filtriranje

3.1.3 Rad sa retkim rečima


Primena Bajesove teoreme pod pretpostavkom klasifikacije između neželjene i legitimne pošte koja sadrži reč replika je slučajna promenljiva sa beta raspodelom. Neki programi koriste sledeću korigovanu verovatnoću:

\[ P'_r(S|W) = \frac{s \cdot P_r(S) + n \cdot P_r(S|W)}{s + n} \]

gde je:
- \( P'_r(S|W) \) – korigovana verovatnoća da je posmatrana poruka neželjena, znajući da sadrži datu reč
- \( s \) – vrednost koju prosleđujemo kao dodatnu informaciju o dolaznoj neželjenoj poruci
- \( P_r(S) \) – verovatnoća da je dolazna poruka neželjena
- \( n \) – broj pojavljivanja date reči tokom faze učenja
- \( P_r(S|W) \) – verovatnoća da je poruka neželjena ukoliko znamo da se data reč nalazi u njoj

\( P_r(S) \) se ponovo može uzeti da je jednako 0.5 da bi izbegli suvišnu sumljanicu oko dolazne elektronske pošte. Za promenljivu s dobra vrednost je 3 što znači da je u fazi učenja neophodno da se data reč mora sadržati u više od tri poruke kako bi imali više poverenja u \( P_r(S|W) \) vrednost nego u podrazumevanu vrednost. Ova formula se može proširiti na slučaj kada je \( n \) jednako nuli i u ovom slučaju se ocenjuje kao \( P_r(S) \).

3.1.4 Druge heuristike

Neutralne reči kao „the”, „a”, „some” ili „is” (na engleskom) ili njihovi ekvivalenti na drugim jezicima mogu da se ignorišu. Neki Bajesovi filtri ignorišu reči čiji je spamicity blizu 0.5 jer veoma slabo dobrište dobrim odlučivanju. Reči koje se uzimaju u razmatranje su one čiji je spamicity blizu 0.0 (karaktetistični znaci da je poruka legitimna) ili 1.0 (karakteristični znaci da je poruka neželjena).

Neki programi uzimaju u obzir činjenicu da li se data reč pojavljuje više puta u ispitivanoj poruci [8], a neki softverski proizvodi koriste šablone (nizove reči) umesto izolovanih reči prirodnih jezika [31]. Na primer ukoliko koristimo šablone za kontekst od četiri reči spamicity će se računati nad nizom od te četiri reči „vijagra je dobra za” umesto izračunavanja spamicity za „vijagra”, „je”, „dobra”, „za”. Samim tim ovaj metod daje više osjetljivosti na kontekst i bolje eliminise šum.

5Ukoliko podaci imaju sum data pripadnost u skupu obuke ne mora biti tačna.
3.1.5 Primena

Kada prihvatimo poruku \( m \) moramo definisati funkciju odlučivanja \( f \) koja dodeljuje poruku \( m \) svojoj klasi. Spam poruke obeležavamo sa \( S \), a legitimne sa \( L \). Ako je \( G_M \) skup poruka tada funkciju \( f \) definišemo na sledeći način:

\[ f : G_M \rightarrow \{ S, L \} \]

Kod ovakvih tehnika prvo moramo proveriti neke karakteristike koje mogu uticati na klasifikaciju poruke. Pozivaćemo se na te karakteristike koristećenjem vektora \( \vec{x} \). Neka je \( P(\vec{x}/c) \) verovatnoća da klasa \( c \) generiše poruku čiji je vektor karakteristika \( \vec{x} \). Ako pretpostavimo da legitimna poruka nikad nije sadržala tekst \( t = "Kupi sada" \) i da je \( x = (m = utv) \) gde su \( u \) i \( v \) dva stringa, tada je verovatnoća \( P(\vec{x}/L) = 0 \). Sada je problem izračunati verovatnoću da poruka koja sadrži karakteristični vektor \( \vec{x} \) pripada klasi \( c \) što zapisujemo kao \( P(c/\vec{x}) \). To možemo dobiti posmatranjem Bajesovog pravila:

\[ P(c/\vec{x}) = \frac{P(\vec{x}/c)P(c)}{P(\vec{x}/S)P(S) + P(\vec{x}/L)P(L)} \]

\( P(\vec{x}) \) predstavlja apriornu verovatnoću pojavljivanja poruke čiji je karakteristični vektor \( \vec{x} \), a \( P(c) \) predstavlja verovatnoću da bilo koja poruka pripada klasi \( c \). Znajući verovatnoće \( P(c) \) i \( P(\vec{x}/c) \) dovoljno je zaključiti \( P(c/\vec{x}) \). Tada imamo sledeće pravilo klasifikacije: Ukoliko je \( P(S/\vec{x}) > P(L/\vec{x}) \) (aposteriorna verovatnoća poruka koje imaju karakterističan vektor \( \vec{x} \) su veće od istih poruka koje pripadaju klasi \( L \)) tada se poruka \( m \) klasifikuje kao neželjena. Da bi smo mogli da klasifikujemo poruku neophodno je da odredimo verovatnoće \( P(c) \) i \( P(\vec{x}/c) \) za bilo koju poruku \( m \) ali to očigledno ne može da se odredi precizno. Međutim možemo da aproksimiramo ove verovatnoće trening podacima. Na primer verovatnoća \( P(S) \) se može grubo odrediti izračunavanjem odnosa broja neželjenih poruka i broja svih poruka u trening podacima.

Zbog jednostavnosti možemo smatrati da je karakterističan vektor binaran gde se prisustvo reči \( w \) u poruci \( m \) reprezentuje jednom jedinicom. Tada možemo reći da je:

\[ P(x_\omega = 1/S) \approx \frac{\text{Broj neželjenih poruka u kojima je } w \text{ obeleženo}}{\text{Broj svih neželjenih poruka}} \]

U suštini mi predstavljamo prisustnost reči \( \omega_i \) u poruci \( m \) vrednosti 1 u karakterističnom vektoru \( \vec{x} = (x_1, x_2, ..., x_n) \). Međutim algoritam \([1]\) mora da računa \( 2^n \) vrednosti \( x \) što je nepraktično. Kako bismo to izbegli uvodimo pretpostavku da su dve prisutne reči uslovno nezavisne jedna od druge što znači da je:

\[ P(\vec{x}/c) = \prod_{i=1}^{n} P(\vec{x}_i/c) \quad \Lambda = \prod_{i=1}^{n} \Lambda_i(\vec{x}_i) \]
Algoritam 1 Trening algoritam za Bajesov klasifikator

1: for all \( c \in \{S, L\} \) do
2: \( \text{Izračunaj } p(c) \)
3: for all \( x_\omega \in \{0, 1\} \) do
4: \( \text{Izračunaj } p(x_\omega / c) \)
5: end for
6: end for
7: for all \( \vec{x}_\omega \in \{0, 1\} \) do
8: \( \text{Izračunaj } p(\vec{x}_\omega / c) \) \( \triangleright \) koristeći Bajesovo pravilo
9: end for
10: for all \( x_\omega \in \{0, 1\} \) do
11: \( \text{Izračunaj } \Lambda(\vec{x}_\omega) \)
12: end for
13: \( \text{Izračunaj } \lambda \frac{P(L)}{P(S)} \)

Algoritam 2 Klasifikacija zasnovana na Bajesovom klasifikatoru

1: \( \text{Izračunaj vektor } \vec{x}_\omega \text{ ulazne poruke m} \)
2: if \( \Lambda(\vec{x}_\omega) > \lambda \frac{P(L)}{P(S)} \) then
3: \( \text{poruka m je neželjena} \)
4: else
5: \( \text{poruka m je legitimna} \)
6: end if

Pri čemu \( \Lambda(\vec{x}) \) predstavlja količnik \( \frac{P(\vec{x}/S)}{P(\vec{x}/L)} \), dok je \( \lambda \) parametar koji ukazuje na rizik kada klasifikujemo legitimnu poruku kao neželjenu.

\[
\lambda = \frac{\mathcal{L}(L, S)}{\mathcal{L}(S, L)}
\]

Funkcija \( \mathcal{L}(c_1, c_2) \) određuje cenu loše klasifikacije pojavom klase \( c_1 \) umesto klase \( c_2 \). Logično je reći da je \( \mathcal{L}(L, L) = \mathcal{L}(S, S) = 0 \). Možemo definisati funkciju rizika kao:

\[
R(c/\vec{x}) = \mathcal{L}(S, c)P(S/\vec{x}) + \mathcal{L}(L, c)P(L, \vec{x})
\]

Klasifikovanje legitimne poruke kao neželjene je generalno mnogo veća greška od dopuštanja da neželjena poruka prođe kroz filter. Ukoliko imamo dva tipa grešaka \( L \rightarrow S \) i \( S \rightarrow L \) pretpostavljamo da je u slučaju greške \( L \rightarrow S \) parametar \( \lambda \) nekoliko puta veći nego u slučaju greške \( S \rightarrow L \). Poruka se klasifikuje kao neželjena ukoliko je sledeći kriterijum ispunjen:

\[
\frac{P(\vec{x}|c = S)}{P(\vec{x}|c = L)} > \lambda
\]

Ukoliko važi pretpostavka nezavisnosti procenjene verovatnoće su precizne i klasifikator na osnovu ovih kriterijuma postiže optimalne rezultate [5].
3.2 K najbližih suseda

Ovaj metod se naziva još i metod zasnovan na memoriji ili instancama. Pripada porodići algoritama učenja koji umesto izvođenja eksplcitne generalizacije upoređuje nove instance sa instancama koje se nakon trening faze nalaze u memoriji. Učenje zasnovano na instancama je vrsta lenjog učenja. Zajednička karakteristika ovih metoda je da svi oni smeštaju trening instance u memorijanske strukture i koriste ih direktno za klasifikaciju. Naj jednostavniji oblik memorijske strukture je višedimenzionalni prostor definisan atributima u inicijalni vektor. Svaka trening instanca je reprezentovana kao tačka u prostoru. Procedura klasifikacije je najčešće jednostavna varijanta algoritma k najbližih suseda.

Klasifikacija novih instanci se obavlja prema principu najbližeg suseda, gde se nova instanca upoređuje sa instancama iz skupa za učenje korišćenjem definisane metrike. Metrika definiše rastojanje instanci na osnovu vrednosti njihovih atributa, a odgovara intuitivnom shvatanju sličnosti instanci tako da ako su instance sličnije rastojanje je manje. Nova instanca se klasifikuje na osnovu pretraživanja skupa za učenje sa ciljem pronalaženja instance koja mu je u smislu rastojanja najbliža. Nova instanca koja se klasifikuje dobija klasu te instance.

Ideja algoritma k najbližih suseda:

1. **Trening** - smeštanje trening poruka
2. **Klasifikacija** - Konkretna poruka \( x \) sadrži k najbližih suseda među porukama u trening skupu. Ukoliko ima više spam poruka među ovim susedima, poruka će biti klasifikovana kao spam, u suprotnom klasifikator će je klasifikovati kao legitimnu poruku.

Kao što se može primetiti, trening faza ne postoji u običajnom smislu. Cena ovoga je sporija procedura odlučivanja jer u cilju klasifikovanja jedne poruke moramo računati rastojanja do svih trening poruka i pronaći k najbližih suseda. Ovo pri najtrivijalnijoj implementaciji može trajati \( O(nm) \) vremena za trening skup od \( n \) poruka koje sadrže vektore obeležija sa \( m \) elemenata. Izvršavanje pametnog indeksiranja u trening fazi će omogućiti smanjenje kompleksnosti klasifikacije poruka na \( O(n) \). Drugi problem koji se javlja kod ovog algoritma je nepostojanje parametara čije će podešavanje smanjiti broj lažno pozitivnih instanci. Taj problem se jednostavno rešava modificacijom klasifikacionog pravila na sledeće \( l/k \) pravilo:

Ukoliko je \( l \) ili više poruka među \( k \) najbližih suseda poruke \( x \) spam klasifikuj \( x \) kao spam, u suprotnom klasifikuj \( x \) kao legitimnu poruku.

---

\(^6\)Lenjog učenje u veštačkoj inteligenciji je poželjna osobina učenja ako postoji zahtev za stalnim menjanjem baze znanja, gde svaka takva promena ne povlači ponavljanje celog postupka učenja već samo efikasno inkrementalno dodavanje znanja.
Pseudo kod algoritma $k$ najbližih suseda:

**Algoritam 3 $K$ najbližih suseda**

1: Ulaz:
   $k$ - broj suseda  
   $z$ - nova instanca opisana sa $m$ obeležija  
   $TS$ - trening skup od $n$ instanci opisan sa $m$ obeležija  
   $Kat$ - klase instanci iz trening skupa  

2: Izlaz:
   $c$ - predviđena kategorija nove instance

3: procedure $kNajблиzihSuseda(k, z, TS, Kat)$

4: for all $x \in \{TS\}$ do  \(\triangleright \) računanje rastojanja od $z$ do svih ostalih instanci $x$ iz trening skupa  
5: \(d(x) = \text{rastojanje}(x, z);\)  
6: end for  
7: $[\text{sortirane instance, index}] = \text{sort}(d);$  \(\triangleright \) sortiranje distanci u rastućem poretku  
8: $kategorija\_suseda = \text{Kat(index}(1:k));$  
9: $c = \text{vecina\_Glasova(kategorija\_suseda);}$  
10: return $c$  
11: end procedure

**Slika 3:** Primer koji ilustruje algoritam $k$ najbližih suseda za $k = 1, k = 2, k = 3$

Mana klasifikatora zasnovanih na memoriji je cena izračunavanja u fazi klasifikacije zbog njihove „lenje” prirode jer svi trening primeri moraju da sprovedu klasifikaciju što je sa velikim skupovima podataka posebno teško.
3.2 K najbližih suseda

3.2.1 Euklidska metrika

Element tehnike klasifikacije zasnovane na instancama koji utiče na oblik modela je metrika, pri čemu je u upotrebi više različitih metrika, a najčešće se koristi euklidska. Euklidsko rastojanje ispunjava sledeće dobro poznate osobine metrike:

1. Pozitivna određenost
   \( d(x, y) \geq 0 \) za svako \( x \) i \( y \)
   \( d(x, y) = 0 \) samo ako je \( x = y \)

2. Simetrija
   \( d(x, y) = d(y, x) \) za svako \( x \) i \( y \)

3. Nejednakost trougla
   \( d(x, z) \leq d(x, y) + d(y, z) \) za svako \( x, y \) i \( z \)

Euklidsko rastojanje instanci \( x \) i \( y \) se može predstaviti na sledeći način:

\[
   d(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\]

U datim izrazima se implicitno pretpostavlja da su svi atributi numeričkog tipa, tj. da su vrednosti kordinata brojevi, a kako bi se definicija euklidskog rastojanja mogla primeniti na nominalne atribute potrebno je definisati operaciju razlike nad nominalnim vrednostima.

Ako su sa \( a_i, a_j \in Dom(A_i) \) označene dve proizvoljne vrednosti nominalnog atributa \( A_i \), onda je razlika vrednosti \( a_i \) i \( a_j \) definisana \( 0 - 1 \) funkcijom razlikovanja na sledeći način:

\[
   a_i - a_j = \begin{cases} 
   0 & \text{za } a_i = a_j \\
   1 & \text{inače}
   \end{cases}
\]

Zbog različitih skala merenja za različite atribute postoji problem vezan za korišćenje različitih metrika. Tako na primer, atribut čiji se raspon vrednosti kreće unutar desetih delova merne jedinice ima zanemariv uticaj na konačni rezultat u odnosu na atribut sa rasponom vrednosti od nekoliko desetina mernih jedinica. Zato je kod metoda klasifikacije zasnovane na instancama uobičajan postupak normalizacije svih numeričkih atributa na intervalu \([0, 1]\), korišćenjem funkcije:

\[
   f(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min}}
\]

gde \( x_{\min} \) i \( x_{\max} \) označavaju najmanju, odnosno najveću vrednost posmatranog atributa. Kod metoda klasifikacije zasnovanim na instancama, neodređene vrednosti atributa se
tretiraju slično nominalnim atributima, tj. proširuje se definicija operacije razlike, tako
da se razlika definiše na način da je neodređena vrednost maksimalno udaljena od bilo
koje posmatrane vrednosti atributa.

3.2.2 Pretraga prostora rešenja

Neke varijante klasifikacije zasnovane na instancama nastoje da redukuju broj instanci
u skupu za učenje, prvenstveno radi smanjenja opsega pretraživanja pri klasifikaciji novih
instanci, jer skup instanci za učenje po pravilu sadrži veliki broj redundantnih instanci.
Kod problema klasifikacije najvažnije su instance koje se nalaze u blizini granica među
klasama, a instance iz unutrašnjosti određenog područja klase mogu se izostaviti bez
posledica na tačnost klasifikacije. Postupak formiranja podskaupa instanci je iterativan, a
sastoji se od uvrštavanja ili eliminisanja instanci prema unapred definisanom kriterijumu.
Zbog toga je u takvom modelu neophodno zadržati reprezentativne instance koje dobro
ograničavaju područje u kojem se nalaze, a iz skupa izbaciti instance koje bitno ne do-
prinose oblikovanju područja odgovarajuće klase. Kriterijumi prihvatanja i eliminisanja
instanci uglavnom pretstavljaju nepovratne strategije pohlepnog karaktera. Najjedno-
stavniji kriterijum je da se ispituje instance prema rezultatu klasifikacije korišćenjem
toda izdvojenih reprezentativnih instanci. Ako je u pitanju netačna klasifikacija instance
ona se pridodaje u skup reprezentativnih instanci jer je evidentno da menjaj granice klasa.
Ukoliko je klasifikacija instance tačna tada se ona proglašava suvišnom jer informacija
koju nosi je već sadržana u skupu pomoću kojeg je klasifikovana[11].

Nedostaci datog kriterijuma za izbor instanci[11]:

- U početnoj fazi procesa pretraživanja postoji nezanemariva verovatnoća odbacivanja
instanci koje se mogu pokazati važnim za tačnost klasifikacije rezultujućeg modela.
- Pored ovoga, izabrani podskaup reprezentativnih instanci ne zavisi samo od polaznog
skupa, već i o redosledu evaluacije instanci.
- Bitan nedostatak se odnosi na loše ponašanje u uslovima šuma u podacima, jer s
obzirom da se u skup uvrštavaju i netačno klasifikovane instance, ovaj kriterijum ima
tendenciju akumuliranja instanci sa šumom u rezultirajućem skupu instanci, što dovodi
do smanjenja njegove reprezentativnosti.

