UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Luka Kavčič

Bayesovo filtriranje nezaželene elektronske pošte

Magistrsko delo

Ljubljana, 2018

UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Luka Kavčič

Mentor: izr. prof. dr. Damjan Škulj

Bayesovo filtriranje nezaželene elektronske pošte

Magistrsko delo

Ljubljana, 2018

Zahvalil bi se očetu, da mi je privzgojil spoštovanje do
intelektualnega mišljenja, mami Jasni za skrb in
Ani, ker mi stoji ob strani.

Bayesovo filtriranje nezaželene elektronske pošte

Naloga izvira iz problematike nezaželene spletne pošte, ki se pojavlja že od časa, ko so nastajale prve spletne skupnosti. Uporabniki spletne pošte porabijo več časa za prebiranje pošte, ko je v nabiralniku nezaželena pošta. Zmanjševanje nezaželene pošte v uporabnikovem nabiralniku je naloga spletnih filtrov. V ta namen se uporablja razvrščanje, s katerim zmanjšujemo možnost prejetja nezaželene pošte. Na podlagi teoretskega izhodišča v nalogi za razvrščanje uporabljam Bayesov naivni klasifikator, s katerim sporočila razvrščamo v dve skupini, in njegovo nadgrajeno različico, ki razvršča sporočila v tri skupine. Predlagana rešitev z uvedbo nove kategorije cilja na izboljšanje razvrščanja v cenovno-občutljivih kontekstih. S pomočjo Bayesove teorije odločanja in računanja verjetnostnih mej lahko spletni filter sporočila razvršča v tri kategorije. Za namen naloge sem sprogramiral lasten spletni filter, ki omogoča primerjavo obravnavanih algoritmov in oceno uspešnosti le-teh. Za učenje filtra in razvrščanje sporočil sem uporabil pošto iz lastnega poštnega računa. Rezultati naloge potrjujejo hipotezo o boljši uspešnosti razvrščanja z uporabo predlagane izboljšave osnovnega algoritma v cenovno-občutljivih primerih.

Ključne besede: Bayes, Naivni Bayesov klasifikator, razvrščanje, nezaželena pošta.

Bayes spam classification

This work addreses the problem of spam mail which has plagued users inboxes since the beggining of first online groups. It takes user more time to read mails in inbox when it contains a lot of unsolicited mail. Therefore, the reduction of spam is a filter's prime concern. To do that, as the name suggest, filtering is used. I use Naive Bayes classificator which classifies mail in two groups and compare it to its' modified version which classifies mails in three groups. Improvent of classification in cost-sensitive scenarios is what the suggested solution tries to do. To achieve this goal I used Bayes decision theory and calculation of probability thresholds. In order to implement the classification algorithms and to test their effectivness, a spam filter was programmed. To train the filter and to test message classification, mail from author's inbox was used. The proposed solution compared to base method, as seen from the results, presents an improvement in spam classification in cost-sensitive scenarios.

Keywords: Bayes, naive Bayes classification, classification, spam.

Kazalo vsebine

Kazalo tabel

Kazalo slik

1 UVOD

Nezaželena elektronska pošta je vsakodnevni pojav, s katerim se uporabniki srečujejo že od začetka svetovnega spleta. Strokovna poročila kažejo, da taka pošta predstavlja 53 odstotkov celotne pošte (Symantec Corporation, 2017). Pri tovrstni pošti gre večinoma za oglaševanje ali spletne prevare (Arnes, 2012). Brisanje elektronskih pisem pomeni izgubo časa za uporabnika pri prebiranju pošte. Taka pošta lahko vsebuje zlonamerno kodo, kar povzroča dodatne stroške uporabnikom storitve.

Številni posamezniki, raziskovalci, podjetja in organizacije se že vrsto let ukvarjajo z reševanjem te problematike. Skozi čas je bilo razvitih več načinov za preprečevanje nezaželene elektronske pošte. V ta namen so razvili poštne filtre. Eden izmed bolj popularnih je filtriranje elektronske pošte na podlagi razvrščanja. Filter na podlagi algoritma razvrsti pošto v različne kategorije. Največkrat se uporablja naivni Bayesov algoritem, pravimo mu tudi klasifikator. Algoritem je bil mnogokrat uporabljen kot osnova za odprtokodne projekte, prav tako tudi za komercialne produkte, saj ga krasijo preprostost, računska učinkovitost in uspešnost (Zhou, Yao in Luo, 2014).

Metoda temelji na uporabi pogojne verjetnosti, s katero je moč izračunati verjetnost, da je sporočilo zaželeno. Gre za razvrščanje v dve skupini, se pravi, sporočilo je lahko zaželeno ali nezaželeno. Navkljub njeni preprostosti metoda ni brez pomanjkljivosti, saj lahko zaradi odločanja samo v dve skupine pride do napake, ki za uporabnika predstavlja dodaten strošek.

Pristop, ki bi lahko izboljšal učinkovitost razvrščanja, je uporaba treh skupin v katere razvrščamo pošto. V tretjo skupino grejo tista sporočila, za katera filter ni prepričan, ali so zaželena ali ne. O zaželenosti oziroma nezaželenosti takih sporočil odloča uporabnik z ročnim razvrščanjem. S tem načinom se lahko izognemo primeru, ko bi lahko filter zaželeno sporočilo označil kot nezaželeno. Za uporabnika to pomeni, da je manj možnosti, da bi izgubil želeno sporočilo. Naivni Bayesov klasifikator na podlagi formule vrne verjetnost o zaželenosti sporočila. Določanje meje verjetnosti pri kateri je sporočilo zaželeno ali nezaželeno, je lahko težavno. Višja meja lahko pomeni večjo preciznost (angl. *precision*) in majhen priklic (angl. *recall*), nižja meja lahko vodi do obratnih rezultatov. Preciznost meri stopnjo do katere so zavrnjena sporočila resnično nezaželena (Androutsopoulos, Koutsias, Chandrinos, Paliouras, Spyropoulos, 2004). Mera lahko zavzema vrednosti od 0 do 1, kjer

»1« pomeni, da so vsa zavrnjena sporočila bila resnično nezaželena. Priklic meri odstotek uspešno zavrnjenih nezaželenih sporočil (Androutsopoulos in drugi, 2000).

Razvrščanje v tri skupine uporablja dve meji, s katerimi razlikujemo med tremi skupinami: zaželena, nezaželena in nerazvrščena. Uporabnik lahko pregleda zaželena sporočila, nezaželena izbriše, nerazvrščena lahko pregleda kasneje. Skupina nerazvrščenih sporočil zmanjšuje napake razvrščanja in prinaša napako nerazvrstitve (Zhou in drugi, 2014). Razvrščanje v tri skupine ni nov pristop, saj je bilo v literaturi že večkrat omenjeno. Robinson predlaga uvedbo območja *nesigurno* (angl. *unsure*) (Robinson v Zhou in drugi 2014, 21). Yih in drugi govorijo o *sivi* pošti (angl. *grey mail*), to je, pošta, ki ne spada v kategorijo zaželene ali nezaželene pošte (Yih, McCann, Kolcz, 2007 v Zhou in drugi 2014, str 20)[1]. Med drugim je razvrščanje v tri skupine uporabljeno tudi v aplikacijah, kot je SpamBayes (»SpamBayes: Bayesian anti-spam classifier written in Python«, b. d.), SpamAssasin (»Apache SpamAssasin«, 2018), BogoFilter (»BogoFilter: Fast Bayesian Spam Filter«, b. d.). Prednost razvrščanja v tri skupine je v tem, da omogoča možnost neodločitve, v primerih ko verjetnost ni izrazito visoka oziroma nizka. Kar nam lahko pride prav v primeru, ko strošek neodločanja ni visok. (Zhou in drugi 2014). Vsako dejanje filtra predstavlja za uporabnika določen strošek. Višina stroška sloni na uporabnikovem vrednotenju posledic izbrane odločitve. Razvrščanju, ki upošteva uporabnikovo vrednotenje posledic odločitve pravimo cenovno občutljivo razvrščanje. V ta namen bom za določanje meja verjetnosti uporabil Bayesovo teorijo odločanja, ki nam omogoča, da na podlagi uporabnikovih stroškovnih preferenc določimo optimalnejše meje verjetnosti (Yao in drugi, 1990). Obstoječi filtri namreč uporabljajo arbitrarno določene meje. SpamBayes ima privzeto nastavitve pri mejah 0.2 in 0.9 (SpamBayes: Bayesian anti-spam classifier written in Python, b. d.). Z predlagano rešitvijo se filter prilagodi uporabniku in zmanjšuje možnost napake.

[1] Yih, W., Mccann, R. in Kołcz, A. (2007). *Improve Spam Filtering by Detecting Gray Mail*. Predstavljeno na Fourth Conference on Email and Anti-Spam, Mountain View, ZDA, Avgust 2-3.

2 TEORETSKO IZHODIŠČE

2.1 NEZAŽELENA ELEKTRONSKA POŠTA

Nezaželeno spletno pošto lahko definiramo kot pošto, ki je poslana večjemu številu naročnikov, brez vednosti oziroma dovoljenja naročnika. Gre za masovno pošiljanje pošte, ki vsebuje: oglaševanje storitev ali izdelkov, oglaševanje pornografskih vsebin, vabila v piramidalne sheme, lažne novice ali programsko opremo (NetZero, b. d.). Poleg oglaševanja se pojavljajo tudi različne vsebine s katerimi naj bi naročnika ogoljufali. Nezaželena spletna pošta predstavlja več kot polovico prometa elektronske pošte. Njen obseg se v zadnjih letih zmanjšuje. Pošiljanje nezaželene pošte je v večini držav sveta zakonsko prepovedano. Zato se ljudje, ki stojijo za takimi dejanji, poslužujejo različnih načinov pošiljanja sporočil brez vednosti o njihovi identiteti. Nezaželena pošta gre z roko v roki z zlonamerno kodo, saj le ta lahko omogoči, da se okužen računalnik uporabi v namene pošiljanja nezaželenih sporočil (Arnes, b. d).