3.2.3 Udaljenost instanci

Pored izbora instanci za pamćenje, na oblik klasifikacionog modela se može uticati
i modifikacijom funkcije rastojanja. Jednak uticaj svih atributa u instanci na konačan
rezultat je jedno od svojstava euklidskog rastojanja, ali su u praksi retki problemi kod ko-
jih svi atributi imaju jednaku vrednost za proces klasifikacije, čime se stvara mogućnost za
poboljšanje tehničke klasifikacije zasnovane na instancama. Modifikacija euklidskog rasto-
janja podrazumeva uvođenje težinskih vrednosti atributa. Ako sa $w_i$ označimo težinski
vrednost pridruženu atributu $A_i$, onda modifikovano Euklidsko rastojanje instanci $x$ i $y$
možemo predstaviti na sledeći način:

\[ d_w(x, y) = \sqrt{\sum_{i=1}^{n} w_i^2(x_i - y_i)^2} \]


Svim atributima je inicijalno pridružena težinska vrednost 1, koja se iterativno modificira pri razmatranju svake od instanci iz skupa za učenje. U podskupu relevantnih instanci se pronalazi instanca y najbliža posmatranoj instanci x, kao i pri klasifikaciji instanci. Ako instance x i y pripadaju istoj klasi, smanjuje se težinska vrednost atributa čije se vrednosti u instancama x i y najviše razlikuju, jer se razlika u vrednostima tih atributa pripisuje slabijoj korelaciji sa klasom, kao i što se u slučaju da instance x i y pripadaju različitim klasama, težinska vrednost atributa sa najvećom razlikom vrednosti povećava. Povećanje, odnosno smanjenje težinske vrednosti proporcionalno je razlici vrednosti atributa u instancama x i y.

3.2.4 | Šum u podacima za učenje

Osnovni oblik tehnike k najbližih suseda je prilično podložan problemu šuma u podacima za učenje, a razlog tome je da se klasifikacija nove instance oslanja na samo jednu (najbližu) instancu iz skupa za učenje. Značajno smanjenje uticaja šuma može se sprovesti proširenjem postupka klasifikacije prema principu k najbližih suseda, gde se umesto izdvajanja samo jedne najbližje instance iz skupa za učenje, izdvaja k najbližih instanci, za neki unapred određen broj k. U klasifikaciji nove instance učestvuje k pronađenih instanci, po principu većinskog glasanja, pri čemu se instanci pridružuje najfrekventnija klasa unutar izdvojenih k instanci. Specijalni slučaj ovog uopštenja za k = 1 predstavlja osnovni algoritam klasifikacije. Neophodno je da k ne bude suviše malo jer kNN klasifikator će biti podložan preprilagođavanju (overfitting[7]) zbog šuma u trening podacima.

Vrednost konstante k zavisi od količine šuma u podacima za učenje, pri čemu ako je više šuma, povoljnije je izabrati veće vrednosti konstante k. Na ovaj način se postiže poboljšanje tačnosti klasifikacije u uslovima šuma, zbog čega je varijanta k najbližih suseda gotovo u potpunosti istisnula osnovni oblik algoritma. Za posmatrani skup instanci, može se dokazati da za \(|S| \to \infty\) i \(k \to \infty\) na način da \(\frac{k}{|S|} \to 0\) verovatnoća pogrešne klasifikacije teži teoretskom minimumu[11].

\[7\]Do preprilagođavanja dolazi kada je model takav da je greška pri treniranju mala, a greška pri testiranju značajno veća. Postoje razni razlozi zašto dolazi do preprilagođavanja modela kao što su prisustvo šuma ili nepostojanje reprezentativnih instanci.
3.3 Metod podržavajućih vektora (SVM)

Metod podržavajućih vektora predstavlja binarni klasifikator koji konstruiše hiper-ravan u i na taj način stvara model koji previdи da joj od dve klase pripada nova instanca. Ovaj metod je razvijen 1995. godine i veoma je popularan zbog dobrih rezultata koji se dobijaju. Ideja metode je da se u vektorskom prostoru u kome su podaci predstavljeni nadе razdvajajuća hiper-ravan tako da su svi podaci iz iste klase sa iste strane ravni, kao što je prikazano na slici 4.

\[ y \cdot w \cdot x + b = 0 \]

\[ y \cdot w \cdot x + b = 1 \]

\[ y \cdot w \cdot x + b = -1 \]

Slika 4: Primer koji ilustruje optimalnu hiper-ravan sa maksimalnom marginom koja razdvaja podatke za trening u 2 klase

Ako znamo da su podaci linearno razdvojivi, u fazi treniranja je neophodno pronaći optimalnu razdvajajuću hiper-ravan, a to je ravan sa maksimalnom marginom. Margina predstavlja širinu razdvajanja između klasa koju treba maksimizovati i iznosi \[ \rho = \frac{2}{||w||} \]. Kada pronađemo najbolju hiper-ravan i njenu jednačinu, imamo model na osnovu kog određujemo klasu nove instance. Na slici 4 je prikazan linearni klasifikator kod koga prava \( w \cdot x + b = 0 \) predstavlja granicu odlučivanja. SVM pronalazi optimalno rešenje koje maksimizuje razdaljinu između hiper-ravni i tačaka koje su blizu razdvajajuće hiper-ravni i predstavlja intuitivno rešenje: ako nema tačaka blizu linije razdvajanja, onda će klasifikacija biti relativno laka.

3.3.1 Linearno razdvojive klase

Definicija: Neka \( X = (x_i, y_i), x_i \in \mathbb{R}^D, y_i \in \{-1, +1\} \) označava skup primera za testiranje u \( D \)-dimenzionalnom prostoru, gde svaki uzorak ima \( D \) atributa.
3.3 Metod podržavajućih vektora (SVM)

Ukoliko su podaci linearno razdvojivi možemo da nademo pravu za dvodimenzionalni slučaj, odnosno hiper-ravan za višedimenzionalni slučaj. Tada izrazom $w \cdot x + b = 0$ možemo opisati hiper-ravan pri čemu je $w$ normala hiper-ravni i $||b||$ vertikalna udaljenost hiper-ravni od kordinatnog početka. Instance najbliže razdvajajućoj hiper-ravni su podržavajući vektori. Cilj je izabrati hiper-ravan maksimalno udaljenu od najbližih uzoraka obe klase, kao što je prikazano na slici 4. Neophodan uslov za pronalaženje razdvajajuće hiper-ravni svodi se na izbor parametara $w$ i $b$, takvih da ulazne podatke možemo opisati sledećim izrazima:

$$x_i \cdot w + b \geq +1 \text{ za } y_i = +1$$
$$x_i \cdot w + b \leq -1 \text{ za } y_i = -1$$

Kombinovanjem dva prethodna izraza dobijamo:

$$y_i(x_i \cdot w + b) - 1 \geq 0, \forall i$$

Da bi izabrali hiper-ravan maksimalno udaljenu od potpornih vektora, potrebno je maksimizovati marginu, što je ekvivalentno pronalaženju:

$$\min ||w|| \text{ takav da } y_i(x_i \cdot w + b) - 1 \geq 0, \forall i$$

3.3.2 Linearno nerazdvojive klase

U slučaju linearno nerazdvajajućih problema, koristimo nelinearni SVM, pri čemu je osnovna ideja da se osnovni (ulazni) vektorski prostor preslika u neki višedimenzioni prostor u kome je skup podataka za trening linearno razdvojiv[13]. Na slici 5 prikazano je preslikavanje u višedimenzioni prostor u kome je skup podataka za trening linearno razdvojiv.

Slika 5: Preslikavanje u višedimenzionalni prostor u kome je skup podataka linearno razdvojiv
3.3 Metod podržavajućih vektora (SVM)

Uvođenjem nenegativne vrednosti $\xi_i$ ublažavamo uslove linearnog SVM-a i dobijamo sledeće izraze:

\[
\begin{align*}
  x_i \cdot w + b &\geq +1 - \xi_i \text{ za } y_i = +1 \\
  x_i \cdot w + b &\leq -1 + \xi_i \text{ za } y_i = -1
\end{align*}
\]

Kombinovanjem dva prethodna izraza dobijamo:

\[
y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0, \xi_i \geq 0, \forall i
\]

Primjenjena metoda naziva se *metoda meke margine* (eng. *soft margin method*), a izvorno je nastala sa idejom dozvoljavaanja pogrešnog označavanja klase pre samog postupka učenja. Slika 6 prikazuje hiper-ravan kroz dve linearno nerazdvojive klase, gde je vidljiv i uzorak sa pogrešne strane hiperravni zbog kojeg prostor nije linearno razdvojiv. Mera rastojanja tog uzorka od pripadajućeg potpornog vektora je $\xi$.

![Slika 6: Primer koji ilustruje dve linearno nerazdvojive klase](image)

Izbor razdvajajuće hiper-ravni svodi se na pronalaženje:

\[
\min \frac{1}{2}||w||^2 + C \sum_{i=1}^{L} \xi_i \text{ takav da } y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0, \xi_i \geq 0, \forall i
\]

gde vrednost $C$ predstavlja faktor greške, kojim dozvoljavamo određene greške pri treniranju, bez čega pronalažak hiper-ravni ne bi bio moguć.

Složenost treniranja SVM algoritma zavisi od kernela kao i od konačnog broja podržavajućih vektora koji se kriju u podacima za rešavanje dualnog problema. Kako broj podržavajućih vektora asimptotski raste linearno sa brojem primera, sledi da je u najboljem slučaju složenost $O(n^2)$ kada je $C$ malo i $O(n^3)$ kada je $C$ veliko. Kada je $C$ preveliko optimizacioni algoritam će pokušati da smanji $||w||$ koliko god je moguće, tako
da hiper-ravan pokušava pravilno da klasifikuje svaku instancu, ali to dovodi do gubitka svojstva generalizacije klasifikatora. S druge strane ukoliko je $C$ premalo, optimizacioni algoritam će pokušati da poveća $||w||$ previše, što dovodi do velike trening greške.

### 3.3.3 Upotreba kernela

Kernel je funkcija koja odgovara skalarnom proizvodu u nekom proširenom prostoru (prostoru veće dimenzije). Računanje skalarnog proizvoda između vektora:

$$K(x_i, x_j) = x_i^T \cdot x_j$$

Preslikavanjem svake tačke u prostor veće dimenzije koristeći transformaciju $\Phi : x \rightarrow \phi(x)$ skalarni proizvod postaje

$$K(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$$

pomoću koje se izračunavanje vrši na mnogo jednostavniji način.

**Primer preslikavanja $\Phi$:**

Ulazni (originalni) dvodimezionalni prostor: $x = (x_1, x_2)$. Neka je $K(x_i, x_j) = (1 + x_i^T x_j)^2$.

Pokažimo da vredi $K(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$, ako je $\phi(x) = [1, x_1^2, \sqrt{2} x_1 x_2, x_2^2, \sqrt{2} x_1 \sqrt{2} x_2]$.

$$K(x_i, x_j) = (1 + x_i^T x_j)^2 = 1 + x_1^2 x_1^2 + 2 x_1 x_1 x_2 x_2 + x_2^2 x_2^2 = \phi(x_i)^T \cdot \phi(x_j)$$

Dakle kerndi omogućavaju pretvaranje nerazdvojivih problema u razdvojive i vrše bolje preslikavanje u prostore viših dimenzija. Matematička teorija definiše uslove koje data funkcija treba da zadovolji da bi predstavljala skalarni proizvod u nekom vektor-skornom prostoru: simetričnost, pozitivna definisanost, zatvorenost za linearne kombinacije, množenje skalaram, itd. Funkcija koja zadovolji te uslove može da se koristi kao kernel.

Primeri uobičajenih kernela:

1. **Linearni**  - $K(x_i, x_j) = x_i^T \cdot x_j$
2. **Polinomijalni**  - $K(x_i, x_j) = (1 + x_i^T x_j)^d$
3. **RBF**  - $K(x_i, x_j) = e^{-|x_i - x_j|^2 / 2\sigma^2}$
4. **Sigmoid**  - $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Najjednostavniji je linearini kernel i odlično se ponaša za linearno odvojive podatke. Za potrebe evaluacije upotrebljen je upravo linearini kernel, a u okviru SVM\textit{light} biblioteke koja je korišćena pri implementaciji SVM algoritma, moguće je koristiti više različitih tipova kernela.

```c
kernelParm.kernel_type = 0;
/* 0=linear, 1=poly, 2=rbf, 3=sigmoid, 4=custom, 5=matrix */```
3.4 Veštačke neuronske mreže

Neuronske mreže nastale su iz težnje za razvijanjem matematičkih struktura koje bi bile u mogućnosti da simuliraju rad ljudskog mozga, kao i da koriste te strukture u rešavanju praktičnih problema. Kako bi razumeli osnovnu strukturu veštačkih neuronskih mreža potrebno je razmotriti osnovnu strukturu ljudskog mozga. Ljudski mozak, čija je struktura složena, a mreža neurona gusta, sastoji se od oko $10^{11}$ neurona koji su međusobno povezani u slojeve, koji čine složenu mrežu. Biološka ćelija koja obrađuje informacije je neuron. Zbog složene strukture neurona još uvek nije došlo do detaljnijih saznanja o funkcionisanju ljudskog mozga.

Sastoje se iz velike klase algoritama koji imaju svoju primenu u klasifikaciji, regresiji i proceni gustine. U suštini, neuronske mreže su određene funkcijama složene reprezentacije koje mogu biti dekomponovane na manje delove (neuroni, procesne jedinice) i reprezentovane grafički kao mreža neurona. Veoma mnogo funkcija se može reprezentovati na ovaj način i zbog toga nije uvek jasno koji algoritam pripada kom polju neuronskih mreža. Međutim možemo reći da je perceptron osnovna jedinica neuronske mreže, dok je višeslojni perceptron jedan od najkorišćenijih algoritama nadgledanog učenja u modelima neuronskih mreža.

3.4.1 Perceptron

Ideja perceptrona je pronaći linearnu funkciju vektora obeležja $f(x) = w^T x + b$ tako da je $f(x) > 0$ za vektore jedne klase i $f(x) < 0$ za vektore druge klase. Neka je $w = (w_1, w_2, ..., w_m)$ vektor koeficijenata (težina) funkcije i $b$ takozvani slobodni koeficijent (bias). Ako označimo klase brojevima $+1$ i $-1$ možemo reći da tražimo funkciju odluke $d(x) = \text{sign}(w^T x + b)$. Funkcija odluke se može reprezentovati grafički kao neuron i zbog toga se perceptron smatra neuronskom mrežom. To je naravno najtrivijalnija mreža sa jednom procesnom jedinicom.

![Slika 7: Perceptron kao neuron](image)

*Slobodni koeficijent ili bias (sklonost) je vrednost koja nam omogućuje transliranje funkcije odluke levo ili desno, što može biti kritično za uspešno učenje.*
3.4 Veštacne neuronske mreže

3. ALGORITMI FILTRIRANJA ELEKTRONSKE POŠTE

Ako vektori koji trebaju da budu klasifikovani imaju samo dve koordinate (važi da $x \in \mathbb{R}^2$) mogu se reprezentovati kao tačke na ravni. Funkcija odluke se može reprezentovati kao linija koja razdvaja ravan na dva dela. Vektori koji pripadaju jednoj polu-ravni biće klasifikovani kao pripadnici jedne klase dok će vektori koji pripadaju drugoj polu-ravni pripadati drugoj klasi. Ukoliko vektor ima tri koordinate, granica odlučivanja će biti u trodimenzionalnom prostoru, uopšteno ukoliko je vektor obeležja $n$-dimenzionalan granica odlučivanja je $n$-dimenzionalna hiper-ravan. Ove činjenice ukazuju na to da je perceptron linearni klasifikator.

Učenje perceptrona je urađeno iterativnim algoritmom. On počinje proizvoljno izabranim parametrima $(w_0, b_0)$ za funkciju odluke i ažurira ih iterativno. Na $n$-toj iteraciji algoritma, trening primer $(x, c)$ je odabran tako da ga trenutna funkcija odluke ne klasifikuje tačno (važi $\text{sign}(w^T x + b) \neq c$). Parametri $(w_n, b_n)$ se ažuriraju korišćenjem pravila:

$$w_{n+1} = w_n + cx$$
$$b_{n+1} = b_n + c$$

Algoritam se zaustavlja kada funkcija odluke tačno klasifikuje sve trening primere. Ukoliko takva funkcija ne postoji (klase nisu linearno razdvojive) algoritam učenja nikada ne konvergira i perceptron se ne može primeniti u ovom slučaju. Ukoliko podaci nisu linearno razdvojivi najbolje je zaustaviti trening algoritam kada broj pogrešno klasifikovanih postane dovoljno mali. Za očekivati je da pri klasifikovanju elektronske pošte podaci uvek budu linearno razdvojivi.

Algoritam 4: Treniranje perceptrona

1. Inicijalizuj $w$ i $b$ na nasumično odabrane vrednosti vrednosti ili na nulu
2. Pronadi trening primer $(w, c)$ za koji važi $\text{sign}(w^T x + b) \neq c$. Ukoliko ne postoji takav primer trening je završen. Sačuvaj poslednje $w$ i $b$ i stani. U suprotnom idi na sledeći korak.
3. Ažuriraj $(w, c)$: $w := w + cx$, $b := b + c$. Idi na prethodni korak.

Algoritam 5: Klasifikacija perceptron pravilom

1. Za datum poruku $x$, odredi njenu klasu kao $\text{sign}(w^T x + b)$

---

9Ovo važi zbog toga što je vektor obeležja koji koristimo mnogo veći od broja trening primera. Znano da su u $n$-dimenzionalnom prostoru $n + 1$ tačka linearno razdvojive u svakom slučaju. Činjenica da je vektor obeležja veći od trening primera može značiti da imamo "previše obeležja" i to nije uvek poželjno (pogledati [15]).

27
3.4 Veštacke neuronske mreže

3.4.2 Višeslojni perceptron

Višeslojni perceptron je funkcija koja se može vizualizovati kao mreža sa nekoliko slojeva neurona povezanih unapred. Neuroni u prvom sloju se nazivaju ulazni neuroni i reprezentuju ulazne promenljive. Neuroni u poslednjem sloju se nazivaju izlazni neuroni i pružaju funkciji rezultujuću vrednost. Slojevi između prvog i poslednjeg se nazivaju skriveni slojevi. Svaki neuron u mreži je sličan perceptronu jer na osnovu ulaznih vrednosti \( x_1, x_2, ..., x_k \) izračunava izlaznu vrednost \( o \) korišćenjem formule:

\[
o = \phi\left(\sum_{i=1}^{k} w_i x_k + b\right)\]

gde \( w_i \) predstavljaju težine, \( b \) slobodnog koeficijenta neurona i \( \phi \) je određena nelinearna funkcija. Najčešće se za \( \phi(x) \) uzima \( \frac{1}{1+e^{-x}} \) ili tanh(\( x \)).

![Diagram višeslojnog perceptrona](image)

**Slika 8:** Struktura višeslojnog perceptrona

Treniranje višeslojnog perceptrona znači pretragu za takvim težinama i biasom svih neurona za koje će mreža imati najmanju moguću grešku na trening skupu. To jest, ukoliko označimo funkciju koju implementira mreža kao \( f(x) \), tada u cilju treniranja mreže moramo da pronademo parametre koji minimalizuju ukupnu grešku pri treniranju:

\[
E(f) = \sum_{i=1}^{n} (f(x_i) - c_i)^2
\]
3.4 Veštačke neuronske mreže

3.4. Veštačke neuronske mreže

Veštačke neuronske mreže (VNM) su kompaktni i često efikasni modeli za rešenje problema klasifikacije i regresije. Ovaj dio dokumenta se fokusira na algoritme filtriranja elektronske pošte, a posebno na višeslojne perceptronove. Ova tip mreža koristi se za klasifikaciju i prepoznavanje neželjenih poruka.

3.4.1 Veštačke neuronske mreže

gde su \((x_i, c_i)\) trening primeri. Ova minimizacija se može obaviti bilo kojim iterativnim algoritmom optimizacije. Najpopularniji je jednostavan gradijentni spust koji se u konkretnijem slučaju naziva *propagiranje greške unazad*. Detaljna specifikacija ovog algoritma je opisana u mnogim knjigama i radovima (pogledati [21] [19] [28]).

Višeslojni perceptronovi su nelinearni klasifikatori jer su njihovi modeli nelinearne granice odluke između klasa. Kao što je malo pre pomenuto, trening podaci koje koristimo su linearno razdvojivi i korišćenje nelinearne granice odlučivanja teško da može poboljšati performanse. Zbog toga su najbolji rezultati koje možemo da očekujemo rezultati jednostavnog perceptrona.

Od svih algoritama mašinskog učenja, višeslojni perceptron verovatno sadrži najveći broj parametara koji se moraju štovati *ad hoc*. Nije najjasnije koliko skrivenih neurona treba da sadrži i koje parametre izabrati za algoritam propagacije unazad u cilju postizanja dobre generalizacije. Mnogi radovi i knjige su napisani pokrivajući ovu temu ali trening višeslojnog perceptrona i dalje nije najjasnije. Na sreću to nas ne sprečava da ovaj metod učenja koristimo jer se uspešno primenjuje u zadacima filtriranja neželjnenih poruka (pogledati [29]).

3.4.3 Karakteristike veštačkih neuronskih mreža

Sledi generalni rezime karakteristika veštačkih neuronskih mreža [27]:


2. Veštačke neuronske mreže mogu da se izbore sa reduantnim karakteristikama podataka jer se težine automatski uče tokom procesa treniranja. U slučaju reduantnih karakteristika algoritam nastoji da težine učini malim.


5. Treniranje veštačkih neuronskih mreža je proces koji troši mnogo vremena, posebno kada je broj skrivenih čvorova veliki. Ipak test primeri se mogu klasifikovati brzo.
3.5 Tehnika maksimalne entropije

Maksimalna entropija je klasičan model koji se često koristi u procesiranju prirodnih jezika [6]. Princip je da se pronade odgovarajuća raspodela verovatnoće \( p(a,b) \) koja maksimizuje entropiju:

\[
H(p) = - \sum_{x \in A \times B} p(x) \log p(x)
\]

gde A označava skup mogućih klasa, a B skup svih mogućih vrednosti vektora obeležja. Ova maksimizacija bi trebalo da održi \( p \) u skladu sa informacijama u trening skupu.

\[
p(a,b) = \frac{1}{Z(b)} \prod_{j=1}^{k} \alpha_{f_j(a,b)}
\]

gde je \( k \) veličina vektora obeležja i

\[
Z(b) = \sum_{a} \prod_{j=1}^{k} \alpha_{f_j(a,b)}
\]

je normalizacioni faktor koji osigurava

\[
\sum_{a} p(a,b) = 1.
\]

Eksponencijalni oblik funkcije raspodele je sledećeg oblika

\[
p(a,b) = \frac{1}{Z(\alpha_1,...,\alpha_m)} e^{[\alpha_1 f_1(a,b)+...+\alpha_m f_m(a,b)]}
\]

\( \alpha_j \) se može izračunati korišćenjem generalizovanog iterativnog skaliranja [17]. Funkcija \( f \) se definiše na sledeći način:

\[
f_{cp,a'}(a,b) = \begin{cases} 
1, \quad \text{ako } a = a' \text{ i } cp(b) = true. \\
0, \quad \text{inače.}
\end{cases}
\]

gde \( cp \) preslikava parove \((a,b)\) u \{true, false\}. Rezultati prikazani u [32] pokazuju da je stopa greške bolja nego kod Bajesovog klasifikatora kada trening primeri rastu.
4 | Podaci

U ovom poglavlju su opisani dostupni korpusi koji su korišćeni za testiranje algoritama koji su implementirani.

<table>
<thead>
<tr>
<th>Korpus</th>
<th>Poruke</th>
<th>Spam</th>
<th>Legitimne</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU1</td>
<td>1099</td>
<td>481</td>
<td>618</td>
</tr>
<tr>
<td>PU2</td>
<td>721</td>
<td>142</td>
<td>579</td>
</tr>
<tr>
<td>PU3</td>
<td>4139</td>
<td>1826</td>
<td>2313</td>
</tr>
<tr>
<td>PUA</td>
<td>1142</td>
<td>571</td>
<td>571</td>
</tr>
<tr>
<td>Enron-Spam</td>
<td>33716</td>
<td>17171</td>
<td>16545</td>
</tr>
<tr>
<td>Ling-Spam</td>
<td>2893</td>
<td>481</td>
<td>2412</td>
</tr>
</tbody>
</table>

**Tabela 5:** Raspodela poruka u korpusima

4.1 | Ling spam korpus


Iako su spam i legitimne poruke izvučene iz različitih izvora, a informacije korisne za neke filtre su uklonjene, Ling spam korpus je predstavljao veoma realnu procenu u vreme njegovog publikovanja i veliki broj studija ga je koristio za evaluaciju.

4.2 | PU1, PU2, PU3 i PUA spam korpusi

Skup podataka koji je korišćen pri istraživanju je *PU1 korpus* [11]. Korpus sadrži 1099 poruka, od kojih su 481 spam. Sastoji se od 4 direktorijuma koji odgovaraju kodiranim verzijama korpusa:

- **bare:** Lemmatiser disabled, stop-list disabled.
- **lemm:** Lemmatiser enabled, stop-list disabled.
- **lemm_stop:** Lemmatiser enabled, stop-list enabled.
- **stop:** Lemmatiser disabled, stop-list enabled.

Svaki od ova četiri direktorijuma sadrži po 10 poddirektorijuma, koji odgovaraju za 10 delova korpusa koji su korišćeni pri evaluaciji. U svakom ponavljanju jedan deo je rezervisan za testiranje dok su drugih 9 korišćeni za učenje. Svaki od ovih 10 poddirektorijuma sadrži i spam i legitimne poruke. Fajlovi čija su imena oblika *spmsg*.txt
predstavljaju spam poruke, dok fajlovi čiji su nazivi oblika *legit*.txt predstavljaju legitimne poruke.

Sve poruke unutar korpusa su predprocesirane: svi prilozi, HTML tagovi, zaglavlja izuzimajući Subject su uklonjeni dok su reči kodirane brojevima.

Razlikuje se od Ling spam korpusa u dva navedena aspekta, ali su inače slični:

- izvedeni su od realnih poruka elektronske pošte koje su slate pojedincima
- sadržaj tih privatnih poruka je kodiran tako da je svaka posebna reč zamenjena brojem

Subject: 10834 82 22908 18063 13824

PU2 korpus je najmanji, dok je PU3 korpus značajno veći od PU1 i PU2 korpusa (videti tabelu 5). PUA korpus sadrži 1142 poruka od čega je polovina legitimna dok polovina predstavlja spam. Sva tri korpusa sadrže poruke kodirane brojevima poput PU1 korpusa zbog privatnosti.

4.3 Enron spam korpus


Slika 9: Primer kodirane poruke iz PU1 korpusa

PU2 korpus je najmanji, dok je PU3 korpus značajno veći od PU1 i PU2 korpusa (videti tabelu 5). PUA korpus sadrži 1142 poruka od čega je polovina legitimna dok polovina predstavlja spam. Sva tri korpusa sadrže poruke kodirane brojevima poput PU1 korpusa zbog privatnosti.
Implementacija

U ovom poglavlju će biti prikazana implementacija algoritama mašinskog učenja za klasifikaciju elektronske pošte. Algoritmi su implementirani u programskom jeziku C++ sa ciljem testiranja na korpusima elektronske pošte koji su opisani u poglavlju 4. Svraha implementacije je gruba procena performans, brzine i vremenske zahtevnosti algoritama. Kod programa se može pogledati u dodatnom poglavlju [A].

Implementirani su sledeći algoritmi:
- Naivni Bajesov klasifikator
- K najbližih suseda
- Metod podržavajućih vektora
- Neuronske mreže (perceptron)

Glavni izvršni fajl će sve implementirane algoritme primeniti na poruke u korpusu i ispišaće statistiku. Podešavanje parametara programa se vrši unutar fajla parametri.txt. Podešavanja su zapisana u parovima PARAMETAR VREDNOST, pojedinačno po liniji. Linije koje počinju # predstavljaju komentare. Podešavanja data konfiguracionim fajlom se mogu premostiti na komandnoj liniji pri pozivanju izvršnog fajla na sledeći način ./master --PARAMETAR=VREDNOST.

U tabeli 6 su prikazani parametri programa sa podrazumevanim vrednostima i opisom.

<table>
<thead>
<tr>
<th>Naziv parametra</th>
<th>Podrazumevana vrednost</th>
<th>Opis</th>
</tr>
</thead>
<tbody>
<tr>
<td>direktorijs-korporusa</td>
<td>../korpusi/PU123ACorpora/pu_corpora_public/pu1</td>
<td>direktorijs koji sadrži željeni korpus</td>
</tr>
<tr>
<td>deo-korporusa</td>
<td>1</td>
<td>deo korpusa koji ostavljamo za validaciju</td>
</tr>
<tr>
<td>odlike-od</td>
<td>1</td>
<td>odlike koje će biti izdvojene od ove reči</td>
</tr>
<tr>
<td>odlike-do</td>
<td>30000</td>
<td>odlike koje će biti izdvojene do ove reči</td>
</tr>
<tr>
<td>produzeni-validacioni-skup</td>
<td>0</td>
<td>da li će trening instance biti dođete skupu za validaciju zajedno sa validacionim instancama (0 - ne, 1 - da)</td>
</tr>
<tr>
<td>nb-lambda</td>
<td>8</td>
<td>lambda Naivnog Bajesovog klasifikator</td>
</tr>
<tr>
<td>knn-k</td>
<td>50</td>
<td>k najbližih suseda</td>
</tr>
<tr>
<td>knn-l</td>
<td>30</td>
<td>ako je l ili više poruka među k najbližih suseda poruke x spam klasifikuj x kao spam</td>
</tr>
<tr>
<td>svm-mm</td>
<td>1</td>
<td>korišćenje meke marginje za SVM (0 - ne, 1 - da)</td>
</tr>
<tr>
<td>klasifikator se testira</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>------------------------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>perceptron</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>naiveBayes</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>svm</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>knn</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>svm-naiveBayes</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>svm-naiveBayes-perceptron</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Tabela 6**: Parametri programa

**Slika 10**: Dijagram klasa 1. deo
5.1 Biblioteka SVM\textsuperscript{light}

Biblioteka SVM\textsuperscript{light} sadrži podršku za klasifikaciju instanci metodom podrzavajućih vektora, uz niz dodatnih alata koji olakšavaju pripremu ulaznih podataka i izbor ispravnih parametara. Biblioteka je implementirana u čistom C-u, a ne u C++ tako da je bilo neophodno uraditi sledeće:

```c
extern "C" {
    #include "svm_common.h"
    #include "svm_learn.h"
}
```


Slika 11: Dijagram klasa 2. deo
Kombinovanje klasifikatora

5.2 Kombinovanje klasifikatora

videti i u samoj implementaciji adapter klase u dodatnom poglavlju A.11.

- **svm_learn** – modul za učenje
- **svm_classify** – modul za klasifikaciju

Glavne odlike programa su sledeće:

✓ brz optimizacioni algoritam
✓ rešava probleme klasifikacije i regresije
✓ izračunava XiAlpha-procenu stope greške, preciznost i odziv
✓ može trenirati SVM sa modelima cena
✓ efikasno obrađuje više hiljada podržavajućих vektora
✓ obrađuje i nekoliko desetina hiljada trening primera

5.2 Kombinovanje klasifikatora

Kombinovanjem klasifikatora dobijamo **Klasifikator1od2**, sa većom preciznošću ukoliko koristimo sledeće klasifikaciono pravilo:

*Klasifikuj poruku x kao neželjenu ukoliko je jedan od dva klasifikatora klasifikuje kao neželjenu, u suprotnom klasifikuj poruku kao legitimnu.*

Ovo može delovati opasno jer legitimna poruka može da se klasifikuje kao neželjena ukoliko je samo jedan klasifikator klasifikuje kao neželjenu, ali pretpostavka je da će izabrani klasifikatori imati nizak stepen lažno pozitivnih instanci. Pri evaluaciji su odabrani metodi podržavajućih vektor a i Naivni Bajes jer se pokazalo u prvih nekoliko korpusa da oni imaju najniži stepen lažno pozitivnih instanci.

Ekvivalentno tome je realizovano pravilo **Klasifikator2od3**, koje određuje klasu instance prema odlukama tri klasifikatora sledećim pravilom:

*Instanca se klasifikuje u klasu C ukoliko se najmanje dva klasifikatora odluče za klasu C.*

Pri evaluaciji je pored metoda podržavajućih vektor a i Naivnog Bajesa odabrano perceptron kao treći klasifikator zbog svoje visoke preciznosti.

36
6 | Eksperimentalni rezultati

U ovom poglavlju su predstavljeni eksperimentalni rezultati testiranja implementiranih algoritama nad različitim korpusima elektronske pošte.

6.1 | PU1 korpus

6.1.1 | Naivni Bajesov klasifikator

Pozivanje programa za klasifikaciju poruka PU1 korpusa Naivnim Bajesovim klasifikatorom za $\lambda = 5$.

```
./master --direktorijum-korpusa=./korpusi/PU123ACorpora/pu_corpora_public/PU1 --naivnibajes=1 --nb-lambda=5
```

<table>
<thead>
<tr>
<th>Stvarna klasa</th>
<th>Predviđena klasa</th>
<th>Ukupno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klasa = SPAM</td>
<td>Klasa = SPAM</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Klasa = LEGITIMNO</td>
<td>13</td>
</tr>
<tr>
<td>Klasa = LEGITIMNO</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Klasa = SPAM</td>
<td>35</td>
</tr>
</tbody>
</table>

$TP + FN = 74$

$FP + TN = 35$

$N = 109$

**Tabela 7:** Matrica konfuzije PU1 korpusa prilikom klasifikacije Naivnim Bajesovim klasifikatorom za $\lambda = 5$

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>$f_1$ mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 5$)</td>
<td>0.553436 s</td>
<td>0.014049 s</td>
<td>0.8807</td>
<td>0.1192</td>
<td>1</td>
<td>0.824324</td>
<td>0.903704</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 50$)</td>
<td>0.552885 s</td>
<td>0.013871 s</td>
<td>0.8624</td>
<td>0.137615</td>
<td>1</td>
<td>0.802632</td>
<td>0.890511</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 500$)</td>
<td>0.550187 s</td>
<td>0.013875 s</td>
<td>0.8532</td>
<td>0.146789</td>
<td>1</td>
<td>0.792208</td>
<td>0.884058</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 999$)</td>
<td>0.551718 s</td>
<td>0.013794 s</td>
<td>0.8532</td>
<td>0.146789</td>
<td>1</td>
<td>0.792208</td>
<td>0.884058</td>
</tr>
</tbody>
</table>

**Tabela 8:** Testiranje performansi klasifikatora PU1 korpusa Naivnim Bajesovim klasifikatorom
6.1.2  K naj bližih suseda

Pozivanje programa za klasifikaciju poruka PU1 korpusa algoritmom K naj bližih suseda za $k = 50$ i $l = 30$.

```
./master --direktorijum-korpusa=../korpusi/PU123ACorpora/pu_corpora_public/pu1
--knn=1 --knn-k=50 --knn-l=30
```

<table>
<thead>
<tr>
<th>Stvarna klasa</th>
<th>Predviđena klasa</th>
<th>Klasa = SPAM</th>
<th>Klasa = LEGITIMNO</th>
<th>Ukupno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klasa = SPAM</td>
<td></td>
<td>58</td>
<td>10</td>
<td>TP + FN = 68</td>
</tr>
<tr>
<td>Klasa = LEGITIMNO</td>
<td></td>
<td>3</td>
<td>38</td>
<td>FP + TN = 41</td>
</tr>
<tr>
<td>Ukupno</td>
<td></td>
<td>TP + FP = 61</td>
<td>FN + TN = 48</td>
<td>N = 109</td>
</tr>
</tbody>
</table>

**Tabela 9:** Matrica konfuzije PU1 korpusa prilikom klasifikacije algoritmom K naj bližih suseda za $k = 50$ i $l = 30$

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greske</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>K naj bližih suseda ($k = 50$ $i l = 30$)</td>
<td>0.006408 s</td>
<td>0.461033 s</td>
<td>0.8807</td>
<td>0.1192</td>
<td>0.852941</td>
<td>0.899225</td>
<td></td>
</tr>
<tr>
<td>K naj bližih suseda ($k = 50$ $i l = 33$)</td>
<td>0.005354 s</td>
<td>0.442313 s</td>
<td>0.8532</td>
<td>0.146789</td>
<td>0.792208</td>
<td>0.884058</td>
<td></td>
</tr>
<tr>
<td>K naj bližih suseda ($k = 50$ $i l = 35$)</td>
<td>0.005546 s</td>
<td>0.443605 s</td>
<td>0.7982</td>
<td>0.2018</td>
<td>1.73494</td>
<td>0.847222</td>
<td></td>
</tr>
<tr>
<td>K naj bližih suseda ($k = 30$ $i l = 20$)</td>
<td>0.005621 s</td>
<td>0.441475 s</td>
<td>0.8624</td>
<td>0.137615</td>
<td>0.819444</td>
<td>0.887218</td>
<td></td>
</tr>
<tr>
<td>K naj bližih suseda ($k = 60$ $i l = 40$)</td>
<td>0.005629 s</td>
<td>0.443173 s</td>
<td>0.8257</td>
<td>0.174312</td>
<td>0.7625</td>
<td>0.865248</td>
<td></td>
</tr>
</tbody>
</table>

**Tabela 10:** Testiranje performansi klasifikatora PU1 korpusa algoritmom K naj bližih suseda
6.1.3  **Metod podržavajućih vektora**

Pozivanje programa za klasifikaciju poruka PU1 korpusa *Metodom podržavajućih vektora* bez meke margine.