V Sloveniji je področje neposrednega trženja s pomočjo elektronskih komunikacij urejeno s tremi zakoni:

- Zakon o elektronskih komunikacijah (Zakon o elektronskih komunikacijah – ZEKom 1, 2012)[2]
- Zakon o elektronskem poslovanju na trgu (Zakon o elektronskem poslovanju na trgu – ZEPT, 2005)[3]
- Zakon o varstvu osebnih podatkov (Zakon o varstvu osebnih podatkov – ZVOP-1, 2004)[4]

Glavna načela povzeta iz zakonov so naslednja:

- pošiljatelj mora predhodno pridobiti soglasje vsakega naslovnika,
- naslovnik ima pravico kadarkoli zavrniti nadaljnjo uporabo svojega elektronskega naslova,
- pošiljatelj mora pri obdelavi osebnih podatkov upoštevati Zakon o varstvu osebnih podatkov.

(SI-Cert, b. d.).

[2] Zakon o elektronskih komunikacijah. (2012). *Uradni list RS, št. 109/12.* 31. December.
[3] Zakon o elektronskem poslovanju na trgu – ZEPT (2006). *Uradni list RS, št. 61/06*. 13. Junij.
[4] Zakon o varstvu osebnih podatkov – ZVOP-1. (2004). *Uradni list RS, št. 86/04.* 5. Avgust.

Zgodovina nezaželene pošte sega v leto 1973, ko je uporabnik ARPANET-a (predhodnik interneta) poslal sporočilo vsem uporabnikom tega omrežja. Šlo je za oglas računalniškega podjetja (Digital Equipment Corporation). Leta 1993 je uporabnik omrežne skupine »USENET« poslal 200 nezaželenih sporočil zaradi napake v programski opremi. Po tem dogodku naj bi nezaželena pošta dobila ime »spam«. Ime se nanaša na skeč britanske televizijske serije »Monty Python's Flying Circus«, v katerem prevladuje beseda »spam«. Spam v angleščini označuje kuhano meso v konzervi. Do leta 2003 se je zgodilo še nekaj takih in podobnih dogodkov. Od takrat dalje so bila takšna dejanja resneje sankcionirana. V ZDA so namreč leta 2003 uvedli zakon »CAN-SPAM«, ki ureja področje elektronskega trženja (Protocol labs, b. d).

2.2 POŠTNI FILTRI

Poštni filter je program napisan z namenom sprejemanja zaželenih in zavračanja nezaželenih sporočil. Poznamo več načinov filtriranja pošte. Eden izmed bolj preprostih je filtriranje na podlagi pravil, ki določajo katera sporočila so zaželena oziroma nezaželena. Na primer, pravilo, zavrača vsa sporočila, ki vsebujejo besedno zvezo »na računu vas čaka«. Za zaželeno pošto bi lahko določili pravilo, da filter sprejme sporočilo, če vsebuje besedo besedo »Python«, ker me zanima programiranje v Python-u. Na podlagi več pravil bi filter lahko določil, katera pravila tako sprejel odločitev. Da bi ustvarili filter, ki dobro dela je potrebno napisati niz pravil, ki imajo smisel. Vedeti moramo, da se snovalci nezaželene pošte vsega tega zavedajo in to znanje uporabljajo pri pisanju nezaželene pošte. To pomeni, da nezaželena sporočila lahko nimajo nezaželenega videza, zaradi česar jo je težko uloviti. Pisanje pravil, ki bi zajela vsa nezaželena sporočila je zahtevno (Chiarella in O'Brien, 2004). Drug način razvrščanja temelji na uporabi seznama pošiljateljev nezaželene pošte. Ko uporabnik prejme nezaželeno pošto, doda pošiljateljev naslov na seznam »blokiranih« naslovov. To pomeni, da sporočila, ki pridejo iz seznama nezaželenih pošiljateljev filter zavrne. Pošiljatelji nezaželenih sporočil se temu izognejo tako, da sporočilo pošljejo iz drugačnega naslova. Na drugi strani imamo način, kjer beležimo seznam pošiljateljev zaželene pošte. Na seznam dodajamo pošiljatelje za katere vemo, da so zaželeni. Pojavi se težava, saj je potrebno vsakega pošiljatelja predhodno odobriti, kar pomeni, da lahko nehote zavrnemo sporočilo, ki je zaželeno.

Poznamo tudi druge načine razvrščanja, ki temeljijo na podlagi verjetnosti. Leta 2002 je Paul Graham objavil, članek »A plan for Spam«, v katerem govori o pomanjkljivostih filtrov, ki uporabljajo pravila (Graham, 2002). V članku predlaga nov način razvrščanja, ki uporablja Bayesov verjetnostni način razmišljanja. To pomeni, da poštni filter s pomočjo Bayesovega teorema izračuna verjetnost zaželenosti sporočila, če vsebuje določene besede. Prednost Bayesovega razvrščanja je v tem, da se filter sproti uči (Chiarella in drugi, 2004).

Delovanje filtra je sestavljeno iz več delov. V prvem delu filter se filter nauči razlikovati med zaželenimi in nezaželenimi sporočili. Za to potrebuje čim večje število sporočil za katera vemo ali so zaželena. V splošnem poštni filtri delujejo po principu vreče besed (angl. *bag of words*), kar pomeni, da se za računanje verjetnosti uporabi besede, ki jih sporočilo vsebuje. Pred samim filtriranjem je potrebno poštno sporočilo pripraviti, da ga lahko filter obdela. Dele sporočila, ki niso relevantni za obdelavo se izbriše in ne upošteva pri računanju verjetnosti. V naslednjem koraku se vsebina pošte v celoti pregleda in razdeli na posamezne besede. Dobljene besede, ki se pogosto uporabljajo v zaželenih in nezaželenih sporočilih se odstrani, saj ne povejo veliko o tem ali je sporočilo zaželeno ali ne. V nadaljevanju se besede skrajša do korena. Naslednji korak je izbira besed, ki se uporabijo pri računanju verjetnosti. Večina filtrov namreč ne uporabi vseh besed v sporočilu, saj je le-teh lahko ogromno. Paul Graham je v svojem filtru uporabil 15 besed (Graham, 2002). Izbira besed poteka s pomočjo za to namenjenih algoritmov (npr. »Mutual Information index«) s katerimi se določi pomembnost besede. Nazadnje nam ostane tabela besed, v kateri so za vsako besedo označene njene frekvence, to je, pojavljanje v poštnih sporočilih (Zhou in drugi 2014). Z dobljeno tabelo lahko filter začne učenje na podlagi izbranega algoritma. Obstaja več različnih algoritmov, med drugim, metoda najbližjih sosedov, nevronske mreže in metoda podpornih vektorjev so samo nekateri izmed njih.

2.3 VERJETNOSTNI RAČUN

Preden se podamo k naivnemu Bayesovemu klasifikatorju moram objasniti nekaj osnovnih pojmov iz računanja verjetnosti. Pojmi, ki jih uporabljamo v verjetnostnem računu so naslednji: poskus, dogodek in verjetnost dogodka (Ferligoj, 1995).

2.3.1 POSKUS

»Poskus je realizacija neke množice skupaj nastopajočih dejstev (kompleksa pogojev). Poskus je torej vsako dejanje, ki ga opravimo v natanko določenih pogojih.«(Ferligoj, 1995, str.69). V situaciji razvrščanja spletne pošte poskus predstavlja novo prispelo sporočilo.

2.3.2 DOGODEK

»Pojav, ki v množico skupaj nastopajočih dejstev ne spada in se lahko v posameznem poskusu zgodi ali pa ne, imenujemo dogodek« (Ferligoj, 1995, str.69). Pri spletni pošti imamo torej, sporočila, ki prihajajo v naš nabiralnik. Zgodi se lahko dogodek, da je sporočilo zaželeno ali ne.

Dogodke razvrstimo na:

- gotov dogodek – vedno se zgodi

- nemogoč dogodek – ne zgodi se

- slučajen dogodek – zgodi se včasih

Poznamo več načinov računanja z dogodki. Pravimo, da je dogodek A način dogodka B ($A \subset B$), če se vsakič, ko se zgodi dogodek A zagotovo zgodi tudi dogodek B. Če je dogodek A način dogodka B in sočasno dogodek B način dogodka A, sta dogodka enaka:

$$A \subset B \wedge B \subset A \Leftrightarrow A = B \qquad (1)$$

(Ferligoj, 1995, str.69)

Vsota dogodkov A in B ($A \cup B$), je takrat, ko se zgodi vsaj eden A in B.

Produkt dogodkov A in B ($A \cap B$), pomeni, da se zgodita dogodka A in B hkrati.

Nasproten dogodek $\bar{A}$ je negacija dogodka A. Vzemimo, da dogodek A pomeni, da smo prejeli zaželeno sporočilo in dogodek B, da smo prejeli nezaželeno sporočilo. Potem je nasprotje dogodka A, to da smo prejeli nezaželeno sporočilo.