```
./master --direktorijum-korpusa=../korpusi/PU123ACorpora/pu_corpora_public/pu1 --svm=1 --svm-mm=0
```

<table>
<thead>
<tr>
<th>Predviđena klasa</th>
<th>Stvarna klasa</th>
<th>Klasa = SPAM</th>
<th>Klasa = LEGITIMNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klasa = SPAM</td>
<td>59</td>
<td>1</td>
<td>TP + FN = 60</td>
</tr>
<tr>
<td>Klasa = LEGITIMNO</td>
<td>2</td>
<td>47</td>
<td>FP + TN = 49</td>
</tr>
<tr>
<td>Ukupno</td>
<td>TP + FP = 61</td>
<td>FN + TN = 48</td>
<td>N = 109</td>
</tr>
</tbody>
</table>

**Tabela 11**: Matrica konfuzije PU1 korpusa prilikom klasifikacije *Metodom podržavajućih vektora* bez meke margine

<table>
<thead>
<tr>
<th>Predviđena klasa</th>
<th>Stvarna klasa</th>
<th>Klasa = SPAM</th>
<th>Klasa = LEGITIMNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klasa = SPAM</td>
<td>61</td>
<td>7</td>
<td>TP + FN = 68</td>
</tr>
<tr>
<td>Klasa = LEGITIMNO</td>
<td>0</td>
<td>41</td>
<td>FP + TN = 41</td>
</tr>
<tr>
<td>Ukupno</td>
<td>TP + FP = 61</td>
<td>FN + TN = 48</td>
<td>N = 109</td>
</tr>
</tbody>
</table>

**Tabela 12**: Matrica konfuzije PU1 korpusa prilikom klasifikacije *Metodom podržavajućih vektora* sa mekom marginom

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stupa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>F1 mera</th>
<th>Broj podržavajućih vektora</th>
<th>Broj evaluacija kernela</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Metod podržavajućih vektora bez meke marginе</em></td>
<td>0.066 s</td>
<td>0.023 s</td>
<td>0.97</td>
<td>0.0275</td>
<td>0.97</td>
<td>0.983</td>
<td>0.975</td>
<td>254</td>
<td>26797</td>
</tr>
<tr>
<td><em>Metod podržavajućih vektora sa mekom marginom</em></td>
<td>0.071 s</td>
<td>0.043 s</td>
<td>0.94</td>
<td>0.0642</td>
<td>1</td>
<td>0.897</td>
<td>0.945</td>
<td>450</td>
<td>21555</td>
</tr>
</tbody>
</table>

**Tabela 13**: Testiranje performansi klasifikatora PU1 korpusa *Metodom podržavajućih vektora* sa i bez meke margine
6.1.4  **Perceptron**

Pozivanje programa za klasifikaciju poruka PU1 korpusa korišćenjem *Perceptron*.

```
./master --direktorijum-korporusa=../korpusi/PU123ACorpora/pu_corpora_public/pu1 --perceptron=1
```

<table>
<thead>
<tr>
<th>Predviđena klasa</th>
<th>Stvarna klasa</th>
<th>Ukupno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klasa = SPAM</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Klasa = LEGITIMNO</td>
<td>2</td>
<td>47</td>
</tr>
<tr>
<td><strong>Ukupno</strong></td>
<td><strong>TP + FN = 60</strong></td>
<td><strong>FP + TN = 49</strong></td>
</tr>
</tbody>
</table>

**Tabela 14:** Matrica konfuzije PU1 korpusa prilikom klasifikacije korišćenjem *perceptron*

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>0.0124 s</td>
<td>0.0002 s</td>
<td>0.9725</td>
<td>0.0275</td>
<td>0.97</td>
<td>0.9833</td>
<td>0.975</td>
</tr>
</tbody>
</table>

**Tabela 15:** Testiranje performansi klasifikatora PU1 korpusa korišćenjem *perceptron*

6.1.5  **Kombinovanje klasifikatora**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Leža pozitivni</th>
<th>Leža negativni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>0</td>
<td>3</td>
<td>0.62 s</td>
<td>0.05 s</td>
<td>0.97</td>
<td>0.0275</td>
<td>1</td>
<td>0.953125</td>
<td>0.976</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>0</td>
<td>4</td>
<td>0.63 s</td>
<td>0.06 s</td>
<td>0.96</td>
<td>0.0367</td>
<td>1</td>
<td>0.9384</td>
<td>0.978</td>
</tr>
</tbody>
</table>

**Tabela 16:** Testiranje performansi kombinovanih klasifikatora nad PU1 korpusom

Na osnovu svih dobijenih rezultata lako se uočava da klasifikator zasnovan na *perceptron* ima najveću tačnost i najkraće vreme treniranja i validacije, dok ga u stopu prati klasifikator zasnovan na *metodu podržavajućih vektora.*
Slika 12: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU1 korpusom

Slika 13: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU1 korpusom
6.2 PU2 korpus

PU2 korpus je najmanji od korpusa korišćenih za testiranja implementiranih klasifikatora. Rezultati testiranja klasifikatora nad PU2 korpusom dati su u tabeli 17.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Lažno pozitivni</th>
<th>Ležno negativni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stroga greška</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Naivni Bajesov klasifikator</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(λ = 5)</td>
<td>0</td>
<td>11</td>
<td>0.359 s</td>
<td>0.009 s</td>
<td>0.845</td>
<td>0.155</td>
<td>1</td>
<td>0.8382</td>
<td>0.912</td>
</tr>
<tr>
<td>(λ = 50)</td>
<td>0</td>
<td>11</td>
<td>0.359 s</td>
<td>0.009 s</td>
<td>0.845</td>
<td>0.155</td>
<td>1</td>
<td>0.8382</td>
<td>0.912</td>
</tr>
<tr>
<td>(λ = 500)</td>
<td>0</td>
<td>11</td>
<td>0.359 s</td>
<td>0.009 s</td>
<td>0.845</td>
<td>0.155</td>
<td>1</td>
<td>0.8382</td>
<td>0.912</td>
</tr>
<tr>
<td>(λ = 999)</td>
<td>0</td>
<td>11</td>
<td>0.359 s</td>
<td>0.009 s</td>
<td>0.845</td>
<td>0.155</td>
<td>1</td>
<td>0.8382</td>
<td>0.912</td>
</tr>
<tr>
<td>kNN (k = 30, l = 20)</td>
<td>0</td>
<td>12</td>
<td>0.004 s</td>
<td>0.2128 s</td>
<td>0.8309</td>
<td>0.169</td>
<td>1</td>
<td>0.8261</td>
<td>0.9047</td>
</tr>
<tr>
<td>kNN (k = 50, l = 30)</td>
<td>0</td>
<td>13</td>
<td>0.004 s</td>
<td>0.2125 s</td>
<td>0.8169</td>
<td>0.1831</td>
<td>1</td>
<td>0.8142</td>
<td>0.8976</td>
</tr>
<tr>
<td>kNN (k = 20, l = 15)</td>
<td>1</td>
<td>10</td>
<td>0.004 s</td>
<td>0.2131 s</td>
<td>0.8451</td>
<td>0.1549</td>
<td>1</td>
<td>0.8124</td>
<td>0.9106</td>
</tr>
<tr>
<td>kNN (k = 15, l = 10)</td>
<td>2</td>
<td>6</td>
<td>0.004 s</td>
<td>0.2129 s</td>
<td>0.8873</td>
<td>0.1127</td>
<td>0.965</td>
<td>0.9016</td>
<td>0.9322</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke marginе</td>
<td>2</td>
<td>2</td>
<td>0.0688 s</td>
<td>0.0158 s</td>
<td>0.9436</td>
<td>0.0563</td>
<td>0.9649</td>
<td>0.9649</td>
<td>0.9649</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>0</td>
<td>8</td>
<td>0.0507 s</td>
<td>0.0215 s</td>
<td>0.8873</td>
<td>0.1127</td>
<td>1</td>
<td>0.8769</td>
<td>0.9344</td>
</tr>
<tr>
<td>Perceptron</td>
<td>0</td>
<td>2</td>
<td>0.0066 s</td>
<td>0.0002 s</td>
<td>0.9718</td>
<td>0.0281</td>
<td>1</td>
<td>0.9661</td>
<td>0.9827</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>0</td>
<td>8</td>
<td>0.41 s</td>
<td>0.03 s</td>
<td>0.89</td>
<td>0.11</td>
<td>1</td>
<td>0.8769</td>
<td>0.9344</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>0</td>
<td>8</td>
<td>0.42 s</td>
<td>0.03 s</td>
<td>0.89</td>
<td>0.11</td>
<td>1</td>
<td>0.8769</td>
<td>0.9344</td>
</tr>
</tbody>
</table>

Tabela 17: Rezultati testiranja implementiranih klasifikatora nad PU2 korpusom

Iz priloženih rezultata se jasno može videti da klasifikator zasnovan na perceptronu daje najbolje rezultate, kako u pogledu performansi tako i u pogledu tačnosti. Od 71 instance za validaciju samo 2 su lažno negativne (spam klasifikovan kao legitimna poruka), a tačnost iznosi 97.18%.
Slika 14: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU2 korpusom

Slika 15: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU2 korpusom
6.3  PU3 korpus

PU3 korpus je značajno veći od PU1 i PU2 korpusa i sadrži 4139 poruka od čega 1826 predstavlja spam dok su preostalih 2313 poruka legitimne. Rezultati testiranja klasifikatora nad PU3 korpusom dati su u tabeli 18.

<table>
<thead>
<tr>
<th>Algorithm (λ)</th>
<th>Lažno pozitivni</th>
<th>Lažno negativni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stepen greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator (λ = 5)</td>
<td>0</td>
<td>29</td>
<td>3.4588 s</td>
<td>0.05933 s</td>
<td>0.9297</td>
<td>0.0702</td>
<td>1</td>
<td>0.8884</td>
<td>0.941</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 50)</td>
<td>0</td>
<td>33</td>
<td>3.4588 s</td>
<td>0.05933 s</td>
<td>0.9201</td>
<td>0.078</td>
<td>1</td>
<td>0.875</td>
<td>0.933</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 500)</td>
<td>0</td>
<td>36</td>
<td>3.497 s</td>
<td>0.0596 s</td>
<td>0.9128</td>
<td>0.0875</td>
<td>1</td>
<td>0.8652</td>
<td>0.9277</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 999)</td>
<td>0</td>
<td>36</td>
<td>3.497 s</td>
<td>0.0596 s</td>
<td>0.9128</td>
<td>0.0875</td>
<td>1</td>
<td>0.8652</td>
<td>0.9277</td>
</tr>
<tr>
<td>kNN (k = 100, l = 65)</td>
<td>4</td>
<td>61</td>
<td>0.0298 s</td>
<td>9.9754 s</td>
<td>0.8426</td>
<td>0.1574</td>
<td>0.9827</td>
<td>0.78819</td>
<td>0.8747</td>
</tr>
<tr>
<td>kNN (k = 81, l = 51)</td>
<td>6</td>
<td>48</td>
<td>0.0276 s</td>
<td>9.9779 s</td>
<td>0.8629</td>
<td>0.1307</td>
<td>0.974</td>
<td>0.8242</td>
<td>0.8928</td>
</tr>
<tr>
<td>kNN (k = 131, l = 81)</td>
<td>7</td>
<td>59</td>
<td>0.0291 s</td>
<td>9.971 s</td>
<td>0.8402</td>
<td>0.1598</td>
<td>0.9696</td>
<td>0.7915</td>
<td>0.8716</td>
</tr>
<tr>
<td>kNN (k = 61, l = 35)</td>
<td>23</td>
<td>23</td>
<td>0.0296 s</td>
<td>9.9091 s</td>
<td>0.8886</td>
<td>0.1114</td>
<td>0.9004</td>
<td>0.9004</td>
<td>0.9004</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margin</td>
<td>7</td>
<td>10</td>
<td>0.8602 s</td>
<td>0.2145 s</td>
<td>0.9588</td>
<td>0.0411</td>
<td>0.9696</td>
<td>0.9572</td>
<td>0.9634</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>0</td>
<td>13</td>
<td>0.3919 s</td>
<td>0.4487 s</td>
<td>0.9685</td>
<td>0.0314</td>
<td>1</td>
<td>0.9467</td>
<td>0.9726</td>
</tr>
<tr>
<td>Perceptron</td>
<td>7</td>
<td>4</td>
<td>0.2327 s</td>
<td>0.0014 s</td>
<td>0.9734</td>
<td>0.0266</td>
<td>0.9697</td>
<td>0.9824</td>
<td>0.976</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>0</td>
<td>10</td>
<td>3.875 s</td>
<td>0.509 s</td>
<td>0.9758</td>
<td>0.0242</td>
<td>1</td>
<td>0.9585</td>
<td>0.9788</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>0</td>
<td>11</td>
<td>4.2471 s</td>
<td>0.5125 s</td>
<td>0.9734</td>
<td>0.0266</td>
<td>1</td>
<td>0.9545</td>
<td>0.9767</td>
</tr>
</tbody>
</table>

Tabela 18: Rezultati testiranja implementiranih klasifikatora nad PU3 korpusom

Dobijeni rezultati su nešto drugačiji u odnosu na PU2 korpus ali to je i razumljivo imajući u vidu da validacioni skup sadrži 413 instanci. Klasifikator zasnovan na perceptronu i dalje ima veoma visoku tačnost 97.34% kao i kod PU2 korpusa ali je broj lažno pozitivnih instanci 7. Kombinovani klasifikatori postižu bolje rezultate jer ne sadrže lažno pozitivne instance uz isti stepen tačnosti ali je vreme treniranja i validacije znatno veće u odnosu na klasifikator zasnovan na perceptronu.
6.3 PU3 korpus

Eksperimentalni rezultati

Slika 16: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PU3 korpusom

Slika 17: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PU3 korpusom
6.4 PUA korpus

PUA korpus sadrži 1142 poruka od čega je polovina legitimna dok polovina predstavlja spam. Rezultati testiranja klasifikatora nad PUA korpusom dati su u tabeli 19.

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Lažno pozitivi</th>
<th>Lažno negativni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 5$)</td>
<td>1</td>
<td>7</td>
<td>0.7176 s</td>
<td>0.0176 s</td>
<td>0.9298</td>
<td>0.0702</td>
<td>0.9825</td>
<td>0.8888</td>
<td>0.9333</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 50$)</td>
<td>1</td>
<td>7</td>
<td>0.7176 s</td>
<td>0.0176 s</td>
<td>0.9298</td>
<td>0.0702</td>
<td>0.9825</td>
<td>0.8888</td>
<td>0.9333</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 500$)</td>
<td>1</td>
<td>7</td>
<td>0.7176 s</td>
<td>0.0176 s</td>
<td>0.9298</td>
<td>0.0702</td>
<td>0.9825</td>
<td>0.8888</td>
<td>0.9333</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 999$)</td>
<td>1</td>
<td>7</td>
<td>0.7176 s</td>
<td>0.0176 s</td>
<td>0.9298</td>
<td>0.0702</td>
<td>0.9825</td>
<td>0.8888</td>
<td>0.9333</td>
</tr>
<tr>
<td>kNN ($k = 61, l = 56$)</td>
<td>46</td>
<td>0</td>
<td>0.0097 s</td>
<td>0.8046 s</td>
<td>0.6754</td>
<td>0.3245</td>
<td>1</td>
<td>0.6064</td>
<td>0.755</td>
</tr>
<tr>
<td>kNN ($k = 51, l = 30$)</td>
<td>43</td>
<td>0</td>
<td>0.0098 s</td>
<td>0.8046 s</td>
<td>0.6228</td>
<td>0.3772</td>
<td>0.2456</td>
<td>1</td>
<td>0.3944</td>
</tr>
<tr>
<td>kNN ($k = 41, l = 25$)</td>
<td>40</td>
<td>0</td>
<td>0.0094 s</td>
<td>0.8058 s</td>
<td>0.6491</td>
<td>0.3509</td>
<td>0.2982</td>
<td>1</td>
<td>0.4594</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margin</td>
<td>4</td>
<td>1</td>
<td>0.1896 s</td>
<td>0.0405 s</td>
<td>0.9561</td>
<td>0.0438</td>
<td>0.9298</td>
<td>0.9814</td>
<td>0.955</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>0</td>
<td>11</td>
<td>0.1301 s</td>
<td>0.0772 s</td>
<td>0.9435</td>
<td>0.0905</td>
<td>1</td>
<td>0.8382</td>
<td>0.912</td>
</tr>
<tr>
<td>Perceptron</td>
<td>5</td>
<td>1</td>
<td>0.0308 s</td>
<td>0.0004 s</td>
<td>0.9473</td>
<td>0.0526</td>
<td>0.9123</td>
<td>0.9813</td>
<td>0.9454</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>1</td>
<td>5</td>
<td>0.8388 s</td>
<td>0.0946 s</td>
<td>0.9474</td>
<td>0.0526</td>
<td>0.9824</td>
<td>0.918</td>
<td>0.949</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>1</td>
<td>5</td>
<td>0.8732 s</td>
<td>0.0951 s</td>
<td>0.9474</td>
<td>0.0526</td>
<td>0.9824</td>
<td>0.918</td>
<td>0.949</td>
</tr>
</tbody>
</table>

**Tabela 19:** Rezultati testiranja implementiranih klasifikatora nad PUA korpusom

Klasifikator $k$ naj bližih suseda daje iznenađujuće loše rezultate. Lažno pozitivne instance su eliminisane tek kada je parametar $l$ vrlo blizak parametru $k$ što dovodi do zaključka da su instance korpusa nepovoljno rasprostranjene u prostoru za upotrebu ovog algoritma dok su se ostali algoritmi dosta dobro pokazali sa tačnošću većom od 90%. Najbolje se pokazao metod podržavajućih vektora sa mekom marginom koji nema lažno pozitivnih instanci pri čemu je vreme treniranja i validacije nisko.
Slika 18: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad PUA korpusom

Slika 19: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad PUA korpusom
6.5 Ling spam korpus

**Ling spam korpus** sastoji se od 481 spam poruke kombinovanih sa 2412 legitimnih poruka. Za razliku od prethodnih korpusa poruke nisu kodirane brojevima i zato je neophodno modifikovati funkciju `IzdvajanjeOdlikaKorpusa` u fajlu `IzdvajanjeOdlika.h` kao i funkciju `citajInstanci IzFajla` zbog određivanja kategorija instanci jer su nazivi fajlova drugaćiju u odnosu na PU korpus. Rezultati testiranja klasifikatora nad Ling spam korpusom dati su u tabeli 20.

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Leživo pozitivni</th>
<th>Leživo negativni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tračnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>$f_1$ mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator $(\lambda = 5)$</td>
<td>4</td>
<td>21</td>
<td>4.9359 s</td>
<td>0.0947 s</td>
<td>0.9135</td>
<td>0.0865</td>
<td>0.983</td>
<td>0.9186</td>
<td>0.9499</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator $(\lambda = 50)$</td>
<td>2</td>
<td>22</td>
<td>4.8562 s</td>
<td>0.0955 s</td>
<td>0.9169</td>
<td>0.0830</td>
<td>0.9917</td>
<td>0.9157</td>
<td>0.9522</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator $(\lambda = 500)$</td>
<td>2</td>
<td>22</td>
<td>4.8562 s</td>
<td>0.0955 s</td>
<td>0.9169</td>
<td>0.0830</td>
<td>0.9917</td>
<td>0.9157</td>
<td>0.9522</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator $(\lambda = 999)$</td>
<td>2</td>
<td>22</td>
<td>4.8562 s</td>
<td>0.0955 s</td>
<td>0.9169</td>
<td>0.0830</td>
<td>0.9917</td>
<td>0.9157</td>
<td>0.9522</td>
</tr>
<tr>
<td>$kNN$ $(k = 151, l = 57)$</td>
<td>0</td>
<td>8</td>
<td>0.0164 s</td>
<td>2.231 s</td>
<td>0.9723</td>
<td>0.0277</td>
<td>1</td>
<td>0.9678</td>
<td>0.9836</td>
</tr>
<tr>
<td>$kNN$ $(k = 101, l = 43)$</td>
<td>0</td>
<td>14</td>
<td>0.0155 s</td>
<td>2.4242 s</td>
<td>0.9515</td>
<td>0.0484</td>
<td>1</td>
<td>0.9451</td>
<td>0.9717</td>
</tr>
<tr>
<td>$kNN$ $(k = 71, l = 31)$</td>
<td>0</td>
<td>13</td>
<td>0.0148 s</td>
<td>2.2015 s</td>
<td>0.955</td>
<td>0.045</td>
<td>1</td>
<td>0.9488</td>
<td>0.9737</td>
</tr>
<tr>
<td>$kNN$ $(k = 51, l = 23)$</td>
<td>0</td>
<td>14</td>
<td>0.0155 s</td>
<td>2.4242 s</td>
<td>0.9515</td>
<td>0.0484</td>
<td>1</td>
<td>0.9451</td>
<td>0.9717</td>
</tr>
<tr>
<td>Metod podržavajućih vektor</td>
<td>4</td>
<td>3</td>
<td>0.1047 s</td>
<td>0.0466 s</td>
<td>0.9757</td>
<td>0.024</td>
<td>0.9834</td>
<td>0.9875</td>
<td>0.9854</td>
</tr>
<tr>
<td>Metod podržavajućih vektor sa mekom marginom</td>
<td>0</td>
<td>9</td>
<td>0.1067 s</td>
<td>0.083 s</td>
<td>0.9688</td>
<td>0.0311</td>
<td>1</td>
<td>0.964</td>
<td>0.982</td>
</tr>
<tr>
<td>Perceptron</td>
<td>1</td>
<td>2</td>
<td>0.037 s</td>
<td>0.0004 s</td>
<td>0.9896</td>
<td>0.01</td>
<td>0.9958</td>
<td>0.9917</td>
<td>0.9937</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>4</td>
<td>4</td>
<td>4.91 s</td>
<td>0.17 s</td>
<td>0.9723</td>
<td>0.027</td>
<td>0.9834</td>
<td>0.9834</td>
<td>0.9834</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>0</td>
<td>4</td>
<td>4.875 s</td>
<td>0.174 s</td>
<td>0.986</td>
<td>0.0138</td>
<td>1</td>
<td>0.984</td>
<td>0.992</td>
</tr>
</tbody>
</table>

Tabela 20: Rezultati testiranja implementiranih klasifikatora nad Ling spam korpusom

Veličina validacionog skupa je 289 instanci. Prilikom testiranja klasifikatora korišćen je folder `lemm_stop` u kojem su nad porukama izvršene lematizacije\(^{11}\) i stop lista reči\(^{12}\).

---


\(^{12}\) Reči koje su filtrirane pre ili posle procesiranja prirodnih jezika. Stop reči najčešće predstavljaju najkorišćenije reči u jeziku kao sto su engleske reči the, is, at, which, on itd.
6.5 Ling spam korpus

Eksperimentalni rezultati

- NB
- kNN
- SVM
- SVMmm
- Perceptron
- SVM/NB
- SVM/NB/Per

Slika 20: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad Ling spam korpusom

Slika 21: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad Ling spam korpusom
6.6 Enron spam korpus

Enron spam korpus je naslednik Ling Spam i PU korpusa koji sadrži hronološki pode- ljene e-mail poruke koje su primali šest Enronovih zaposlenih kombinovanih sa spamom iz različitih izvora. Korpus se sastoji iz paketa od šest skupova podataka za trening i testiranje u kojima su poruke hronološki sortirane.

6.6.1 enron1

<table>
<thead>
<tr>
<th>Legitima posta</th>
<th>Spam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vlasnik: farmer-d</td>
<td>Vlasnik: GP</td>
</tr>
<tr>
<td>Ukupan broj: 3672 mejlova</td>
<td>Ukupan broj: 1500 mejlova</td>
</tr>
<tr>
<td>Datum prvog mejla: 1999-12-10</td>
<td>Datum prvog mejla: 2003-12-18</td>
</tr>
<tr>
<td>Datum poslednjeg mejla: 2002-01-11</td>
<td>Datum poslednjeg mejla: 2005-09-06</td>
</tr>
<tr>
<td>Brisanje slicnih: Ne</td>
<td>Brisanje slicnih: Ne</td>
</tr>
<tr>
<td>Enkodiranje: Ne</td>
<td>Enkodiranje: Ne</td>
</tr>
</tbody>
</table>

Spam:Legitimne poruke odnos ≈ 1:2.5
Ukupan broj mejlova (legitimi + spam): 5172

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Ložno pozitivni</th>
<th>Ložno negativni</th>
<th>Vreme treninga</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>$f_1$ mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 5$)</td>
<td>5</td>
<td>50</td>
<td>8.2027 s</td>
<td>0.1517 s</td>
<td>0.89</td>
<td>0.11</td>
<td>0.9864</td>
<td>0.8792</td>
<td>0.9297</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 50$)</td>
<td>4</td>
<td>52</td>
<td>8.4335 s</td>
<td>0.1514 s</td>
<td>0.888</td>
<td>0.112</td>
<td>0.9891</td>
<td>0.8753</td>
<td>0.9287</td>
</tr>
<tr>
<td>kNN ($k = 301, l = 150$)</td>
<td>3</td>
<td>111</td>
<td>0.011 s</td>
<td>3.97 s</td>
<td>0.772</td>
<td>0.228</td>
<td>0.9918</td>
<td>0.7673</td>
<td>0.8652</td>
</tr>
<tr>
<td>kNN ($k = 301, l = 156$)</td>
<td>0</td>
<td>121</td>
<td>0.0118 s</td>
<td>3.9652 s</td>
<td>0.758</td>
<td>0.242</td>
<td>1</td>
<td>0.7531</td>
<td>0.8591</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margin</td>
<td>-</td>
<td>-</td>
<td>&gt; 10 min</td>
<td>Najverovatnije podaci nisu linearno razdvojivi!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metod podržavajućih vektor sa mekom marginom</td>
<td>4</td>
<td>29</td>
<td>0.367 s</td>
<td>0.2805 s</td>
<td>0.934</td>
<td>0.066</td>
<td>0.9892</td>
<td>0.9264</td>
<td>0.9567</td>
</tr>
<tr>
<td>Perceptron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>7</td>
<td>20</td>
<td>8.5609 s</td>
<td>0.4428 s</td>
<td>0.946</td>
<td>0.054</td>
<td>0.981</td>
<td>0.9476</td>
<td>0.964</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabela 21: Rezultati testiranja implementiranih klasifikatora nad enron1 korpusom

50
6.6 Enron spam korpus

**Slika 22:** Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron1 korpusom

**Slika 23:** Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron1 korpusom
## 6.6 Enron spam korpus

### 6.6.2 enron2

**Legitima posta**

- Vlasnik: kaminski-v
- Ukupni broj: 4361 mejlova
- Datum prvog mejla: 1999-12-10
- Datum poslednjeg mejla: 2001-05-22
- Brisanje slicnih: Ne
- Enkodiranje: Ne

**Spam**

- Vlasnik: SpamAssassin + HoneyPot
- Ukupni broj: 1496 mejlova
- Datum prvog mejla: 2001-05-25
- Datum poslednjeg mejla: 2005-07-22
- Brisanje slicnih: Da
- Enkodiranje: Ne

Spam:Legitima poruke odnos $\approx 1 : 3$

Ukupni broj mejlova (legitimi + spam): 5857