Nezdružljiva dogodka sta dogodka A in B, takrat, ko se ne moreta zgoditi skupaj. Produkt teh dveh dogodkov je nemogoč dogodek.

$$A \cap \mathrm{B} = \mathrm{N}. \tag{2}$$

Na našem primeru je to jasno razvidno, saj se ne more zgoditi, da bi hkrati prejeli zaželeno in nezaželeno sporočilo.

Sestavljen dogodek A je tisti dogodek, ki ga lahko izrazimo kot vsoto nezdružljivih dogodkov. Lahko bi rekli, da je dogodek, da prejmemo sporočilo sestavljen dogodek.

Elementaren dogodek je dogodek, ki ni sestavljen.

Popoln sistem dogodkov imenujemo množico dogodkov $S = \{A_1, A_2, \ldots, A_n\}$, če se v vsaki ponovitvi poskusa zgodi natanko eden od dogodkov iz množice S. (3)

Vsi dogodki so mogoči

$$A_i \neq \mathrm{N} \tag{4}$$

paroma nezdružljivi

$$A_i \cap A_j = \mathrm{N} \ \mathrm{i} \neq j \tag{5}$$

Njihova vsota je gotov dogodek

$$A_1 \cup A_2 \cup \ldots \cup A_n = G \tag{6}$$

(Ferligoj, 1995, str.69)

2.3.3 VERJETNOST

»Verjetnost dogodka A v danem poskusu je število P(A) pri katerem se navadno ustali relativna frekvenca dogodka A v velikem številu ponovitev« (Ferligoj, 1995, str.77)

Relativna frekvenca je število ponovitev poskusa v katerih se zgodi dogodek A (*k*), v primerjavi z vsemi ponovitvami poskusa (*n*).

$$f(A) = \frac{k}{n} \tag{7}$$

Lastnosti verjetnosti:

Nenegativnost

Ker je relativna frekvenca vedno nenegativna.

$$\mathrm{P(A)} \geq 0 \tag{8}$$

Normiranost

Verjetnost gotovega dogodka (G) je 1.

$$\mathrm{P(G)} = 1 \tag{9}$$

Aditivnost

Če sta dogodka A in B nezdružljiva potem velja:

$$\mathrm{P(A} \cap \mathrm{B)} = \mathrm{P(A)} + \mathrm{P(B)} \tag{10}$$

2.3.4 POGOJNA VERJETNOST

Pogojna verjetnost P(A/B) pomeni, verjetnost dogodka A, če se je zgodil dogodek B.

$$\mathrm{P(A \mid B)} = \frac{\mathrm{P(A} \cap \mathrm{B)}}{\mathrm{P(B)}} \tag{11}$$

kjer je $\mathrm{P(B)} > 0$.

Lahko izpeljemo,

$$P(A \cap B) = P(A \mid B)P(B) \tag{12}$$

in

$$P(B \cap A) = P(B \mid A)P(A) \tag{13}$$

Ker je

$$P(A \cap B) = P(B \cap A) \tag{14}$$

sledi,

$$P(A \mid B)P(B) = P(B \mid A)P(A). \tag{15}$$

dobimo *Bayesov izrek*:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{16}$$

Uporabimo *zakon popolne verjetnosti*

$$P(A) = \sum_{i=1}^{n} P(H_i)P(A \mid H_i) \tag{17}$$

Popoln sistem dogodkov razbijemo na posamezne dele (A_1, A_2, A_3,… A_n). Elementom te množice pravimo tudi hipoteze (Kolmogorov, 1956).

Bayesovo formulo lahko posplošimo, če vanj vstavimo formula iz zakona o popolni verjetnosti.

$$P(H_i \mid B) = \frac{P(H_i)P(A \mid H_i)}{\sum_{i=1}^{n} P(H_i)P(A \mid H_i)} \tag{18}$$

Thomas Bayes je živel med leti 1701 in 1761. Rojen je bil v Walesu. Bil je pripadnik duhovščine, ukvarjal se je z znanostjo, predvsem z matematiko. Študiral je logiko in teologijo na Edinburški Univerzi, leta 1742 je postal član znanstvene akademije (angl. *Royal Society*). Pojav zavarovalništva in igre na srečo so pripeljale do tega, da je Bayes objavil publikacijo v kateri se je ukvarjal z verjetnostno porazdelitvijo. Bayesov teorem, kot ga poznamo danes, je neodvisno odkril francoski matematik Pierre-Simon Laplace (Papoulis, 1991).

2.4. NAIVNI BAYESOV KLASIFIKATOR

Kot že ime pove, metoda razvršča predmet preučevanja v zato namenjene razrede (C_i). Predmet preučevanja lahko predstavimo z vektorjem $\vec{x} = (x_1, x_2, \dots, x_n)$, kjer so elementi vektorja značilnosti (npr. besede v sporočilu) predmeta, ki ga preučujemo (npr. elektronsko sporočilo).

Pri naivnem Bayesovem klasifikatorju se predpostavlja neodvisnost elementov (x_n), to je, da niso med seboj povezani. Gre se za to, da zmnožimo pogojne verjetnosti vseh posameznih elementov. V primeru razvrščanja so to posamezne besede, ki so v sporočilu.

$$P(X_i|A) = P(X_i, \dots, X_n|\mathrm{A}) = \prod_{i=1}^{n} P\,(X_i|\mathrm{A}) \tag{19}$$

V prejšnjem poglavju predstavljeno Bayesovo formulo (18) lahko nadgradimo z spodnjo formulo.

$$P = (\mathrm{A}|x_i) = \frac{P(A)\prod_{i=1}^{n} P(x_i|)}{P(x_i|A_i)P(\mathrm{A}) + \dots + P(x_i|\mathrm{A})P(\mathrm{A})} \tag{20}$$

Z uporabo zgornje formule lahko izračunamo verjetnost, da je sporočilo nezaželeno, če vsebuje določene besede.

2.5 BAYESOVA TEORIJA ODLOČANJA

Bayesova teorija odločanja odgovarja na izziv odločanja v tveganju. Teorija pravi, da je optimalna tista odločitev, pri kateri je strošek najmanjši. Stroškovna funkcija pove, kakšen je strošek posamezne odločitve (Duda in Heart 2000). Odločitve, ki jih sprejemamo, nas lahko različno stanejo. Na primer, če filter zavrne zaželeno sporočilo lahko to za uporabnika predstavlja večji strošek, kot če filter sprejme nezaželeno spročilo. Najprej moramo uvesti končno množico odločitev $(a_i, \ldots, a_n)$. Element množice predstavlja posamezno odločitev. Stroškovno funkcijo zapišemo takole: $\lambda(a_i|B_i)$. Funkcija predstavlja strošek, ki je povezan z odločitvijo pri določenem dogodku. Recimo, da funkcija predstavlja strošek, ko se filter odloči, da sprejme nezaželeno sporočilo. Strošek odločitve je odvisen od tega kako ga uporabnik določi oziroma dojema. Pri odločanju v negotovosti potrebujemo poleg ovrednotenih odločitev tudi podatke o pogojni verjetnosti dogodka. To je verjetnost dogodka, pod pogojem, da se zgodi nek drug dogodek. Na primer, verjetnost, da je sporočilo nezaželeno, če vsebuje določeno besedo.

Poglejmo sedaj, kako lahko izračunamo pričakovan strošek.

$$R(a_i|X) = \sum_{j=1}^{n} \lambda(a_i|B_i)P(B_i|X) \tag{21}$$

Omenjeni formuli pravimo tudi pogojno tveganje (Duda in drugi, 2000).

Teorija odločanja pravi, da moramo za vsako možno odločitev izračunati pogojno tveganje in poiskati tisto odločitev, kjer je tveganje najmanjše. Na ta način pridemo do rešitve pri kateri je skupno tveganje najmanjše. (Duda in drugi, 2000).

2.6. RAZVRŠČANJE POŠTE

Elektronsko sporočilo lahko predstavimo z vektorjem x in pripadajočimi elementi $(x_1, x_2, \ldots, x_n)$, ki povejo ali se beseda pojavlja v sporočilu ali ne.

C - zaželeno sporočilo

C^c - nezaželeno sporočilo

Uvesti moram tudi diskriminantno funkcijo $f(x)$, ki mi bo služila za razvrščanje na podlagi različnih vrednosti. V tem primeru je diskriminantna funkcija verjetnost, da je sporočilo nezaželeno, če vsebuje določene besede. V nalogi se osredotočam na razvrščanje pošte v različne kategorije glede na meje diskriminantne funkcije $f(x)$. Na ta način lahko definiramo dve območji, to je, pozitivno in negativno. Naj omenim, da je podlaga za uporabo različnih območij teorija grobih množic ki jo je uvedel Poljak Zdzisław I. Pawlak leta 1981 (Pawlak, 1981). Grobe množice imenujemo tako, ker jim ne moremo z gotovostjo določiti meje. Zato jim določimo meje z navadnimi množicami. V nalogi bom govoril o območjih razvrščanja.

$POZ_{(y)}C = \{x \mid f(x) \geq \gamma\}$ - območje zaželenih sporočil

$\mathrm{NEG}_{(y)}C = \{x \mid f(x) < \gamma\}$ - območje nezaželenih sporočil

Pri razvrščanju v tri skupine se odločamo med tremi skupinami. Uporabimo par meja $\alpha \; in \; \beta$ ($0 \leq \beta \leq \alpha \leq 1$), s katerimi razločujemo različne vrednosti funkcije $f(x)$.