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Ljato pozitivni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 5$)</td>
<td>33</td>
<td>70</td>
<td>11.2648 s</td>
<td>0.1953 s</td>
<td>0.8283</td>
<td>0.1716</td>
<td>0.9271</td>
<td>0.8571</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 50$)</td>
<td>28</td>
<td>75</td>
<td>11.2931 s</td>
<td>0.1959 s</td>
<td>0.8233</td>
<td>0.1716</td>
<td>0.9382</td>
<td>0.85</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 500$)</td>
<td>27</td>
<td>70</td>
<td>11.2209 s</td>
<td>0.1948 s</td>
<td>0.8216</td>
<td>0.1783</td>
<td>0.9404</td>
<td>0.8419</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ($\lambda = 999$)</td>
<td>26</td>
<td>80</td>
<td>11.2553 s</td>
<td>0.1954 s</td>
<td>0.8233</td>
<td>0.1766</td>
<td>0.9426</td>
<td>0.8422</td>
</tr>
<tr>
<td>kNN ($k = 201, l = 90$)</td>
<td>0</td>
<td>124</td>
<td>0.0175 s</td>
<td>7.8313 s</td>
<td>0.7933</td>
<td>0.2033</td>
<td>1</td>
<td>0.7851</td>
</tr>
<tr>
<td>kNN ($k = 201, l = 100$)</td>
<td>0</td>
<td>135</td>
<td>0.0165 s</td>
<td>7.8189 s</td>
<td>0.775</td>
<td>0.225</td>
<td>1</td>
<td>0.7714</td>
</tr>
<tr>
<td>kNN ($k = 201, l = 80$)</td>
<td>4</td>
<td>93</td>
<td>0.0173 s</td>
<td>7.8501 s</td>
<td>0.8383</td>
<td>0.1616</td>
<td>0.9912</td>
<td>0.8284</td>
</tr>
<tr>
<td>kNN ($k = 301, l = 130$)</td>
<td>0</td>
<td>118</td>
<td>0.0169 s</td>
<td>7.8623 s</td>
<td>0.8033</td>
<td>0.1967</td>
<td>1</td>
<td>0.7933</td>
</tr>
<tr>
<td>Metod podržavajučih vektora bez meke margin</td>
<td>23</td>
<td>12</td>
<td>0.8559 s</td>
<td>0.1817 s</td>
<td>0.9492</td>
<td>0.0583</td>
<td>0.9492</td>
<td>0.9728</td>
</tr>
<tr>
<td>Metod podržavajučih vektora sa mekom marginom</td>
<td>1</td>
<td>25</td>
<td>0.3545 s</td>
<td>0.4186 s</td>
<td>0.9567</td>
<td>0.0433</td>
<td>0.9978</td>
<td>0.9476</td>
</tr>
<tr>
<td>Perceptron</td>
<td>23</td>
<td>6</td>
<td>0.0808 s</td>
<td>0.0007 s</td>
<td>0.9516</td>
<td>0.0483</td>
<td>0.9492</td>
<td>0.9862</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>34</td>
<td>16</td>
<td>11.5764 s</td>
<td>0.6133 s</td>
<td>0.9167</td>
<td>0.0833</td>
<td>0.9249</td>
<td>0.9632</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>13</td>
<td>17</td>
<td>11.7054 s</td>
<td>0.6169 s</td>
<td>0.95</td>
<td>0.05</td>
<td>0.9713</td>
<td>0.9628</td>
</tr>
</tbody>
</table>

**Tabela 22:** Rezultati testiranja implementiranih klasifikatora nad enron2 korpusom
6.6 Enron spam korpus

Slika 24: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron2 korpusom.

Slika 25: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron2 korpusom.
6.6 Enron spam korpus

6.6.3 enron3

Legitima posta
--------
- Vlasnik: kitchen-l
- Ukupan broj: 4012 mejlova
- Datum prvog mejla: 2001-02-07
- Datum poslednjeg mejla: 2002-02-06
- Brisanje slicnih: Ne
- Enkodiranje: Ne

Spam
--------
- Vlasnik: BG
- Ukupan broj: 1500 mejlova
- Datum prvog mejla: 2004-08-01
- Datum poslednjeg mejla: 2005-07-31
- Brisanje slicnih: Da
- Enkodiranje: Ne

Spam:Legitmine poruke odnos ≈ 1 : 3

Ukupan broj mejlova (legitimni + spam): 5512

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Leživo pozitivni</th>
<th>Vreme treniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stopa greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>f1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator (λ = 5)</td>
<td>45</td>
<td>11.1431 s</td>
<td>0.1793 s</td>
<td>0.7818</td>
<td>0.2182</td>
<td>0.8866</td>
<td>0.8243</td>
<td>0.8544</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 50)</td>
<td>40</td>
<td>10.7045 s</td>
<td>0.1806 s</td>
<td>0.7909</td>
<td>0.2091</td>
<td>0.8992</td>
<td>0.8263</td>
<td>0.8613</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 500)</td>
<td>36</td>
<td>10.6999 s</td>
<td>0.1805 s</td>
<td>0.7963</td>
<td>0.2036</td>
<td>0.9093</td>
<td>0.8261</td>
<td>0.8657</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 999)</td>
<td>35</td>
<td>10.599 s</td>
<td>0.1805 s</td>
<td>0.7945</td>
<td>0.2054</td>
<td>0.9118</td>
<td>0.8227</td>
<td>0.865</td>
</tr>
<tr>
<td>kNN (k = 101, l = 75)</td>
<td>0</td>
<td>0.0193 s</td>
<td>7.8384 s</td>
<td>0.7364</td>
<td>0.2636</td>
<td>1</td>
<td>0.7324</td>
<td>0.8456</td>
</tr>
<tr>
<td>kNN (k = 101, l = 50)</td>
<td>0</td>
<td>0.0201 s</td>
<td>7.8384 s</td>
<td>0.7364</td>
<td>0.2636</td>
<td>1</td>
<td>0.7324</td>
<td>0.8456</td>
</tr>
<tr>
<td>kNN (k = 301, l = 200)</td>
<td>0</td>
<td>0.0203 s</td>
<td>7.8384 s</td>
<td>0.7364</td>
<td>0.2636</td>
<td>1</td>
<td>0.7324</td>
<td>0.8456</td>
</tr>
<tr>
<td>kNN (k = 201, l = 60)</td>
<td>0</td>
<td>0.01949 s</td>
<td>7.8384 s</td>
<td>0.7364</td>
<td>0.2636</td>
<td>1</td>
<td>0.7324</td>
<td>0.8456</td>
</tr>
<tr>
<td>Metod podržavajučih vektorja bez meke margine</td>
<td>17</td>
<td>19</td>
<td>1.7605 s</td>
<td>0.1861 s</td>
<td>0.9345</td>
<td>0.0654</td>
<td>0.9572</td>
<td>0.9524</td>
</tr>
<tr>
<td>Metod podržavajučih vektorja sa mekom marginom</td>
<td>0</td>
<td>40</td>
<td>0.5046 s</td>
<td>0.3903 s</td>
<td>0.9272</td>
<td>0.0727</td>
<td>1</td>
<td>0.9085</td>
</tr>
<tr>
<td>Perceptron</td>
<td>4</td>
<td>7</td>
<td>0.1505 s</td>
<td>0.0007 s</td>
<td>0.98</td>
<td>0.02</td>
<td>0.9899</td>
<td>0.9825</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>44</td>
<td>22</td>
<td>11.2177 s</td>
<td>0.5693 s</td>
<td>0.88</td>
<td>0.12</td>
<td>0.8892</td>
<td>0.9413</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>1</td>
<td>26</td>
<td>11.2792 s</td>
<td>0.5737 s</td>
<td>0.951</td>
<td>0.049</td>
<td>0.9975</td>
<td>0.9384</td>
</tr>
</tbody>
</table>

Tabela 23: Rezultati testiranja implementiranih klasifikatorov nad enron3 korpusom
Slika 26: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron3 korpusom

Slika 27: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron3 korpusom
6.6 Enron spam korpus

6.6.4 enron4

Legitimna posta
----------
- Vlasnik: williams-w3
- Ukupan broj: 1500 mejlova
- Datum prvog mejla: 2001-04-02
- Datum poslednjeg mejla: 2002-02-07
- Brisanje slicnih: Ne
- Enkodiranje: Ne

Spam
----------
- Vlasnik: GP
- Ukupan broj: 4500 mejlova
- Datum prvog mejla: 2003-12-18
- Datum poslednjeg mejla: 2005-09-06
- Brisanje slicnih: Ne
- Enkodiranje: Ne

Spam:Legitilme poruke odnos = 3:1

Ukupan broj mejlova (legitimni + spam): 6000

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Ložno pozitivni</th>
<th>Ložno negativni</th>
<th>Vreme u treniranju</th>
<th>Težnja</th>
<th>Slope greške</th>
<th>Preciznost</th>
<th>Odživ</th>
<th>( f ) mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator (( \lambda = 5 ))</td>
<td>34</td>
<td>50</td>
<td>10.9389 s</td>
<td>0.1904 s</td>
<td>0.86</td>
<td>0.14</td>
<td>0.7638</td>
<td>0.6875</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (( \lambda = 50 ))</td>
<td>33</td>
<td>52</td>
<td>10.7158 s</td>
<td>0.1901 s</td>
<td>0.8583</td>
<td>0.1417</td>
<td>0.7708</td>
<td>0.6811</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (( \lambda = 500 ))</td>
<td>31</td>
<td>56</td>
<td>10.8306 s</td>
<td>0.2174 s</td>
<td>0.855</td>
<td>0.145</td>
<td>0.7847</td>
<td>0.6686</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (( \lambda = 999 ))</td>
<td>31</td>
<td>56</td>
<td>10.6929 s</td>
<td>0.1911 s</td>
<td>0.855</td>
<td>0.145</td>
<td>0.7847</td>
<td>0.6686</td>
</tr>
<tr>
<td>kNN (( k = 201, l = 100 ))</td>
<td>130</td>
<td>4</td>
<td>0.0174 s</td>
<td>6.6853 s</td>
<td>0.7766</td>
<td>0.2233</td>
<td>0.0972</td>
<td>0.7777</td>
</tr>
<tr>
<td>kNN (( k = 301, l = 200 ))</td>
<td>115</td>
<td>15</td>
<td>0.0142 s</td>
<td>6.6884 s</td>
<td>0.7833</td>
<td>0.2167</td>
<td>0.2014</td>
<td>0.6591</td>
</tr>
<tr>
<td>kNN (( k = 301, l = 240 ))</td>
<td>80</td>
<td>58</td>
<td>0.0148 s</td>
<td>6.6733 s</td>
<td>0.77</td>
<td>0.23</td>
<td>0.993</td>
<td>0.2531</td>
</tr>
<tr>
<td>kNN (( k = 301, l = 295 ))</td>
<td>1</td>
<td>422</td>
<td>0.0142 s</td>
<td>6.6506 s</td>
<td>0.295</td>
<td>0.705</td>
<td>0.444</td>
<td>0.5246</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margini</td>
<td>29</td>
<td>3</td>
<td>0.9095 s</td>
<td>0.1769 s</td>
<td>0.9467</td>
<td>0.0533</td>
<td>0.7986</td>
<td>0.9745</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>31</td>
<td>1</td>
<td>0.8711 s</td>
<td>0.3394 s</td>
<td>0.9467</td>
<td>0.0533</td>
<td>0.7847</td>
<td>0.9912</td>
</tr>
<tr>
<td>Perceptron</td>
<td>22</td>
<td>2</td>
<td>0.1259 s</td>
<td>0.0007 s</td>
<td>0.96</td>
<td>0.04</td>
<td>0.8472</td>
<td>0.9839</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>60</td>
<td>0</td>
<td>11.5695 s</td>
<td>0.5355 s</td>
<td>0.9</td>
<td>0.1</td>
<td>0.5833</td>
<td>1</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>22</td>
<td>1</td>
<td>11.6689 s</td>
<td>0.5311 s</td>
<td>0.9617</td>
<td>0.0383</td>
<td>0.8472</td>
<td>0.9918</td>
</tr>
</tbody>
</table>

Tabela 24: Rezultati testiranja implementiranih klasifikatora nad enron4 korpusom
Slika 28: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron4 korpusom

Slika 29: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron4 korpusom
6.6 Enron spam korpus

6.6.5 enron5

Legitimna posta
--------
- Vlasnik: beck-s
- Ukupan broj: 1500 mejlova
- Datum prvog mejla: 2000-01-17
- Datum poslednjeg mejla: 2001-05-24
- Brisanje slicnih: Ne
- Enkodiranje: Ne

Spam
--------
- Vlasnik: SpamAssassin + HoneyPot
- Ukupan broj: 3675 mejlova
- Datum prvog mejla: 2001-05-25
- Datum poslednjeg mejla: 2005-07-22
- Brisanje slicnih: Da
- Enkodiranje: Ne

Spam : Legitimne poruke odnos \(\approx 3 : 1\)

Ukupan broj mejlova (legitimni + spam): 5175

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Loši pozitivni</th>
<th>Loši negativni</th>
<th>Vreme treiniranja</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stope greške</th>
<th>Preciznost</th>
<th>Odziv</th>
<th>(f_1) mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator ((\lambda = 0))</td>
<td>6</td>
<td>23</td>
<td>8.975 s</td>
<td>0.1716 s</td>
<td>0.942</td>
<td>0.058</td>
<td>0.9589</td>
<td>0.8589</td>
<td>0.9061</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ((\lambda = 50))</td>
<td>3</td>
<td>31</td>
<td>9.091 s</td>
<td>0.1768 s</td>
<td>0.932</td>
<td>0.068</td>
<td>0.9794</td>
<td>0.8218</td>
<td>0.8937</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ((\lambda = 500))</td>
<td>3</td>
<td>40</td>
<td>8.9127 s</td>
<td>0.1758 s</td>
<td>0.914</td>
<td>0.086</td>
<td>0.9794</td>
<td>0.7814</td>
<td>0.8693</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator ((\lambda = 999))</td>
<td>3</td>
<td>40</td>
<td>8.9127 s</td>
<td>0.1758 s</td>
<td>0.914</td>
<td>0.086</td>
<td>0.9794</td>
<td>0.7814</td>
<td>0.8693</td>
</tr>
<tr>
<td>(kNN (k = 201, l = 100))</td>
<td>18</td>
<td>57</td>
<td>0.0142 s</td>
<td>5.679 s</td>
<td>0.85</td>
<td>0.15</td>
<td>0.8767</td>
<td>0.6919</td>
<td>0.7734</td>
</tr>
<tr>
<td>(kNN (k = 201, l = 120))</td>
<td>1</td>
<td>145</td>
<td>0.0144 s</td>
<td>5.6944 s</td>
<td>0.708</td>
<td>0.292</td>
<td>0.9931</td>
<td>0.5</td>
<td>0.6651</td>
</tr>
<tr>
<td>(kNN (k = 301, l = 165))</td>
<td>11</td>
<td>59</td>
<td>0.0143 s</td>
<td>5.6711 s</td>
<td>0.86</td>
<td>0.14</td>
<td>0.9246</td>
<td>0.6959</td>
<td>0.7941</td>
</tr>
<tr>
<td>(kNN (k = 301, l = 180))</td>
<td>1</td>
<td>127</td>
<td>0.0142 s</td>
<td>5.6558 s</td>
<td>0.774</td>
<td>0.256</td>
<td>0.9931</td>
<td>0.5331</td>
<td>0.6938</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margine</td>
<td>10</td>
<td>12</td>
<td>0.5515 s</td>
<td>0.1221 s</td>
<td>0.956</td>
<td>0.044</td>
<td>0.9315</td>
<td>0.9189</td>
<td>0.9252</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>3</td>
<td>22</td>
<td>0.3831 s</td>
<td>0.3383 s</td>
<td>0.95</td>
<td>0.05</td>
<td>0.9794</td>
<td>0.8667</td>
<td>0.9196</td>
</tr>
<tr>
<td>Perceptron</td>
<td>6</td>
<td>7</td>
<td>0.0712 s</td>
<td>0.0007 s</td>
<td>0.974</td>
<td>0.026</td>
<td>0.9589</td>
<td>0.9524</td>
<td>0.9556</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>9</td>
<td>6</td>
<td>9.059 s</td>
<td>0.5068 s</td>
<td>0.97</td>
<td>0.03</td>
<td>0.9383</td>
<td>0.958</td>
<td>0.9481</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>4</td>
<td>10</td>
<td>9.0774 s</td>
<td>0.5088 s</td>
<td>0.972</td>
<td>0.028</td>
<td>0.9726</td>
<td>0.9342</td>
<td>0.953</td>
</tr>
</tbody>
</table>

Tabela 25: Rezultati testiranja implementiranih klasifikatora nad enron5 korpusom
6.6 Enron spam korpus 6 EKSPERIMENTALNI REZULTATI

Slika 30: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron5 korpusom

Slika 31: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron5 korpusom
6.6 Enron spam korpus

6.6.6 enron6

Legitima posta
---------
- Vlasnik: lokay-m
- Ukupan broj: 1500 mejlova
- Datum prvog mejla: 2000-06-06
- Datum poslednjeg mejla: 2002-03-25
- Brisanje slicnih: Ne
- Enkodiranje: Ne

Spam
---------
- Vlasnik: BG
- Ukupan broj: 4500 mejlova
- Datum prvog mejla: 2004-08-01
- Datum poslednjeg mejla: 2005-07-31
- Brisanje slicnih: Da
- Enkodiranje: Ne

Spam:Legitime poruke odnos = 3:1
Ukupan broj mejlova (legitimni + spam): 6000

<table>
<thead>
<tr>
<th>Algoritam</th>
<th>Leža pozitimi</th>
<th>Leža negativni</th>
<th>Vreme trening a</th>
<th>Vreme validacije</th>
<th>Tačnost</th>
<th>Stepne greške</th>
<th>Preciznost</th>
<th>Odštev</th>
<th>F1 mera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naivni Bajesov klasifikator (λ = 5)</td>
<td>11</td>
<td>51</td>
<td>12.5867 s</td>
<td>0.1949 s</td>
<td>0.8967</td>
<td>0.1033</td>
<td>0.9281</td>
<td>0.9357</td>
<td>0.821</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 50)</td>
<td>9</td>
<td>58</td>
<td>12.713 s</td>
<td>0.1961 s</td>
<td>0.8883</td>
<td>0.1116</td>
<td>0.9412</td>
<td>0.7128</td>
<td>0.8113</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 500)</td>
<td>8</td>
<td>72</td>
<td>12.6436 s</td>
<td>0.1951 s</td>
<td>0.8667</td>
<td>0.1333</td>
<td>0.9477</td>
<td>0.6682</td>
<td>0.7838</td>
</tr>
<tr>
<td>Naivni Bajesov klasifikator (λ = 999)</td>
<td>8</td>
<td>73</td>
<td>12.6462 s</td>
<td>0.1952 s</td>
<td>0.865</td>
<td>0.135</td>
<td>0.9477</td>
<td>0.6651</td>
<td>0.7816</td>
</tr>
<tr>
<td>kNN (k = 201, l = 120)</td>
<td>13</td>
<td>146</td>
<td>0.0205 s</td>
<td>8.1379 s</td>
<td>0.735</td>
<td>0.265</td>
<td>0.915</td>
<td>0.4895</td>
<td>0.6378</td>
</tr>
<tr>
<td>kNN (k = 301, l = 150)</td>
<td>96</td>
<td>0</td>
<td>0.0175 s</td>
<td>8.1801 s</td>
<td>0.84</td>
<td>0.16</td>
<td>0.3725</td>
<td>1</td>
<td>0.5428</td>
</tr>
<tr>
<td>kNN (k = 301, l = 175)</td>
<td>17</td>
<td>101</td>
<td>0.0184 s</td>
<td>8.1445 s</td>
<td>0.8033</td>
<td>0.1967</td>
<td>0.8889</td>
<td>0.5738</td>
<td>0.6974</td>
</tr>
<tr>
<td>kNN (k = 301, l = 200)</td>
<td>1</td>
<td>353</td>
<td>0.0181 s</td>
<td>8.2025 s</td>
<td>0.41</td>
<td>0.59</td>
<td>0.9934</td>
<td>0.3009</td>
<td>0.462</td>
</tr>
<tr>
<td>Metod podržavajućih vektora bez meke margin</td>
<td>15</td>
<td>22</td>
<td>1.4252 s</td>
<td>0.2117 s</td>
<td>0.9383</td>
<td>0.0616</td>
<td>0.902</td>
<td>0.8625</td>
<td>0.8818</td>
</tr>
<tr>
<td>Metod podržavajućih vektora sa mekom marginom</td>
<td>14</td>
<td>8</td>
<td>0.4801 s</td>
<td>0.4916 s</td>
<td>0.9633</td>
<td>0.0367</td>
<td>0.9085</td>
<td>0.9456</td>
<td>0.9267</td>
</tr>
<tr>
<td>Perceptron</td>
<td>11</td>
<td>11</td>
<td>0.1157 s</td>
<td>0.0008 s</td>
<td>0.9633</td>
<td>0.0367</td>
<td>0.9281</td>
<td>0.9281</td>
<td>0.9281</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator</td>
<td>19</td>
<td>2</td>
<td>13.078 s</td>
<td>0.686 s</td>
<td>0.965</td>
<td>0.035</td>
<td>0.8758</td>
<td>0.9853</td>
<td>0.9273</td>
</tr>
<tr>
<td>SVM/Naivni Bajesov klasifikator/Perceptron</td>
<td>9</td>
<td>6</td>
<td>13.8014 s</td>
<td>1.0644 s</td>
<td>0.975</td>
<td>0.025</td>
<td>0.9608</td>
<td>0.9423</td>
<td>0.9515</td>
</tr>
</tbody>
</table>

Tabela 26: Rezultati testiranja implementiranih klasifikatora nad enron6 korpusom

Napomena: Perceptron nije konvergirao pa je neophodno povećati interval povratnih vrednosti funkcije string_hash sa 31973 na 83339 i vrednost parametara odlike-do sa 30000 na 80000.
Slika 32: Dijagram ispravno i neispravno klasifikovanih instanci svih implementiranih algoritama nad enron6 korpusom

Slika 33: Dijagram vremena treniranja i validacije svih implementiranih algoritama nad enron6 korpusom
7 | Zaključak

Filtriranje neželjene elektronske pošte postaje važan aspekt modernih e-mail sistema zbog velikog povećanja obima spama u poslednjih nekoliko godina. Mašinsko učenje privlači pažnju mnogih istraživača na ovom polju kao moćna računarska metodologija koja može da pomogne u ublažavanju takvih problema. U ovom radu su primenjeni i evaluirani neki algoritmi mašinskog učenja nad različitim korpusima elektronske pošte u cilju pronalaženja optimalnog algoritma za rešavanje problema spama. Neophodni parametri su određivani eksperimentalno sa velikim brojem kombinacija, tako da su njihove vrednosti verovatno dosta blizu idealnih.

U ovom poglavlju će biti navedeni zaključci o eksperimentalnim rezultatima prethodnog poglavlja. Iz dobijenih rezultata izvedeni su sledeći zaključci:

* Najmanje vremena za treniranje troši algoritam $k$ najближих suseda, dok najmanje vremena za validaciju troši klasifikator zasnovan na perceptronu.

* Najviše vremena za treniranje troši Naivni Bajesov klasifikator, dok najviše vremena za validaciju troši algoritam $k$ najближих suseda.

* Kako se određivanje vrednosti parametara $k$ i $l$ za algoritam $k$ najближих suseda vrši empirijski, a sama tačnost klasifikacije se pokazala kao najlošijom od svih primenjenih algoritama dolazi se do zaključka da algoritam $k$ najближих suseda ne treba koristiti kod problema filtriranja neželjene elektronske pošte.

* Kod Naivnog Bajesovog klasifikatora se pokazalo da uvećavanjem parametra $\lambda$ koji ukazuje na rizik kada klasifikujemo legitimnu poruku kao neželjenu smanjujemo broj lažno pozitivnih instanci ali i povećavamo broj lažno negativnih sa gotovo istim stepenom tačnosti. Kako je fundamentalna potreba bilo kog filtera neželjene elektronske pošte da nikada ne označi dobru poruku kao neželjenu, odnosno da broj lažno pozitivnih bude što manji u interesu nam je da parametar $\lambda$ bude što veći.

* Najveću tačnost klasifikacije sa najnižim brojem lažno pozitivnih instanci pokazali su metod podržavajućih vektora i klasifikator zasnovan na perceptronu. Oba algoritma troše jako malo vremena na treniranje i validaciju u poređenju sa ostalim implementiranim algoritmicima. Iz svega navedenog se zaključuje da su metod podržavajućih vektora i klasifikator zasnovan na perceptronu kao i njihovo kombinovanje najbolji izbor za rešavanje problema spama.

Na kraju se izvodi zaključak da je oblast suzbijanja spama danas zrela i dosta dobro razvijena. Ali zašto je onda naš inbox i dalje često pun spama? To je verovatno zato što algoritmi koji se koriste nisu savršeni a sami spameri koriste razne trikove kojima zaobilaze metode filtriranja.
Bibliografija


A | Dodatak

U nastavku se nalazi kod programa koji je korišćen u implementaciji i evaluaciji algoritama mašinskog učenja za filtriranje neželjene elektronske pošte.

A.1 | FileUtil.h