α - meja verjetnosti zaželene pošte

β - meja verjetnosti nezaželene pošte

Tako lahko razločimo tri območja v katere filter razvršča:

$POZ_{(\alpha,\beta)}C = \{x \mid f(x) \geq \alpha\}$ - območje zaželenih sporočil

$\text{MEJ}_{(\alpha,\beta)}C = \{x \mid \beta < f(x) < \alpha\}$ – (mejno) območje nerazvrščenih sporočil

$\text{NEG}_{(\alpha,\beta)}C = \{x \mid f(x) < \beta\}$ - območje nezaželenih sporočil

Sporočilo je označeno za zaželeno, če je vrednost funkcije $f(x)$ večja ali enaka α. V primeru, da je vrednost funkcije $f(x)$ enaka ali manjša od β, je sporočilo označeno kot nezaželeno. Ko je vrednost verjetnosti med obema mejama, gre pošta v skupino nerazvrščeno (Zhou in drugi, 2014).

2.7. RAČUNANJE MEJNIH VREDNOSTI VERJETNOSTI

Poglejmo sedaj postopek za računanje mejnih vrednosti verjetnosti, da je sporočilo nezaželeno, če vsebuje določene besede.

Potrebujemo še množico možnih odločitev $\{a_P, a_M, a_M\}$. Množica ima tri elemente, ki predstavljajo tri možne odločitve pri razvrščanju. V spodnjih razlagah x predstavlja sporočilo.

a_P- $x \in POZ(C)$

a_M- $x \in MEJ(C)$

a_N- $x \in \text{NEG}(C)$

Stroškovne funkcije za vse možne odločitve so predstavljene v tabeli 2.1.

Tabela 2.1 : Odločitvena pravila

odločitev filtra	C(P):pozitivno	C^c(N):negativno
a_P - sprejme	$\lambda_{PP}=\lambda(a_P \mid C)$	$\lambda_{PN}=\lambda(a_p \mid C^c)$
a_M - odloži	$\lambda_{MP}=\lambda(a_M \mid C)$	$\lambda_{MP}=\lambda(a_M \mid C^c)$
a_N - zavrne	$\lambda_{NP}=\lambda(a_{NP} \mid C)$	$\lambda_{NN}=\lambda(a_N \mid C^c)$

Poznamo tri možne odločitve: sprejemaje, nerazvrščanje in zavračanje sporočil. Stroškovnih funkcij je dvakrat toliko, saj imamo dve možni stanji, to je, da je sporočilo zaželeno oziroma nezaželeno. Vzemimo za primer stroškovno funkcijo $\lambda_{PP} = \lambda(a_P \mid C)$, ki predstavlja strošek, če filter sprejme sporočilo, ko je dejansko stanje sporočila zaželeno. Na drugi strani funkcija $\lambda_{PN} = \lambda(a_P \mid C^c)$, predstavlja strošek, če filter sprejme sporočilo, ki ni zaželeno. Bayesova teorija odločanja zahteva, da vsaki funkciji določimo vrednost.

Na podlagi stroškovnih funkcij iz tabele lahko določimo tudi pogojna tveganja za vse možne odločitve:

$$R(a_P \mid x) = \lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x) \tag{22}$$

$$R(a_M \mid x) = \lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x) \tag{23}$$

$$R(a_N \mid x) = \lambda_{NP}P(C \mid x) + \lambda_{NN}P(C^c \mid x) \tag{24}$$

Postopek Bayesove teorije odločanja pravi, da moramo za vsako možno odločitev, poiskati tisto, ki ima najmanjšo vrednost pogojnega tveganja. Tako lahko zapišemo naslednja pravila:

(P) Sporočilo gre v *pozitivno* območje, če velja:

$\lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x) \leq \lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x)$ in

$\lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x) \leq \lambda_{NP}P(C \mid x) + \lambda_{NN}P(C^c \mid x)$.

(M) Sporočilo gre v *mejno* območje, če velja:

$\lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x) \leq \lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x)$ in

$\lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x) \leq \lambda_{NP}P(C \mid x) + \lambda_{NN}P(C^c \mid x)$.

(N) Sporočilo gre v *negativno* območje, če velja:

$\lambda_{NP}P(C \mid x) + \lambda_{NN}P(C^c \mid x) \leq \lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x)$ in

$\lambda_{NP}P(C \mid x) + \lambda_{NN}P(C^c \mid x) \leq \lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x)$.

Glede na to, da je $P(C \mid x) + P(C^c \mid x) = 1$ lahko prej našteta pravila poenostavimo samo z verjetnostjo $P(C \mid x)$ in stroškovno funkcijo λ.

Na podlagi predstavljenih pravil lahko izpeljemo parametre, ki nam bodo služili kot mejna vrednost verjetnosti.

Prvi del pravila (P)

$$\lambda_{PP}P(C \mid x) + \lambda_{PN}P(C^c \mid x) \leq \lambda_{MP}P(C \mid x) + \lambda_{MN}P(C^c \mid x)$$

$$\Leftrightarrow \lambda_{PP}P(C \mid x) + \lambda_{PN}\big(1 - P(C \mid x)\big) \leq \lambda_{MP}P(C \mid x) + \lambda_{MN}\big(1 - P(C \mid x)\big)$$

$$\Leftrightarrow \lambda_{PP}P(C \mid x) + \lambda_{PN} - \lambda_{PN}P(C \mid x) \leq \lambda_{MP}P(C \mid x) + \lambda_{MN} - \lambda_{MN}P(C \mid x)$$

$$\Leftrightarrow \lambda_{PN} - \lambda_{MN} \leq \lambda_{MP}P(C \mid x) - \lambda_{MN}P(C \mid x) - \lambda_{PP}P(C \mid x) + \lambda_{PN}P(C \mid x)$$

$$\Leftrightarrow \frac{(\lambda_{PN} - \lambda_{NN})}{(\lambda_{PN} - \lambda_{MN}) + (\lambda_{MP} - \lambda_{PP})} \leq P(C \mid x)$$

$$\Leftrightarrow P(C \mid x) \geq \frac{(\lambda_{PN} - \lambda_{NN})}{(\lambda_{PN} - \lambda_{MN}) + (\lambda_{MP} - \lambda_{PP})} \tag{25}$$

Podobno naredimo še za ostala pravila.

$$P(C \mid x) \geq \frac{(\lambda_{PN} - \lambda_{NN})}{(\lambda_{PN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{PP})} \tag{26}$$

(M)

$$P(C \mid x) \geq \frac{(\lambda_{PN} - \lambda_{MN})}{(\lambda_{PN} - \lambda_{MN}) + (\lambda_{MP} - \lambda_{PP})} \tag{27}$$

$$P(C \mid x) \geq \frac{(\lambda_{MN} - \lambda_{NN})}{(\lambda_{MN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{MP})} \tag{28}$$

(N)

$$P(C \mid x) \geq \frac{(\lambda_{PN} - \lambda_{NN})}{(\lambda_{PN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{PP})} \tag{29}$$

$$P(C \mid x) \geq \frac{(\lambda_{MN} - \lambda_{NN})}{(\lambda_{MN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{MP})} \tag{30}$$

Ker vemo, da je mejno območje med pozitivnim in negativnim območjem.

(POZ $\geq$ MEJ $\geq$ NEG) lahko uporabimo formuli (27) in (28):

$$\alpha = \frac{(\lambda_{PN} - \lambda_{MN})}{(\lambda_{PN} - \lambda_{MN}) + (\lambda_{MP} - \lambda_{PP})} \tag{31}$$

$$\beta = \frac{(\lambda_{MN} - \lambda_{NN})}{(\lambda_{MN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{MP})} \tag{32}$$

2.8. RAČUNANJE VERJETNOSTI ZAŽELENOSTI SPOROČILA

Za računanje verjetnosti sporočila bom uporabil spodnjo formulo.

$$P = (\mathrm{C}|X_i) = \frac{P(\mathrm{C}) \prod_{i=1}^{n} P(X_i|\mathrm{C})}{P(\mathrm{C}) \prod_{i=1}^{n} P(X_i|\mathrm{C}) + P(\mathrm{C}^c) \prod_{i=1}^{n} P(X_i|\mathrm{C}^c)} \tag{33}$$

S pomočjo parametrov pridobljenih v prejšnjem poglavju in vrednosti verjetnosti lahko določimo pravila za razvrščanje pošte.

(P) Če je $P(C \mid x) \geq \alpha$ izbremo $x \in POZ(C)$.

(M) Če je $\beta < P(C \mid x) < \alpha$ izberemo $x \in MEJ(C)$.

(C) Če je $P(C \mid x) \leq \beta$ izberemo $x \in \mathrm{NEG}(C)$.

V primeru poštnega filtra gre za verjetnost, da je pošta zaželena pri pogoju, da pošta vsebuje določene besede. Na podlagi izračunane verjetnosti, torej ali je večja verjetnost, da je pošta zaželena ali nezaželena, filter pošto ustrezno razvrsti. Poleg te osnovne metode bom preverjal različico, kjer filter razvršča v tri skupine. Pri tem načinu imamo tri skupine: zaželena, nezaželena in nerazvrščena.

Vzemimo za primer naslednji scenarij. Na podlagi verjetnosti, da je pošta neželena jo lahko razvrstimo v tri skupine. In sicer, če je verjetnost med 0 in 0,2 spada pošta v neželeno skupino, če je verjetnost med 0,2 in 0,8 spada v kategorijo nerazvrščeno, v primeru, da je verjetnost med 0,8 in 1 gre pošta v želeno pošto.

3 METODOLOGIJA

3.1. PREVERJANJE USPEŠNOSTI

V osnovi se uspešnost metode meri tako, da se preverja ustreznost razvrstitve sporočila. Stvar vseeno ni tako preprosta, kar lahko jasno vidimo na dveh primerih napačne razvrstitve. V enem primeru imamo napačno razvrstitev, kjer je bilo nezaželeno sporočilo razvrščeno med želena sporočila. V drugem primeru je filter razvrstil zaželeno sporočilo med neželena sporočila. Lahko rečemo, da je v prvem primeru napaka manjša, kot v drugem primeru. Za uporabnika predstavlja večji strošek izgubljena želena pošta v primerjavi z stroškom brisanja neželene pošte (Zhou in drugi, 2014).

Pri problemih razvrščanja se navadno uporabljata dve meri uspešnosti: natančnost in stopnja napake.

Zaradi večjega stroška napake v primeru izgube zaželenega sporočila moramo uporabiti uteženi meri uspešnosti. Utež pomeni koliko večji strošek je razvrstitev zaželenega sporočila v nezaželeno pošto. *W* predstavlja utež, v nalogi bom uporabil tri različne vrednosti uteži (1, 3, 9). V naslednjih enačbah uporabljam sledeče znake:

- $n_L \rightarrow L$- število pravilno razvrščenih zaželenih sporočil

- $n_S \rightarrow S$- število pravilno razvrščenih nezaželenih sporočil

- $n_L \rightarrow S$- število nepravilno razvrščenih zaželenih sporočil

- $n_S \rightarrow L$- število nepravilno razvrščenih nezaželenih sporočil

- NL- število vseh zaželenih sporočil

- NS- število vseh nezaželenih sporočil

Utežena natančnost:

$$WAcc = \frac{w \cdot n_{L\to L} + n_{S\to S}}{w \cdot N_L + N_S} \tag{34}$$

Utežena stopnja napake:

$$WErr = \frac{w \cdot n_{L\to S} + n_{S\to L}}{w \cdot N_L + N_S} \tag{35}$$

Poleg omenjenih uteženih mer uspešnosti bom uporabil tudi osnovno mero, kjer filter ni prisoten. Torej vsa zaželena sporočila dosežejo uporabnika, prav tako vsa nezaželena sporočila dosežejo uporabnika.

Utežena osnovna stopnja napake:

$$WErr^b = \frac{N_s}{w \cdot N_L + N_S} \tag{36}$$

Z mero uteženе osnovne stopnje napake in uteženе stopnje napake lahko izračunam skupno stroškovno razmerje. To je strošek časa, ki ga uporabnik porabi za ročno brisanje vse nezaželene pošte brez uporabe filtra, v primerjavi z časom brisanja nezaželene pošte in časom, da uporabnik povrne zaželeno pošto (Androutsopoulos in drugi, 2000).

Skupno stroškovno razmerje:

$$TCR = \frac{WErr^b}{WErr} = \frac{N_s}{w \cdot n_{L\to S} + n_{S\to L}} \tag{37}$$

Poleg tega se bom poslužil še dveh mer, ki se uporabljata pri merjenju uspešnosti poštnih filtrov. To sta priklic nezaželene pošte (angl. *spam recall*) in natančnost (angl. *spam precision*). Priklic je število pravilno razvrščenih nezaželenih sporočil v primerjavi z nezaželenimi sporočili, ki so bila pravilno razvrščena in nezaželenimi sporočili, ki so bila napačno razvrščena.

Priklic:

$$priklic = \frac{n_{S \to S}}{n_{S \to S} + n_{L \to L}} \tag{38}$$

Preciznost so nezaželena sporočila, ki so bila pravilno razvrščena v nezaželeno pošto v primerjavi z zaželenimi sporočili, katera so bila razvrščena v nezaželeno pošto in nezaželenimi sporočili, katera so bila razvrščena v nezaželeno pošto (Zhou in drugi, 2014).

Preciznost:

$$preciznost = \frac{n_{S \to S}}{n_{S \to S} + n_{L \to S}} \tag{39}$$

Strošek (angl. *cost*) meri koliko je uporabnika stalo razvrščanje glede na njegove stroškovne preference. Uporabil bom dve različni formuli za računanje stroška, saj preverjamo dva načina razvrščanja. Formula namreč temelji na uporabnikovi stroškovni tabeli.

V primeru dvo-skupinskega razvrščanja dobim tabelo 3.1.

Tabela 3.1 : Razvrščanje v dve skupini glede na zaželenost sporočila in odločitev filtra

odločitev filtra	C(P):pozitivno	C^c(N):negativno
a_P - sprejme	n_{PP}^D	n_{PN}^D
a_N - zavrne	n_{NP}^D	n_{NN}^D

n_{PP}^D – pravilno razvrščena zaželena sporočila

n_{NP}^D – nepravilno razvrščena zaželena sporočila

n_{PN}^D – pravilno razvrščena nezaželena sporočila

n_{NN}^D – nepravilno razvrščena nezaželena sporočila

^{d} - razvrščanje v dve skupini

U – število sporočil

Strošek za dvo-skupinsko razvrščanje:

$$cost^{\,d} = \frac{1}{U}\left[\left(\lambda_{PP}^d\, n_{PP}^d + \lambda_{PN}^d\, n_{PN}^d\right) + \left(\lambda_{NP}^d\, n_{NP}^d + \lambda_{NN}^d\, n_{NN}^d\right)\right] \tag{40}$$

V primeru razvrščanja v tri skupine uporabimo spodnjo tabelo (3.2).

Tabela 3.2: : Razvrščanje v dve skupini glede na zaželenost sporočila in odločitev filtra

odločitev filtra	C(P):pozitivno	C^c(N):negativno
a_P - sprejme	n_{PP}^T	n_{PN}^T
a_M - odloži	n_{MP}^T	n_{MN}^T
a_N - zavrne	n_{NP}^T	n_{NN}^T

n_{PP}^t – pravilno razvrščena zaželena sporočila

n_{NP}^t – nepravilno razvrščena zaželena sporočila

n_{MP}^t – odložena zaželena sporočila

n_{MN}^t – odložena nezaželena sporočila

n_{PN}^t – pravilno razvrščena nezaželena sporočila

n_{NN}^t – nepravilno razvrščena nezaželena sporočila

^{t} - razvrščanje v tri skupini

Strošek za tri skupinsko razvrščanje:

$$cost^{\,t} = [(\lambda_{PP}^t\, n_{PP}^t + \lambda_{PN}^t\, n_{PN}^t) + (\lambda_{MP}^t\, n_{MP}^t + \lambda_{MN}^t\, n_{MN}^t) + (\lambda_{NP}^t\, n_{NP}^t + \lambda_{NN}^t\, n_{NN}^t)] \quad (41)$$

Glavna prednost merjenja stroška je ta, da vzame obzir napako in strošek nerazvrščenih sporočil (Zhao in drugi, 2013).

3.2. PODATKI

Za izvedbo naloge bom potreboval podatke, torej pošto, katero bo filter razvrščal. Uporabil bom pošto iz osebnega elektronskega poštnega naslova, ker že vsebuje relativno veliko število sporočil. Pošto bom ustrezno kategoriziral, da mi bo lahko služila pri preverjanju metod. Dobljeno pošto bom razdelil na dva dela. Prvi del bo služil učenju poštnega filtra. Pri učenju bo filter imel za vsako sporočilo podatek o njegovi zaželenosti. Drugi del je namenjen preverjanju uspešnosti poštnega filtra, v tej fazi filter nima podatka o zaželenosti, saj ga izračuna na podlagi formule obravnavane v prejšnjih poglavjih. Pošto sem zbiral od maja 2017 do septembra 2017. Nabralo se je preko 3000 sporočila od tega so bila večinoma zaželena sporočila (96%). Zato sem število zaželenih sporočil zmanjšal, da ustreza poročilu o številu nezaželenih sporočil v globalnem merilu (50%-60%). V tabeli 3.3 so podatki o sporočilih.

Tabela 3.3: Delež sporočil glede na zaželenost in fazo razvrščanja

	zaželeno		nezaželeno		
	n	%	n	%	skupaj
učenje	135	0,467128028	154	0,532872	289
preverjanje	140	0,463576159	162	0,536424	302

3.3. DELOVANJE FILTRA

Simulacija delovanja spletnega filtra je sestavljena iz dveh delov:

- UČENJE (na podlagi znanja o zaželenosti sporočila)

- RAZVRŠČANJE (na podlagi algoritma)

3.3.1 UČENJE

Prvi program je namenjen učenju spletnega filtra. To pomeni, da filter prebere posamezno sporočilo, tako da ga razdeli na posamezne besede. Besede dodaja v seznam besed, kjer se beleži njihova prisotnost v sporočilih. Gledamo torej, kolikokrat se beseda pojavi in v katerem sporočilu, se pravi, zaželenem ali nezaželenem. Opisane frekvence služijo za računanje verjetnosti nezaželenosti sporočila. Poleg frekvenc se za vsako besedo računa še indeks skupne informacije, ki služi kot metoda za zmanjševanje števila besed uporabljenih v sporočilu, saj meri njen doprinos pri računanju verjetnosti.