```c++
/*
 * FileUtil.h - Fajl sistem operacije
 *
*/
#ifndef __FileUtil_h__
#define __FileUtil_h__
#include <iostream>
#include <fstream>
#include <iomanip>
#include <sstream>
#include <string>
#include <vector>
#include <iterator>
#include <algorithm>
#include <sys/types.h>
#include <dirent.h>

#include "Izuzetak.h"
using namespace std;

#ifndef DT_REG
#define DT_REG 8
#define NOT_LINUX
#include <sys/stat.h>
#endif

#ifndef DT_DIR
#define DT_DIR 4
#endif

vector<string> citajDirektorijume(const string& dir, unsigned short int type = DT_REG)
{
    vector<string> rezultat;
    DIR* pDir = opendir(dir.c_str());
    if (pDir)
    {
        dirent* pEntry;
        while ( (pEntry = readdir(pDir)) )
        {
            if (pDir)
            {
                struct stat info;
                if (stat((dir + '/' + string(pEntry->d_name)).c_str(), &info) != 0)
                    throw Izuzetak("Nedostupni atributi fajla " + string(pEntry->d_name));
                if ((type & DT_REG && S_ISREG(info.st_mode)) || (type & DT_DIR && S_ISDIR(info.st_mode)))
                    rezultat.push_back(string(pEntry->d_name));
            }
            else
                throw Izuzetak("Nije moguce procitati direktorijum " + dir);
            return rezultat;
        }
    }
    closedir(pDir);
    return rezultat;
}
```
// Citanje fajla kao string. Baca izuzetak na neuspeh.
string citajFajl(const string& fileName)
{
    ifstream ulazniFajl(fileName.c_str());
    if (!ulazniFajl) throw Izuzetak("Nije moguce procitati fajl " + fileName);

    // citanje fajla karakter po karakter
    ostringstream os;
    ulazniFajl >> noskipws;
    copy(istream_iterator<char>(ulazniFajl), istream_iterator<char>(), ostream_iterator<char>(os));
    ulazniFajl.close();
    return os.str();
}
#endif
// Vraca vrednost atributa sa datim indeksom
// Nije dobro koristiti attributes[index] jer ce to ubaciti vrednosti u mapu
inline bool getAttribute(int index) const
{
    if (attributes.find(index) == attributes.end())
        return false;
    else
        return (attributes.find(index)->second);
}

inline void setAttribute(int index, bool value)
{
    if (index > attributeCount || index < 1)
        throw Izuzetak("Neispravan indeks atributa");
    if (value)
        attributes[index] = true;
    else
        if (attributes.find(index) != attributes.end())
            attributes.erase(attributes.find(index));
}

// Ispis atributa na izlaz
friend ostream& operator<<(ostream& out, const Attributes::value_type& attr)
{
    return out << attr.first << ":" << attr.second;
}

// Ispis instanci na izlaz
friend ostream& operator<<(ostream& out, const Instance& inst)
{
    out << "Instanca(";
    int gotovo = 0;
    for (Attributes::const_iterator i = inst.attributes.begin(); i != inst.attributes.end(); i++)
    {
        if (gotovo++ != 0) out << ", ";
        out << &i;
    }
    out << "; " << inst.kategorija << ");";
    return out;
}

/**
 * Predstavlja skup instanci.
 * U sustini je apstrakcija vectora instanci.
 * Skup podataka takođe mora proveriti da sve instance imaju
 * isti broj atributa što nisam implementirao.
 * Druga vazna razlika je da vector instanci obično rukuje
 * "po vrednosti" i kopira napred nazad,
 * a koristi se kao "normalna klasa" pozivanjem.
 */

class Dataset
{
public:
    typedef vector<Instance> Instances;
    Instances instances;

    Dataset(): instances() {}
    Dataset(Instances instances): instances(instances) {}

    void addInstance(Instance instance)
    {
        instances.push_back(instance);
    }

    void addInstances(vector<Instance> newInstances)
    {
        instances.insert(instances.end(), newInstances.begin(), newInstances.end());
    }

#endif
A.3 | IzdvajanjeOdlika.h

#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <algorithm>
#include <cctype>
#include "Instance.h"
#include "FileUtil.h"
#include "Izuzetak.h"

using namespace std;

int string_hash(char *key, int len)
{
    int hash, i;
    for(hash = i = 0; i < len; ++i)
    {
        hash += key[i];
        hash += (hash << 10);
        hash ^= (hash >> 6);
    }
    hash += (hash << 3);
    hash ^= (hash >> 11);
    hash += (hash << 15);
    return hash % 31973; // 83339
}

class IzdvajanjeOdlikaKorpusa: public IzdvajanjeOdlika
{
    public:
        // Konstruktor
        // pocetnaRec - rec koja odgovara prvom atributu
        // krajnjaRec - rec koja odgovara poslednjem atributu
        virtual Instance kreirajInstancu(const string &s) const = 0;
    };

    virtual ~IzdvajanjeOdlikaKorpusa(){}
// citaj brojeve iz stringa,
// za one koji su manji od brAtributa podesi odgovarajuce atribute u instanci
int n;
string rec; // koristiti za ne PU korpuse
while (in)
{
    // citamo rec po rec
    n = string_hash((char*) rec.c_str(), rec.length()); //procitane reci kodiram u brojeve
    if (n >= pocetnaRec && n <= krajnjaRec)
        rezultat.setAttribute(n - pocetnaRec + 1, true);
}
return rezultat;
};

class CitanjeInstanciKorpusa
{
public:
    string rootDirektorijum;
    const IzdvajanjeOdlika* odlike;

    CitanjeInstanciKorpusa(const string& rootDirektorijum, const IzdvajanjeOdlika* odlike)
    {
        if (rootDirektorijum[rootDirektorijum.length() - 1] != '/')
            this->rootDirektorijum = rootDirektorijum + '/';
        else
            this->rootDirektorijum = rootDirektorijum;
        this->odlike = odlike;
    }

    Instance citanjeInstanciIzFajla(const string& filename) const
    {
        string file = rootDirektorijum + filename;
        string s = citajFajl(file);

        if (!odlike)
            throw Izuzetak("Izdvajac odlika nije naveden!");
        Instance inst = odlike->kreirajInstancu(s);

        if (filename.find("legit", 0) != string::npos)
            inst.kategorija = LEGITIMNO;
        else if (filename.find("spam", 0) != string::npos)
            inst.kategorija = SPAM;
        return inst;
    }

    vector<Instance> citanjeInstanciIzDirektorijuma(const string& dir) const
    {
        vector<Instance> rezultat;
        vector<Instance> fajlovi = citajDirektorijume(rootDirektorijum + dir + '/');

        for (vector<Instance>::iterator i = fajlovi.begin(); i != fajlovi.end(); i++)
            rezultat.push_back(citanjeInstanciIzFajla(dir + /* + */));
        return rezultat;
    }

};
// Kreiranje trening skupa iz korpusa
// n je vektor od 1 do 10 - deo koji se izostavlja iz treniranja
Dataset* kreiranjeTreningSkupaOdKorpusa(const vector<int>& n)
{
    // dodavamo direktorijume koji se ne nalaze u datom vektoru
    vector<int> dodavanjeDirektorijuma;
    for (int i = 1; i <= 10; i++)
    {
        if (find(n.begin(), n.end(), i) == n.end())
            dodavanjeDirektorijuma.push_back(i);
    }
    return kreiranjeValidacionogSkupaOdKorpusa(dodavanjeDirektorijuma);
}

// Kreiranje validacionog skupa iz korpusa
// n je vektor od 1 do 10 - delovi koji se uzimaju za validaciju
Dataset* kreiranjeValidacionogSkupaOdKorpusa(const vector<int>& n)
{
    Dataset* rezultat = new Dataset();
    for (vector<int>::const_iterator i = n.begin(); i != n.end(); i++)
    {
        if (*i > 10 || *i < 1)
            throw Izuzetak("Ocekuje se n od 1 do 10!");
        ostringstream os;
        os << "part" << *i;
        rezultat->addInstances(citanjeInstanciIzDirektorijuma(os.str()));
        cout << " Dodane instance iz foldera " << os.str() << ", velicina skupa je sada: "
            << rezultat->instances.size() << " instanci" << endl;
    }
    return rezultat;
};
#endif
/* Klasifikator.h - osnovna klasa za klasifikator */

#ifndef __Klasifikator_h__
#define __Klasifikator_h__

#include "Instance.h"
#include <vector>
#include <ctime>
using namespace std;

/* Osnovna klasa za klasifikator. Navodi samo interfejs. */
class Klasifikator {
public:
  virtual string getName() = 0;
  virtual void treniraj(const Dataset* data) = 0;
  virtual MejlKlasa klasifikuj(const Instance& inst) const = 0;
};

/* Trivijalni klasifikator koji klasifikuje sve kao jednu istu klasu */
class TrivijalniKlasifikator: public Klasifikator {
  private:
    MejlKlasa kategorija;
  public:
    TrivijalniKlasifikator(MejlKlasa kategorija): kategorija(kategorija) {}
    virtual ~TrivijalniKlasifikator(){};

    string getName() {
      return "Trivijalni klasifikator";
    }

    virtual void treniraj(const Dataset* data) {
    }

    virtual MejlKlasa klasifikuj(const Instance& inst) const {
      return kategorija;
    };

  /* Klasa za procenu performansi klasifikatora */
class ProcenaPerformansiKlasifikatora {
  public:
    static void testirajKlasifikator(Klasifikator* c, const Dataset* treningSkup, const Dataset* validacioniSkup) {
      cout << "Trening klasifikator: " << c->getName() << endl;
      clock_t trenirajStart = clock();
      c->treniraj(treningSkup);
    }
};
clock_t trenirajKraj = clock();
cout << "Testiranje performansi klasifikatora" << endl;
int fp = 0;
int fn = 0;
int tp = 0;
int tn = 0;
int velicinaValidacionogSkupa = validacioniSkup->instances.size();
clock_t ispravanStart = clock();
for (vector<Instance>::const_iterator i = validacioniSkup->instances.begin();
i != validacioniSkup->instances.end(); i++)
{
    MejklKlasa stvarna_klasa = i->kategorija;
    MejklKlasa predvidjena_klasa = c->klasifikuj(*i);
    if (stvarna_klasa != predvidjena_klasa)
    {
        if (stvarna_klasa == LEGITIMNO)
        {
            fp++;
        }
        else
        {
            fn++;
        }
    }
    else if (stvarna_klasa == predvidjena_klasa)
    {
        if (stvarna_klasa == LEGITIMNO)
        {
            tp++;
        }
        else
        {
            tn++;
        }
    }
}
clock_t ispravanKraj = clock();
double preciznost = ((double)(tp))/(tp+fp);
double odziv = ((double)(tp))/(tp+fn);
cout << "Matrica konfuzije:" << endl;
cout << "-----------" << endl;
cout << "| " << tp << " | " << fn << " |" << endl;
cout << "-----------" << endl;
cout << "| " << fp << " | " << tn << " |" << endl;
cout << "-----------" << endl;

cout << "Rezultati (" << c->getName() << "): " << endl;
cout << "Velicina validacionog skupa: " << velicinaValidacionogSkupa << endl;
cout << "Lazno pozitivni (LEGITIMNO klasifikovano kao SPAM): " << fp << endl;
cout << "Lazno negativni (SPAM klasifikovano kao LEGITIMNO): " << fn << endl;
cout << "Vreme treniranja: " << ((double)(trenirajKraj - trenirajStart))/CLOCKS_PER_SEC << endl;
cout << "Vreme validacije: " << ((double)(ispravanKraj - ispravanStart))/CLOCKS_PER_SEC << endl;
cout << "Preciznost: " << ((double)(velicinaValidacionogSkupa - (fp + fn)))/velicinaValidacionogSkupa << endl;
cout << "Stopa greske: " << ((double)(fp + fn))/velicinaValidacionogSkupa << endl;
cout << "f1 mera: " << (double) (2 * preciznost * odziv) / (preciznost + odziv) << endl;
"
A.6 Klasifikator1od2.h

```cpp
/*
 * Klasifikator1od2.h - klasifikator koji jednostavno kombinuje odluke dva klasifikatora
 */

#ifndef __KLASIFIKATOR1OD2_H__
#define __KLASIFIKATOR1OD2_H__

#include "Instance.h"
#include "Klasifikator.h"
#include "Izuzetak.h"
#include <vector>
using namespace std;

/**
 * 1od2 klasifikator uzima rezultate dva klasifikatora i "jake klase".
 * Instanca je klasifikovana u klasu C ako je C jaka klasa i najmanje jedan klasifikator
 * klasifikuje instancu u klasu C ili C nije jaka klasa i oba klasifikatora
 * klasifikuju instancu u klasu C.
 */

class Klasifikator1od2: public Klasifikator
{
private:
    Klasifikator* c1;
    Klasifikator* c2;
    MejlKlasa jakaKlasa;
public:
    Klasifikator1od2(Klasifikator* c1, Klasifikator* c2, MejlKlasa jakaKlasa):
        c1(c1), c2(c2), jakaKlasa(jakaKlasa) {}

    virtual "Klasifikator1od2()";

    string getName() const
    {
        return "Klasifikator 1 od 2";
    }

    void treniraj(const Dataset* data)
    {
        if (c1 == NULL || c2 == NULL)
            throw Izuzetak("Neinicijalizovan klasifikator!");
        c1->treniraj(data);
        c2->treniraj(data);
    }

    MejKlasa klasifikuj(const Instance& inst) const
    {
        MejKlasa d1 = c1->klasifikuj(inst);
        MejKlasa d2 = c2->klasifikuj(inst);
        if (d1 == jakaKlasa || d2 == jakaKlasa)
            return jakaKlasa;
        return (d1 == d2 ? d1 : NEDEFINISANO);
    }
};
#endif
```

73
#ifndef __Klasifikator2od3_h__
#define __Klasifikator2od3_h__

#include "Instance.h"
#include "Klasifikator.h"
#include "Izuzetak.h"
#include <vector>
using namespace std;

class Klasifikator2od3: public Klasifikator
{
private:
    Klasifikator* c1;
    Klasifikator* c2;
    Klasifikator* c3;
public:
    Klasifikator2od3(Klasifikator* c1, Klasifikator* c2, Klasifikator* c3): c1(c1), c2(c2), c3(c3) {}
    virtual ~Klasifikator2od3(){};

    string getName() const{
        return "Klasifikator 2 od 3";
    }

    void treniraj(const Dataset* data)
    {
        if (c1 == NULL || c2 == NULL || c3 == NULL)
            throw Izuzetak("Neinicijalizovani klasifikator");
        c1->treniraj(data);
        c2->treniraj(data);
        c3->treniraj(data);
    }

    MejlKlasa klasifikuj(const Instance& inst) const
    {
        MejlKlasa d1 = c1->klasifikuj(inst);
        MejlKlasa d2 = c2->klasifikuj(inst);
        MejlKlasa d3 = c3->klasifikuj(inst);
        if (d1 == NEDEFINISANO || d2 == NEDEFINISANO || d3 == NEDEFINISANO)
            return NEDEFINISANO;
        int c = 0;
        if (d1 == SPAM)
            c++;
        if (d2 == SPAM)
            c++;
        if (d3 == SPAM)
            c++;
        return (c >= 2 ? SPAM : LEGITIMNO);
    }
};
#endif
A.8 KlasifikatorKNajblizihSuseda.h