Indeks skupne informacije je podan s formulo (Androutsopoulos in drugi, 2000).

$$MI = log \frac{P(C|x)}{P(x) \cdot P(C)} \tag{39}$$

Simbol *C* predstavlja kategorijo razvrščanja (npr. zaželena pošta). Z simbolom *x* označujem posamezno besedo, ki je v sporočilu. Besede, ki imajo večji indeks skupne informacije, bodo izbrane za računanje verjetnosti zaželenosti sporočila. Na podlagi poizkušanja sem prišel do spoznanja, da že izbor 15 besed daje dobre rezultate. Za vsako izmed teh besed filter preveri indeks skupne informacije. Besede z najvišjim indeksom (prvih 15) se uporabijo pri računanju verjetnosti nezaželenosti sporočila. Ko filter prebere vsa sporočila namenjena učenju, imamo za vsako besedo izračunan indeks skupne informacije in verjetnost pojavljanja v nezaželenem sporočilu.

3.3.2 RAZVRŠČANJE

Za preverjanje uspešnosti filtra bom uporabil drug program, kjer simuliram prejemanje sporočil in razvrščanje na podlagi izbranega algoritma. Na koncu program preveri dejansko zaželenost sporočila in jo primerja z odločitvijo filtra, kar služi kot osnova za računanje izbranih kazalcev uspešnosti.

Podobno kot v prvem delu filter prebira sporočila besedo za besedo. V kolikor beseda obstaja v seznamu besed, jo filter uporabi za nadaljnje delo. Besede, ki jih filter ne pozna, se ne uporabijo pri računanju verjetnosti. Za vsako sporočilo imamo torej spisek besed, ki jih je filter prepoznal in so v določenem sporočilu.

Izračunana verjetnost služi kot podlaga za odločanje filtra. V nalogi bom preverjal tri različne stroškovne scenarije, tako bom prišel do treh različnih verjetnostnih meja za razvrščanje.

Vsako odločitev filtra o nezaželenosti oziroma zaželenosti sporočila se preverja z dejanskim stanjem zaželenosti sporočila. S tem dobimo podatke, ki jih uporabimo pri računanju uspešnosti filtra. Programa sta napisana v programskem jeziku Python (Python, b. d.), uporabil sem tudi knjižnico za relacijske baze (mySql, b. d.), s katero sem ustvaril bazo v kateri hranim podatke o posamezni besedi.

3.3 PRIPRAVA SPOROČIL:

Sporočila sem iz poštnega računa prenesel na trdi disk osebnega računalnika s pomočjo programa *Gmvault* (Gmvault, b.d.). Dobljena sporočila sem razdelil na dva enakomerna dela. Prvi del sporočil sem uporabil za učenje filtra, drugi za razvrščanje (preverjanje). Vsa sporočila sem moral pretvoriti v UTF8 format zato, da je lahko filter prebral vse znake v sporočilu. Sporočila, ki sem jih uporabil sem tudi ustrezno označil, tako da sem sporočilu imenu datoteke sporočila dodal oznako »S«, če je bilo nezaželeno in »S«, če je bilo zaželeno (npr. mail34534_H.txt). To je bilo potrebno zaradi učenja in zaradi preverjanja uspešnosti filtra.

3.4 OPIS ZGRADBE FILTRA:

V tem delu bom na delih izvorne kode opisal kako je filter zgrajen in kako deluje. Sestavljen je iz dveh programov, izmed katerih je eden namenjen učenju filtra, drugi razvrščanju sporočil.

3.4.1 UČENJE

Program najprej ustvari podatkovno bazo v katero bo shranjeval prebrane besede, njihove frekvence in izračunane verjetnosti. Podatkovna baza je ustvarjena z odprtokodnim orodjem »mySQL« za urejanje podatkovnih baz. Celoten del programa je na voljo v prilogi A.

Slika 3.1: Kreiranje podatkovne baze

```
1.  c.execute('''''CREATE TABLE IF NOT EXISTS DICTIONARY
2.          (ID INT PRIMARY KEY  NOT NULL,
3.           WORD     TEXT UNIQUE NOT NULL,
4.           N                INT NOT NULL,
5.           NS               INT NOT NULL,
6.           NL               INT NOT NULL,
7.           WM               INT NOT NULL,
8.           WIS              INT NOT NULL,
9.           WP               INT NOT NULL,
10.          MI               INT NOT NULL,
11.          WIH              INT NOT NULL);''')
```

V nadaljevanju določim mapo iz katere filter pobira sporočila za učenje. Sledi zanka, v kateri filter obdeluje vsako sporočilo posebej. Najprej preveri ali je sporočilo zaželeno ali ne. V fazi učenja mora filter vedeti za vsako sporočilo njegovo zaželenost, saj le tako lahko beleži frekvence v sporočilih.

Slika 3.2: Začetek zanke sporočil

```
emails = [os.path.join("/path/to/mails/train",f) for f in os.listdir("/path/to/mails/train")]

for mail in emails:

    mailcheck = mail[-5:-4]

    print "MAIL NUMBER:", number_mails

    if mailcheck == "H":
        number_ham_mails = number_ham_mails + 1
        actval = 0

    elif mailcheck == "S":
        number_spam_mails = number_spam_mails + 1
        actval = 1
    else:
        pass
```

Ustvaril sem še dve zanki, v eni filter razdeli sporočilo na vrstice, v drugi razdeli vrstice na besede. Ko dobim posamezne besede lahko filter odstrani določene znake, ki jih pri nadaljnjem delu ne potrebuje.

Slika 3.3: Zanka posameznih besed

```
with open(mail) as m:
    ##################### WORD PROCESSING ############################################
    ########

        for i,line in enumerate(m):
            words = line.split()
            for word in words:


                ############ INCREMENT WORD COUNTER ####################
                words_in_mail = words_in_mail + 1
                ##########################################################
                word = word.decode("utf-8")
                word = word.lower()


                word = re.sub(r'[^\w\s]','',word)
                ########### DELIMITER CONDITIONS ########################
                if word == ">" or word =="|" or word =="{" or word == "}" or word == ">>" or word == "[" or word == "]" \
                or word == "#" or word == "*" or word == "-" or word == "<a" or word == "/>" or word == "</tr>"\
                or word == "<td" or word == "--" or word == "<p" or word == "<TR>" or word =="<TD" or word == "<tr>" or word == "=":
                    continue
```

V spodnjem delu izvorne kode, je prikazan del program, kjer filter preverja ali je obravnavana beseda že v podatkovni bazi ali je še ni. V primeru, da beseda še ni shranjena, jo doda in zapišem še vse ostale statistike, ki jih za besede vodi.

Slika 3.4: Dodajanje besede v podatkovno bazo

```
1.  if test == 0:
2.
3.      if actval == 0:
4.          ns = 1
5.          nl = 2
6.          n = 2
7.          c.execute("INSERT OR IGNORE INTO DICTIONARY (ID,WORD,N,NS,NL, WM, WIS, WP, M
    I, WIH) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", (idn, word, n, ns, nl, 1, 0, 0, 0, 0
    ));
8.      else:
9.          ns = 2
10.         nl = 1
11.         n = 2
12.         c.execute("INSERT OR IGNORE INTO DICTIONARY (ID,WORD,N,NS,NL, WM, WIS, WP, M
    I, WIH) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", (idn, word, n, ns, nl, 1, 0, 0, 0, 0
    ));
13.     conn.commit()
14.     ### INDEX INCREMENT##
15.     idn = idn + 1
16.
17.     spamprob = float(number_spam_mails / number_mails)  ### PROBABILITY OF MAIL BEIN
    G A SPAM MAIL
18.     hamprob = float(number_ham_mails / number_mails)
19.     wordinspamprob = float(ns / n) #### PROBABILITY OF WORD BEING IN A SPAM MAIL
20.     wordinhamprob = float(nl / n) #### PROBABILITY OF WORD BEING IN A SPAM MAIL
21.     ######################## UPDATE PROB DATA ########################################

22.     c.execute("SELECT COUNT (*) FROM DICTIONARY")
23.     number_of_words_in_dictionary = c.fetchone()[0]
24.     ######################## MI INDEX CALC ######################
25.     try:
26.         mi = math.log(wordinspamprob/((2/number_of_words_in_dictionary)*spamprob))
27.         #mi = math.log((wordinspamprob)/(2 / number_of_words_in_dictionary))
28.     except Exception as e:
29.         mi = 0
30.     c.execute("UPDATE DICTIONARY SET WIS = ?, WIH = ?, MI = ? WHERE WORD=?", (wordin
    spamprob, wordinhamprob, mi, word))
31.     conn.commit()
32. else:
33.     pass
34. ################### mark that word is in message ##############################
```

Če je beseda, ki je v obravnavi, že v podatkovni bazi potem filter posodobi podatke o njeni prisotnosti v sporočilih in izračunane verjetnosti, da se le-teh pojavlja.

Filter potem nadaljuje z naslednjo besedo, dokler ne pride do zadnje besede v sporočilu. Ko ni več besed v sporočilu, nadaljuje z naslednjim sporočilom. Tako filter prebere vsa sporočila in doda besede v podatkovno bazo.

Slika 3.5: Posodabljanje frekvenc in verjetnosti besed v bazi

```
 if foc == 0:
     #c.execute("UPDATE DICTIONARY SET WM = ? WHERE WORD=?", (1, word))
     conn.commit()
     #### GET STATS FOR WORD#################################################################
     get_data  = c.execute("SELECT N, NS, NL FROM DICTIONARY WHERE WORD = ?", (word,))

     word_frequency = 0
     spam_word_frequency = 0
     ham_word_frequency = 0

     for data in get_data:

         word_frequency = data[0]
         spam_word_frequency = data[1]
         ham_word_frequency = data[2]


     ns = spam_word_frequency
     nl = ham_word_frequency
     n = word_frequency + 1


     if actval == 1:

         ns = ns + 1


     elif actval == 0:

         nl = nl + 1
     else:
         pass


     spamprob = float(number_spam_mails / number_mails)  ### PROBABILITY OF MAIL BEING A SPAM MAIL
     hamprob = float(number_ham_mails / number_mails)

     wordinspamprob = float(ns / n) #### PROBABILITY OF WORD BEING IN A SPAM MAIL
     wordinhamprob = float(nl / n) #### PROBABILITY OF WORD BEING IN A SPAM MAIL


     c.execute("SELECT COUNT (*) FROM DICTIONARY")
     number_of_words_in_dictionary = c.fetchone()[0]
     ######################### MI INDEX CALC ######################

     try:
         mi = math.log(wordinspamprob/((n/number_of_words_in_dictionary)*spamprob))
         #mi = math.log((wordinspamprob)/(n/number_of_words_in_dictionary))
     except Exception as e:
         mi = 0

     c.execute("UPDATE DICTIONARY SET N = ?, NS = ?, NL = ?, WIS = ?, WIH = ?, MI = ?,  WM = ? WHERE WORD=?", (n, ns, nl, wordinspamprob, wordinhamprob, mi, 1,  word))
     ##################################################################
     conn.commit()
 else:
```

3.4.3 RAZVRŠČANJE

Na podlagi učenja, ki ga je opravil v prejšnji fazi, filter nadaljuje z razvrščanjem. Podobno, kot v prejšnjem delu filter obdeluje vsako sporočilo in besedo posebej. Na ta način dobi seznam vseh besed, ki so v sporočilu in jih ima shranjene v podatkovni bazi. Celoten program je na voljo v prilogi B.

Slika 3.6: Prikaz zanke v kateri filter obdeluje posamično besedo

```
1.  with open(mail) as m:
2.   ##################### WORD PROCESSING ##############################################
    ####
3.
4.       for i,line in enumerate(m):
5.           words = line.split()
6.           for word in words:
7.
8.               word = word.decode("utf-8")
9.               word = word.lower()
10.
11.
12.              word = re.sub(r'[^\w\s]','',word)
13.
14.
15.              count = c.execute("SELECT count(*) FROM DICTIONARY WHERE WORD = ?", (wo
    rd,))
16.
17.              for row in count:
18.                  test = row[0]
19.
20.
21.              if test == 1:
22.
23.                  get_data = c.execute("SELECT WM FROM DICTIONARY WHERE WORD = ?", (w
    ord,))
24.                  foc = 0
25.                  for data in get_data:
26.                      foc = data[0]
27.
28.                  if foc == 0:
29.                      c.execute("UPDATE DICTIONARY SET WM = ? WHERE WORD=?", (1, word
    ))
30.                      conn.commit()
31.                  else:
32.                      pass
33.
34.
35.              else:
36.                  pass
```

Seznam besed, ki so v sporočilu in v bazi omeji na manjše število, tako, da vzame 15 besed, ki so po indeksu skupne informacije najviše uvrščene. Na podlagi dobljenega izbora besed lahko filter izračuna verjetnost, da je sporočilo zaželeno.

Slika 3.7: Izračun verjetnosti zaželenosti sporočila

```
1.  get_data  = c.execute("SELECT WIS, WIH, WORD FROM DICTIONARY  WHERE WM = ? ORDER BY
    MI DESC LIMIT 15 ", (1,))
2.      #get_data  = c.execute("SELECT WIS, WIH FROM DICTIONARY  WHERE WM = ?", (1,))
3.
4.      all_words_in_spam_probs = 1
5.      all_words_in_ham_probs = 1
6.      all_words_list = 1
7.
8.      mi_list_prob_spam = []
9.      mi_list_prob_ham = []
10.     word_list = []
11.
12.     for data in get_data:
13.
14.         all_words_in_spam_probs = data[0]
15.         all_words_in_ham_probs = data[1]
16.         all_words_list = data[2]
17.
18.         if all_words_in_spam_probs == 0:
19.             all_word_in_spam_probs = 1
20.         else:
21.             pass
22.
23.         if all_words_in_ham_probs == 0:
24.             all_words_in_ham_probs = 1
25.         else:
26.             pass
27.
28.         mi_list_prob_spam.append(all_words_in_spam_probs)
29.         mi_list_prob_ham.append(all_words_in_ham_probs)
30.         word_list.append(all_words_list)
31.
32.
33.     result_joint_prob_spam = np.prod(np.array(mi_list_prob_spam))
34.     result_joint_prob_ham = np.prod(np.array(mi_list_prob_ham))
35.
36.
37.
38.     total_prob = (hamprob * result_joint_prob_ham) / ((result_joint_prob_spam * spam
    prob) + (result_joint_prob_ham * hamprob))
```

Sledi del, kjer filter na podlagi dobljenih verjetnosti filter razvrsti sporočila v različne kategorije. Izračunano verjetnost primerja z verjetnostnimi pragovi in tako določi v katero skupino gre sporočilo. Pri dvo-skupinskem razvrščanju primerja verjetnost z enim pragom, pri tri-skupinskem razvrščanju z dvema pragovoma. Z spreminjanjem vrednosti verjetnostnih pragov in vrednosti uteži *w* sem simuliral različne uporabnike spletne pošte. Z spreminjanjem stroškovnih preferenc se spreminjajo meje verjetnosti razvrščanja.

Slika 3.8: Določanje verjetnostnih pragov

```
######## TWO WAY ####################

    if total_prob >= 0.8889:
     predval = 0

    elif total_prob < 0.8889:
     predval = 1

    else:
     pass


    ######## THREE_WAY ##################

    #if total_prob > 0.0123 and total_prob < 0.8889:
    #    predval = 2
    #elif total_prob <= 0.0123:
    #    predval = 1
    #elif total_prob >= 0.8889:
    #    predval = 0
    #else:
    #    pass
```

V zadnjem delu ima filter nalogo, da na podlagi razvrščenih sporočil izračuna vse mere uspešnosti, ki sem jih v namen raziskave določil.

Slika 3.9: Izračun uspešnosti

```
1.  if predval == 1 and actval == 1:            ### HIT SPAM ###
2.       nss = nss + 1
3.       counthit = counthit + 1
4.
5.
6.      elif predval == 0 and actval == 0:          ### HIT HAM ###
7.       nll = nll + 1
8.       counthit = counthit + 1
9.
10.
11.     elif predval == 0 and actval == 1:          ### MISS SPAM ###
12.      nsl = nsl + 1
13.      countmiss = countmiss + 1
14.
15.
16.     elif predval == 1 and actval == 0:          ### MISS HAM ###
17.      nls = nls + 1
18.      countmiss = countmiss + 1
19.
20.     elif predval == 2:
21.         nm = nm + 1
22.     else:
23.      pass
24.
25.
26.
27.     w = 9
28.
29.     try:
30.         wacc = ((nll*w) + nss) / ((number_ham_mails* w) + number_spam_mails)
31.     except Exception as e:
32.         print "division by zero1"
33.
34.     try:
35.         werr = ((nls*w) + nsl) / ((number_ham_mails* w) + number_spam_mails)
36.     except Exception as e:
37.         print "division by zero2"
38.     try:
39.         werrb = (number_spam_mails) / ((number_ham_mails* w) + number_spam_mails)
40.     except Exception as e:
41.         print "division by zero3"
42.     try:
43.         tcr = werrb / werr
44.     except Exception as e:
45.         print "division by zero4"
46.     try:
47.         recall = (nss)/ (nss + nsl)
48.     except Exception as e:
49.         print "division by zero5"
50.     try:
51.         precision = (nss) / (nss + nls)
52.     except Exception as e:
53.         print "division by zero6"
54.
55.
56.
57.     file1.write("%s\n" %wacc)
58.     file2.write("%s\n" %werr)
59.     file3.write("%s\n" %werrb)
60.     file4.write("%s\n" %tcr)
61.     file5.write("%s\n" %recall)
62.     file6.write("%s\n" %precision)
63.     file12.write("%s %s\n" %(actval, total_prob))
```

4 REZULTATI

4.1 RAČUNANJE MEJNIH VREDNOSTI VERJETNOSTI

V nalogi preverjam kako se obnese tri-skupinsko razvrščanje v primerjavi z dvo-skupinskim razvrščanjem v stroškovno občutljivih situacijah.

V treh scenarijih bom uporabil različne stroške razvrščanja. V ta namen moram določiti strošek vsake možne odločitve razvrščanja. Pri tem moram upoštevati nekatere omejitve, da lahko pridem do smiselnih meja verjetnosti. Veljati morajo naslednji pogoji, da lahko dobim parametre z vrednostmi med 0 in 1.

Na podlagi formule

$$\alpha = \frac{(\lambda_{PN} - \lambda_{MN})}{(\lambda_{PN} - \lambda_{MN}) + (\lambda_{MP} - \lambda_{PP})}$$

sledi,

$\lambda_{PN} > \lambda_{MN}$.

(40)

Podobno sledi iz formule

$$\beta = \frac{(\lambda_{MN} - \lambda_{NN})}{(\lambda_{MN} - \lambda_{NN}) + (\lambda_{NP} - \lambda_{MP})}$$

sledi,

$$\lambda_{MN} > \lambda_{NN}.$$

(41)

Z upoštevanjem zgornjih pogojev lahko določim stroške različnih odločitev pri razvrščanju. Lahko si predstavljamo, da imamo tri različne uporabnike spletne pošte, kateri imajo različen odnos do prejemanja pošte. Vrednosti stroškov uporabnikov se razlikujejo v primeru napačnega razvrščanja. Ostale vrednosti so pri vseh uporabniki iste.