```cpp
/*
 * KlasifikatorKNajblizihSuseda.h - definicija kNN klasifikatora
 */
#endif __KlasifikatorKNajblizihSuseda_h__
#define __KlasifikatorKNajblizihSuseda_h__
#include "Instance.h"
#include "Klasifikator.h"
#include "Izuzetak.h"
#include <vector>
using namespace std;
/**
 * Rastojanje izmedju instanci
 */
class Rastojanje
{
public:
    virtual double rastojanje(const Instance& a, const Instance& b) const = 0;
};
class ApsolutnoRastojanje: public Rastojanje
{
public:
    ApsolutnoRastojanje(){
    virtual "ApsolutnoRastojanje()"
    double rastojanje(const Instance& a, const Instance& b) const
    {
        double rezultat = 0;
        if (a.attributeCount != b.attributeCount)
            throw Izuzetak("Nemoguce je racunanje rastojanja izmedju vektora razlicitih dimenzija!");
        Instance::Attributes::const_iterator aAtribut = a.attributes.begin();
        Instance::Attributes::const_iterator bAtribut = b.attributes.begin();
        while (aAtribut != a.attributes.end() || bAtribut != b.attributes.end())
        {
            if (aAtribut == a.attributes.end())
                rezultat++;
                bAtribut++;
            else if (bAtribut == b.attributes.end())
                rezultat++;
                aAtribut++;
            else if (aAtribut->first < bAtribut->first)
                rezultat++;
                aAtribut++;
            else if (aAtribut->first > bAtribut->first)
                rezultat++;
                bAtribut++;
            else
                if (aAtribut->second != bAtribut->second) rezultat++;
                aAtribut++;
                bAtribut++;
            }
        return rezultat;
    }
};
```
class KlasifikatorKNajblizihSuseda : public Klasifikator
{
private:
    int k; // pozitivan i neparan
    int l;
    Dataset::Instances instances;
    Rastojanje* d;

    /* Binarne f-je za poredjenje dva para za primenu partial_sort algoritma */
    class poredjenjeParova
    {
        public:
            bool operator()(const pair<int, double> a, const pair<int, double> b)
            {
                return (a.second < b.second);
            }
    };

    public:
    /* d - rastojanje
    k - k najblizih suseda
    l - broj suseda koji moraju biti spam da bi klasifikovali poruku kao spam */
    KlasifikatorKNajblizihSuseda(Rastojanje* d, int k, int l)
    {
        if (k < 1 || l > k)
            throw Izuzetak("k mora biti pozitivno i vece od l!");
        this->k = k;
        this->d = d;
        this->l = l;
    }
    KlasifikatorKNajblizihSuseda(Rastojanje* d, int k)
    {
        if (k < 1 || k % 2 == 0)
            throw Izuzetak("k mora biti pozitivno i neparno!");
        this->k = k;
        this->d = d;
        this->l = (k - 1)/2 + 1;
    }

    virtual ~KlasifikatorKNajblizihSuseda() {};

    string getName() const
    {
        return "Klasifikator K Najblizih Suseda";
    };

    void treniraj(const Dataset* data) const
    {
        instances = data->instances;

        // provera da li ima nedefinisanih instanci
        for (Dataset::Instances::iterator i = instances.begin(); i != instances.end(); i++)
        {
            if (i->kategorija == NEDEFINISANO)
                throw Izuzetak("Nedefinisane instance nisu dozvoljene u trening skupu!");
        }

        // za svaku instancu racunamo rastojanje do instance koja se klasifikuje
        vector<pair<int, double>> v(instances.size());

        for (int i = 0; i < (int)instances.size(); i++)
        {
            v[i] = pair<int, double>(i, d->rastojanje(instances[i]));
        }

        partial_sort(v.begin(), v.begin() + k, v.end(), poredjenjeParova());
    }
}
A.9 NaivniBajesovKlasifikator.h

// gledamo koja klasa dominira
int brSpam = 0;
for (vector<pair<int, double> >::iterator i = v.begin(); i != v.begin() + k; i++)
    if (instances[i->first].kategorija == SPAM)
        brSpam++;

if (brSpam >= l)
    return SPAM;
else
    return LEGITIMNO;
}
#endif

A.9 NaivniBajesovKlasifikator.h

/*
 * NaivniBajesovKlasifikator.h - definicija naivnog Bajesovog klasifikatora
 */

#ifndef __NaivniBajesovKlasifikator_h__
#define __NaivniBajesovKlasifikator_h__

#include "Instance.h"
#include "Klasifikator.h"
#include "Izuzetak.h"
#include <vector>
using namespace std;

/**
 * Naivni Bajesov klasifikator
 */
class NaivniBajesovKlasifikator: public Klasifikator
{
    private:
        int brAtributa; // broj atributa u instanci
        struct doublePar
        {
            double value[2];
            double& operator[](int index)
            {
                return value[index];
            }
            double operator[](int index) const
            {
                return value[index];
            }
        };
        vector<doublePar> verovatnoca; // cuva verovatnoca za sve vrednosti svih atributa
        double lambda; // sklonost klasifikacije
        double pL_pS; // P(L) / P(S)
        static const double VELIKA_VREDNOST = 1000000; // koristim da ne bih kod racunanja
        static const double MALA_VREDNOST = 0.0000001; // dosao do beskonacnosti ili NaN vrednosti
        static const double VELIKA_VREDNOST = 1000000; // kao kod na primer deljenja nulom
    public:
        NaivniBajesovKlasifikator(double lambda): lambda(lambda){};
        virtual ~NaivniBajesovKlasifikator(){};
        string getName()
        {
            return "Naivni Bajesov klasifikator";
        }
};
void treniraj(const Dataset* data)
{
    // odbacivanje praznog skupa podataka
    int brInstanci = data->instances.size();
    if (brInstanci == 0)
        throw Izuzetak("Prazan skup podataka!");

    // racunanje P(L) i P(S)
    int brLegitimnih = 0;
    int brSpam = 0;

    Dataset::Instances::const_iterator i;
    for (i = data->instances.begin(); i != data->instances.end(); i++)
    {
        if (i->kategorija == SPAM)
            brSpam++;
        else if (i->kategorija == LEGITIMNO)
            brLegitimnih++;
        else
            throw Izuzetak("Neklasifikovana poruka u trening skupu!");
    }

    if (brLegitimnih == 0 || brSpam == 0)
        throw Izuzetak("Gde su se bar po jedna legitimna i spam poruka u trening skupu!");

    pL_pS = (double)brLegitimnih/(double)brSpam;

    // racunanje verovatnoce za sve atribute i sve vrednosti
    brAtributa = data->instances[0].attributeCount;
    if (brAtributa == 0)
        throw Izuzetak("Instance nemaju atribut!");

    verovatnoca = vector<doublePar>(brAtributa);
    for (int attrib = 0; attrib < brAtributa; attrib++)
    {
        int brTacnoKlasifikovanihSpamPoruka = 0;
        int brTacnoKlasifikovanihLegitimnihPoruka = 0;
        for (i = data->instances.begin(); i != data->instances.end(); i++)
        {
            if (i->getAttribute(attrib + 1) == true)
            {
                if (i->kategorija == SPAM)
                    brTacnoKlasifikovanihSpamPoruka++;
                else
                    brTacnoKlasifikovanihLegitimnihPoruka++;
            }
        }

        // verovatnoca P(x) = P(x | S)/P(x | L)
        if (brTacnoKlasifikovanihSpamPoruka == 0)
            verovatnoca[attrib][1] = MALA_VREDNOST;
        else if (brTacnoKlasifikovanihLegitimnihPoruka == 0)
            verovatnoca[attrib][1] = VELIKA_VREDNOST;
        else
            verovatnoca[attrib][1] = ((double)brTacnoKlasifikovanihSpamPoruka/
                (double)brTacnoKlasifikovanihLegitimnihPoruka)*pL_pS;

        int brPogresnoKlasifikovanihLegitimnihPoruka = brLegitimnih - brTacnoKlasifikovanihLegitimnihPoruka;
        int brPogresnoKlasifikovanihSpamPoruka = brSpam - brTacnoKlasifikovanihSpamPoruka;

        if (brPogresnoKlasifikovanihSpamPoruka == 0)
            verovatnoca[attrib][0] = MALA_VREDNOST;
        else if (brPogresnoKlasifikovanihLegitimnihPoruka == 0)
            verovatnoca[attrib][0] = VELIKA_VREDNOST;
        else
            verovatnoca[attrib][0] = ((double)brPogresnoKlasifikovanihSpamPoruka/
                (double)brPogresnoKlasifikovanihLegitimnihPoruka)*pL_pS;
    }
}
MejlKlasa klasifikuj(const Instance& inst) const
{
    // racunanje verovatnoca P(x|S)/P(x|L) za instancu. 
    // naivno pretpostavljamo da je to proizvod verovatnoca odnosa atributa 
    double ukupnaVerovatnoca = 1;
    for (int i = 0; i < brAtributa; i++)
    {
        ukupnaVerovatnoca = ukupnaVerovatnoca * verovatnoca[i][inst.getAttribute(i + 1)];
    }
    // poruka se klasifikuje kao spam ako je verovatnoca > lambda * P(L)/P(S)
    if (ukupnaVerovatnoca > lambda * pL_pS)
        return SPAM;
    else
        return LEGITIMNO;
}

// tražimo instance koje nisu klasifikovane korektno
// shodno tome azuriramo tezinski vektor
for (Dataset::Instances::const_iterator i = data->instances.begin();
i != data->instances.end(); i++)
{
    if (klasifikuj(*i) != i->kategorija)
    {
        konvergira = false;
        double c = i->kategorija == SPAM ? 1 : -1;
        b += c;
        for (Instance::Attributes::const_iterator attr = i->attributes.begin(); attr != i->attributes.end(); attr++)
        {
            int atributIndeks = attr->first;
            w[atributIndeks - 1] += c;
        }
    }
}
if (preostaleIzborite <= 0)
throw Izuzetak("Perceptron ne konvergira. Najverovatnije podaci nisu linearno razdvojeni!");

MejlKlasa klasifikuj(const Instance& inst) const
{
    double rezultat = 0;
    for (Instance::Attributes::const_iterator i = inst.attributes.begin();
i != inst.attributes.end(); i++)
    {
        int atributIndeks = i->first;
        rezultat += w[atributIndeks - 1];
    }
    if (rezultat + b > 0)
        return SPAM;
    else
        return LEGITIMNO;
};

*/
LEARN_PARM learnParm;
KERNEL_PARM kernelParm;
long kernelCacheSize;

// Model
MODEL model;

// Konvertuje instance u DOC strukturu koju koristi SVMlight
static void instanciaKonverzijaDOC(const Instance& inst, DOC& doc)
{
    // alocira memoriju za odgovarajuci broj zapisa
    doc.words = (WORD*) malloc(sizeof(WORD) * (inst.attributes.size() + 1));
    if (doc.words == NULL)
        throw Izuzetak("Nije moguce alocirati memoriju!");
    int curWordIndex = 0;
    for (Instance::Attributes::const_iterator i = inst.attributes.begin(); i != inst.attributes.end(); i++)
    {
        doc.words[curWordIndex].wnum = i->first;
        doc.words[curWordIndex].weight = (FVAL)i->second;
        curWordIndex++;
    }
    doc.words[curWordIndex].wnum = 0;
    doc.words[curWordIndex].weight = 0.0;
    doc.queryid = 0; // neophodno za rangiranje
    doc.costfactor = 1; // troskovi reklasifikacije za ovu instancu
    doc.docnum=-1; // pozicija dokumenta u nizu trening skupa
    doc.twonorm_qrepprod_ss(doc.words, doc.words); // neophodno za unutarne upotrebe za SVMLight-a
}

public:
// Kreiranje klasifikatora
SVMKlasifikator(bool softmargin = false)
{
    // Podesavanje podrazumevanih parametara ucenja
    // Konfigurisanje parametara ucenja
    // podrazumevane vrednosti su u svm_learn_main.c::read_input_parameters
    learnParm.type = CLASSIFICATION; /* tip ucenja je klasifikacija, a mogu biti jos i regresija, ranking ili optimizacija*/
    learnParm.biased_hyperplane = 1; /* koriscenje wx + b za razdvajajucu hiper-ravan */
    learnParm.remove_inconsistent = 0; /* ako je ukljuceno izostavlja primere sa alfa uz C
    i model se ponovo trenira */
    learnParm.skip_final_opt_check = 0; /* ne preskakanje Karush-Kuhn-Tucker uslova na kraju optimizacije
    za primere uklonjene smnjivajem */
    learnParm.svm_maxqpsize = 10; /* velicina radnog skupa q */
    learnParm.svm_nevarsize = 0; /* nove promenljive se unose u radni skup u novoj iteraciji */
    learnParm.svm_iter_to_shrink = -9999; /* iteracije b posle kojih primer moze da se ukloni smnjivajem */
    learnParm.svm_c = 0.0; /* gornja granica parametra C na alfa*/
    learnParm.svm_costratio = 1.0; /* faktor kojim se mnozi C za pozitivne primere */
    learnParm.svm_costratio_unlab = 1.0; /* faktor kojim se mnozi C za neobelezena primere */
    learnParm.svm_unlabbound = 1E-5; /* tolerisana greka za razstojanja koja se koristi 
    za kriterijum zaustavljanja*/
    learnParm.compute_loo = 0; /* ako je kategorizovana procenom jednog izostavljanja */
    learnParm.xa_depth = 0; /* parametar u xi/alpha proceni za leave-one-out odsecanje 
    granice broja porzavajucih vektora */
    learnParm.precfile[0] = 0; /* fajl za predikciju neobelezenih primera */
    learnParm.alfafile[0] = 0; /* fajl u kome se smesta optimalno alfa, koriste se prazni stringovi ukoliko alfa ne treba da budu izlazni*/

    // konstante koje ne treba dirati
    learnParm.epsilon_const = 1E-20;
    learnParm.epsilon_shrink = 1E-6;
    learnParm.opt_precision = 1E-21;
    learnParm.epsilon_a = 1E-15;

    // postavka kernela
    kernelParm.custom[0] = 0;
kernelParm.kernel_type = 0; /* 0=linear, 1=poly, 2=rbf, 3=sigmoid, 4=custom, 5=matrix */
kernelParm.poly_degree = 3;
kernelParm.rbf_gauss = 1.0;
kernelParm.coef_lin = 1;
kernelParm.coef_const = 1;
kernelCachedSize = 40;

// provera doslednosti parametara
if(learnParm.svm_iter_to_shrink == -9999) {
  if(kernelParm.kernel_type == LINEAR)
    learnParm.svm_iter_to_shrink=2;
  else
    learnParm.svm_iter_to_shrink=100;
}
if((learnParm.skip_final_opt_check)
&& (kernelParm.kernel_type == LINEAR)) {
  cout << "Preskakanje poslednje provere optimalnosti linearnih kernela!\n\n";
  learnParm.skip_final_opt_check=0;
}
if((learnParm.skip_final_opt_check)
&& (learnParm.remove_inconsistent))
  throw Izuzetak("Neophodno je odraditi poslednju proveru optimalnosti prilikom uklanjanja nekozistentnih primara!");
if((learnParm.svm_maxqpsize <2))
  throw Izuzetak("Maksimalna velicina podproblema kvadratnog programiranja nije u validnom opsegu od [2,...]!");
if((LearnParm.svm_maxqpsize>learnParm.svm_newvarsinqp))
  throw Izuzetak("Maksimalna velicina podproblema kvadratnog programiranja mora biti veca od broja novih promenljivih koje ulaze u radni skup pri svakoj iteraciji!");
if((learnParm.svm_iter_to_shrink1)
  throw Izuzetak("Maksimalan broj iteracija za smanjivanje nije u validnom opsegu [1,...]!");
if((LearnParm.svm_c<0))
  throw Izuzetak("Parametar C mora biti veci od nule!");
if((Learn Parm.transduction_posratio1)
  throw Izuzetak("Deo neobelezenih primara koji su klasifikovani kao pozitivni mora biti manji od 1.0!");
if((learnParm.epsilon_crit<=0))
  throw Izuzetak("Parametar sv_costratio mora biti veci od nule!");
if((LearnParm.rho<0))
  throw Izuzetak("Parametar rho za xi/alpha procenu i leave-one-out odsecanje mora biti veci od nule (obicno 1.0 ili 2.0)!");
if((learnParm.xa_depth<0) || (learnParm.xa_depth >100))
  throw Izuzetak("Parametar xa_depth za xi/alpha procenu mora biti u granicama [0..100]!");

// level of SVM-light debugging infos
verbosity = 1;
if((!softmargin)
  learnParm.svm_c = 1E15; // parametar C kod tvrde margine
})
else
  // stelovanje cene kostanja kojon se smnoz parametar C za pozitivne primere kod neke margine
  learnParm.svm_costratio = 0.3;
}
virtual "SVMKlasifikator"();

string getName()
{
  return "SVM Klasifikator";
}

void treniraj(const Dataset* data)
{
  // provera da li ima neeklasifikovanih instanci
  if (data->instances.size() == 0) throw Izuzetak("Za treniranje se ocekuje neprazan skup podataka!");
  "prevodjenje vektora instanci u skup DOC format za SMLight";
  DOC *docs; // trening primeri
  double *target; // oznake
  long totwords = data->instances[0].attributeCount; // broj atributa
  long totdoc = data->instances.size(); // broj primera
}
docs = (DOC*) malloc(sizeof(DOC)*totdoc);
target = (double *) malloc(sizeof(double)*totdoc);
if (!docs || !target)
    throw Izuzetak("Neuspesna alokacija memorije!");

int curDoc = 0;
for (Dataset::Instances::const_iterator i = data->instances.begin(); i != data->instances.end(); i++)
{
    instancaKonverzijaDOC(*i, docs[curDoc]);
    if (i->kategorija == NEDEFINISANO)
        throw Izuzetak("Nedefinisane instance nisu dozvoljene u trening skupu!");
    else if (i->kategorija == SPAM)
        target[curDoc] = 1.0;
    else
        target[curDoc] = -1.0;
    curDoc++;
}

// model za ucenje
if(kernelParm.kernel_type == LINEAR)
{
    cout << "treniranje klasifikatorsa" << endl;
    svm_learn_classification(docs,target,totdoc,totwords,&learnParm, &kernelParm,NULL,&model);
}
else
{
    KERNEL_CACHE kernelCache;
    kernel_cache_init(&kernelCache,totdoc,kernelCacheSize);
    svm_learn_classification(docs, target, totdoc, totwords, &learnParm, &kernelParm, &kernelCache, &model);
    // oslobodi memoriju koriscenu za kesisranje.
    kernel_cache_cleanup(&kernelCache);
}

MejlKlasa klasifikuj(const Instance& inst) const
{
    if (model.sv_num == 0)
        throw Izuzetak("Neobuceni klasifikator!");
    DOC doc;
    instancaKonverzijaDOC(inst, doc);
    if (classify_example(const_cast<MODEL*>(&model), &doc) > 0)
        return SPAM;
    else
        return LEGITIMNO;
}

*/
#include <iostream>
#include "Instance.h"
#include "IzdvajanjeOdlika.h"
#include "Klasifikator.h"
#include "NaivniBajesovKlasifikator.h"
#include "KlasifikatorKNajblizihSuseda.h"
#include "SVMKlasifikator.h"
#include "PerceptronKlasifikator.h"
#include "Klasifikator1od2.h"
#include "Klasifikator2od3.h"
#include "Izuzetak.h"
#include "Konfiguracija.h"
using namespace std;

int main(int argc, char* argv[]) {
    try {
        cout << "Klasifikacija poruka el. poste algoritmima masinskog ucenja." << endl;

        // Citanje parametara
        Konfiguracija podesavanja("parametri.txt", argc, argv);

        podeljivanje.setDefault("deokorpusa", ",0123Acorpora/pas_corpora_public/psl");
        podeljivanje.setDefault("deo-korpusa", "1"); //deo korpusa koji ostavljamo za validaciju
        podeljivanje.setDefault("odlike-od", "1"); //odlike koje ce biti izdvojene od ove reci
        podeljivanje.setDefault("odlike-do", "30000"); //odlike koje ce biti izdvojene do ove reci
        podeljivanje.setDefault("produzeni-validacioni-skup", "1"); // da li ce trening instance
        // biti dodate skupu za validaciju
        podeljivanje.setDefault("nb-lambda", "1"); // lambda Naivnog Bajesovog klasifikatora
        podeljivanje.setDefault("knn-k", "50"); // k najblizih suseda
        podeljivanje.setDefault("knn-l", "30"); // ako je l ili vise poruka medju k najblizih suseda
        // poruke x spam klasifikuj x kao spam
        podeljivanje.setDefault("svm-mm", "0"); // koriscenje neke nargine za SVM

        // kao klasifikator se testira (0 - ne testirati, 1 - testirati ) */
        podeljivanje.setDefault("naivnibajes", "1");
        podeljivanje.setDefault("knn", "1");
        podeljivanje.setDefault("perceptron", "1");
        podeljivanje.setDefault("trivijalni", "1");
        podeljivanje.setDefault("svm", "1");
        podeljivanje.setDefault("svm-naivnibajes", "1");
        podeljivanje.setDefault("svm-naivnibajes-perceptron", "1");

        vector<int> deoKorpusa = podeljivanje.getIntVectorParam("deo-korpusa", 1, 10);
        int odlikeOdReci = podeljivanje.getIntParam("odlike-od", 1, 90000);
        int odlikeDoReci = podeljivanje.getIntParam("odlike-do", 1, 90000);
        if (odlikeDoReci < odlikeOdReci)
            throw Izuzetak("parametar odlike-do treba biti veci od parametra odlike-od");
        int produzenaValidacija = podeljivanje.getIntParam("produzeni-validacioni-skup", 0, 1);
        int nbLambda = podeljivanje.getIntParam("nb-lambda", 1, 1000);
        int knnK = podeljivanje.getIntParam("knn-k", 1, 1000);
        int knnL = podeljivanje.getIntParam("knn-l", 1, 30);
        if (knnL > knnK)
            throw Izuzetak("knn-k treba da bude veca od knn-l");

        int svmMekarNargina = podeljivanje.getIntParam("svm-mm", 0, 1);
        int naivnibajes = podeljivanje.getIntParam("naivnibajes", 0, 1);
        int knn = podeljivanje.getIntParam("knn", 0, 1);
        int perceptron = podeljivanje.getIntParam("perceptron", 0, 1);
        int trivijalni = podeljivanje.getIntParam("trivijalni", 0, 1);
        int svm = podeljivanje.getIntParam("svm", 0, 1);
        int svmNaivnibajes = podeljivanje.getIntParam("svm-naivnibajes", 0, 1);
        int svmNaivnibajesPerceptron = podeljivanje.getIntParam("svm-naivnibajes-perceptron", 0, 1);

        // Priprema citanja instanci korpusa
        IzdvajanjeOdlika* io = new IzdvajanjeOdlikaKorpusa(odlikeOdReci, odlikeDoReci);
        CitanjeInstanciKorpusa korpus(podeljivanje.parametri["direktorijum-korpusa"], io);

        cout << "Pripremanje trenirajuceg skupa ( deo-korpusa=" << deoKorpusa.size() << " i odlike-=od-reci=" << odlikeOdReci << " i odlike-=do-reci=" << odlikeDoReci << " )" << endl;

        Dataset* treningSkup = korpus.kreiranjeTreningSkupa(deoKorpusa);
        if (produzenaValidacija)
            validacioniSkup->addInstances(treningSkup->instances);

        // kNN klasifikator
        Rastojanje* d = new ApsolutnoRastojanje();
        Klasifikator* knn = new KlasifikatorKNajblizihSuseda(d, knnK, knnL);
        // Naivni Bajesov klasifikator
        Klasifikator* nb = new NaivniBajesovKlasifikator(nbLambda);
        // Trivijalni klasifikator
        Klasifikator* tc = new TrivijalniKlasifikator(LEGITIMNO);
    } catch (Izuzetak& iz) {
        cout << iz.message() << endl;
    }

    // kNN klasifikator
    Rastojanje* d = new ApsolutnoRastojanje();
}

// Naivni Bajesov klasifikator
Klasifikator* nb = new NaivniBajesovKlasifikator(nbLambda);
// Trivijalni klasifikator
Klasifikator* tc = new TrivijalniKlasifikator(LEGITIMNO);
// SVM klasifikator
Klasifikator* svm = new SVMKlasifikator(svmMekaMargina == 1);
// Perceptron klasifikator
Klasifikator* perceptron = new PerceptronKlasifikator();
// SVM sa nekon marginin
Klasifikator* svmSoft = new SVMKlasifikator(true);
// 1 od 2 (SVM meka margina/Naivni Bajes)
Klasifikator* svmnb = new Klasifikator1od2(svmSoft, nb, SPAM);
// 2 od 3 (SVM meka margina/Naivni Bajes/Perceptron)
Klasifikator* svmnbperceptron = new Klasifikator2od3(svmSoft, nb, perceptron);

if (radiTrivijalniKlasifikator)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(tc, treningSkup, validacioniSkup);
if (radiPerceptron)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(perceptron, treningSkup, validacioniSkup);
if (radiNaivniBajes)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(nb, treningSkup, validacioniSkup);
if (radiSVM)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(svm, treningSkup, validacioniSkup);
if (radiSvmNaivniBajes)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(svmnb, treningSkup, validacioniSkup);
if (radiSvmPerceptron)
    ProcenaPerformansiKlasifikatora::testirajKlasifikator(svmnbperceptron, treningSkup, validacioniSkup);

cout << "Uspešno završeno.\n" << endl;

} catch(Izuzetak e){
    cerr << "Greska: " << e.message << endl;
    return 1;
}
return 0;