Prvemu uporabniku predstavlja napačno razvrščena zaželena pošta isti strošek kot napačno razvrščena zaželena pošta. Iz tabele 4.1 je to jasno razvidno, če pogledamo vrednost celice, ko je odločitev filtra »zavrnitev« in stanje sporočila »zaželeno«. Poštni filter se je napačno odločil tako, da je zavrnil zaželeno sporočilo. Poglejmo še vrednost celice, ko filter sporočilo sprejme, a je stanje sporočila »nezaželeno«. Opazimo, da je vrednost v obeh celicah tri. Uporabniku ta dva primera predstavljata isti strošek.

Tabela 4.1: Strošek napačne razvrstitve zaželenega sporočila je enak strošku razvrstitve nezaželenega sporočila.

odločitev filtra	stanje sporočila	
	Zaželeno	nezaželeno
$a_{P\ –\ sprejme}$	0	3
$a_{M\ –\ odloži}$	1	1
$a_{N\ –\ zavrne}$	3	0

Ker imam določene stroške za vse možne situacije pri razvrščanju lahko izračunam mejne vrednosti verjetnosti.

$$\alpha = \frac{(\lambda_{PN}-\lambda_{MN})}{(\lambda_{PN}-\lambda_{MN})+(\lambda_{MP}-\lambda_{PP})} \qquad \beta = \frac{(\lambda_{MN}-\lambda_{NN})}{(\lambda_{MN}-\lambda_{NN})+(\lambda_{NP}-\lambda_{MP})}$$

$$\alpha = \frac{(3-1)}{(3-1)+(1-0)} \qquad \beta = \frac{(1-0)}{(1-0)+(3-1)}$$

$$\alpha = \frac{2}{3} \qquad \beta = \frac{1}{3}$$

$$\alpha = 0.6667 \qquad \beta = 0.3333$$

Filter lahko na podlagi izračunanih vrednosti parametrov razvršča pošto v tri skupine.

Če je verjetnost sporočila, večja ali enaka od 0.6667 gre sporočilo v zaželeno pošto.

Če je verjetnost med 0.6667 in 0.3333 gre sporočilo v nerazvrščeno pošto. Uporabnik mora sam določiti ali je sporočilo zaželeno ali nezaželeno.

V primeru, da je vrednost manjša ali enaka 0.3333 gre sporočilo v nezaželeno pošto.

Podobno kot za prvega uporabnika moramo določiti stroškovne preference za drugega uporabnika. Predstavljeni so v tabeli 4.2.

Tabela 4.2: Strošek napačne razvrstitve zaželenega sporočila trikrat večji od stroška napačne razvrstitve nezaželenega sporočila.

odločitev filtra	stanje sporočila	
	Zaželeno	nezaželeno
$a_{P\ -\ sprejme}$	0	6
$a_{M\ -\ odloži}$	1	1
$a_{N\ -\ zavrne}$	18	0

Dobim vrednosti parametrov:

Parameter $\alpha = 0{,}8333$ in parameter $\beta = 0.5556$

V tabeli 4.3 lahko vidimo vrednosti stroškov za tretjega uporabnika.

Tabela 4.3: strošek napačne razvrstitve zaželenega sporočila je devetkrat večji od stroška napačne razvrstitve nezaželenega sporočila.

odločitev filtra	stanje sporočila	
	Zaželeno	nezaželeno
$a_{P\ -\ sprejme}$	0	9
$a_{M\ -\ odloži}$	1	1
$a_{N\ -\ zavrne}$	81	0

Izračunana parametra:

$\alpha = 0.8889\ \beta = 0.0123$

4.2 REZULTATI RAZVRŠČANJA

V spodnjem delu bom predstavil rezultate razvrščanje glede na tip razvrščanja in pravilnost razvrstitve.

Tabela 4.4: Deleži razvrščenih sporočil glede na tip razvrščanja

Utež	Tip razvrščanja	Verjetnostne meje	$n_L \rightarrow L$	$n_S \rightarrow S$	$n_L \rightarrow S$	$n_S \rightarrow L$	n_M
1	Tri-skupinsko	α = 0.6667, β = 0.3333	36,42%	50,66%	8,28%	2,32%	2,32%
	Dvo-skupinsko	α = 0.6667	36,42%	51,32%	9,93%	2,32%	0,00%
3	Tri-skupinsko	α = 0.8333, β = 0.5556	32,78%	51,32%	9,60%	1,66%	4,64%
	Dvo-skupinsko	α = 0.8333	32,78%	51,99%	13,58%	1,66%	0,00%
9	Tri-skupinsko	α = 0.8889, β = 0.0123	31,79%	45,36%	4,30%	1,32%	17,22%
	Dvo-skupinsko	α = 0.8889	31,79%	52,32%	14,57%	1,32%	0,00%

$n_L \rightarrow L$ - pravilno razvrščena zaželena sporočila

$n_S \rightarrow S$ - pravilno razvrščena nezaželena sporočila

$n_L \rightarrow S$ - nepravilno razvrščena zaželenih sporočil

$n_S \rightarrow L$ - nepravilno razvrščena nezaželena sporočila

n_M - nerazvrščena sporočila (mejno območje)

Iz tabele 4.4 je razvidno, kako je filter razvrščal sporočila glede na tip razvrščanja. Pri prvem uporabniku, kjer ima utež vrednost ena, lahko vidimo, da je odstotek pravilno razvrščenih nezaželenih sporočil (51,32%) višji pri dvo-skupinskem razvrščanju. Odstotek pravilno razvrščenih zaželenih sporočil (36,42%) je enak pri obeh tipih razvrščanja. Oba tipa imata isto zgornjo verjetnostno mejo. Pri nepravilno razvrščenih zaželenih sporočilih lahko opazimo, da je odstotek nižji pri tri-skupinskem razvrščanju (8,28%). Medtem, ko je odstotek nepravilno razvrščenih nezaželenih sporočil isti (2,32%). V zadnjem stolpcu lahko vidimo, da ima tri-skupinsko razvrščanje vrednost 2,32%. Filter je torej, dobra dva odstotka sporočil umestil v nerazvrščeno kategorijo in tako znižal število napačno razvrščenih nezaželenih sporočil.

Na primeru, kjer ima utež vrednost tri, lahko opazimo podobne rezultate. Odstotek pravilno razvrščenih zaželenih sporočil je isti pri obeh tipih razvrščanja (32,78%). Pri pravilno razvrščenih nezaželenih sporočilih je višje uvrščeno dvo-skupinsko razvrščanje (51,99%).

Večja razlike med načinom razvrščanja se pokaže pri nepravilno razvrščenih zaželenih sporočilih. Filter je napačno razvrstil za štiri odstotke (13,58%) več sporočil kot pa pri tri-skupinskem razvrščanju. Odstotek nepravilno razvrščenih nezaželenih sporočil je isti pri obeh tipih (1,66%). Pri tri-skupinskem razvrščanju je filter 4,64% sporočil uvrstil v kategorijo nerazvrščeno.

Poglejmo še deleže razvrščenih sporočil, ko ima utež vrednost devet. Ponovno imata oba tipa razvrščanja isti odstotek pravilno razvrščenih zaželenih sporočil. Pri pravilno je razvrščenih nezaželenih je opaziti razliko, saj ima dvo-skupinsko razvrščanje višjo vrednost (52,32%). Opazimo lahko tudi, da je pri dvo-skupinskem razvrščanju filter naredil za skoraj deset odstotkov več napak (14,57%), ko je razvrščal zaželena sporočila. Pri razvrščanju nezaželenih sporočil ni razlik. Filter je pri tri-skupinskem razvrščanju uvrstil v kategorijo nerazvrščeno 17,22% sporočil.

Vidimo lahko, da tri-skupinsko razvrščanje zmanjšuje število napačno razvrščenih zaželenih sporočil. Večja kot je utež, večje postanejo razlike med tipom razvrščanja. Uvedba nerazvrščene pošte prepreči, možnost napake, ki bi za uporabnika predstavljala velik strošek.

Tabela 4.5 prikazuje rezultate razvrščanja glede na mere uspešnosti in vrsto razvrščanja.

Tabela 4.5: Rezultati razvrščanja glede na mere uspešnosti

Utež	Tip razvrščanja	Verjetnostne meje	Strošek	Natančnost	Priklic	TCR	Wacc	Werr	Werrb
	Tri-skupinsko	$\alpha = 0.6667, \beta = 0.3333$	0,34	0,85	0,94	4,50	0,88	0,12	0,54
1	Dvo-skupinsko	$\alpha = 0.6667$	0,37	0,85	0,94	4,50	0,88	0,12	0,54
	Tri-skupinsko	$\alpha = 0.8333, \beta = 0.5556$	0,04	0,89	0,97	2,75	0,42	0,10	0,28
3	Dvo-skupinsko	$\alpha = 0.8333$	2,54	0,85	0,94	1,80	0,46	0,15	0,28
	Tri-skupinsko	$\alpha = 0.8889, \beta = 0.0123$	2,11	0,91	0,97	1,34	0,16	0,09	0,11
9	Dvo-skupinsko	$\alpha = 0.8889$	11,92	0,85	0,94	0,64	0,19	0,18	0,11

Vidimo, da pri uporabniku, ki ne loči med napačnim razvrščanjem ne pride do razlik med dvo-skupinskim in tri-skupinskim razvrščanjem. Opaziti je razliko pri strošku razvrščanja, kjer ima tri-skupinsko razvrščanje manjšo vrednost.

Pri drugem uporabniku se že pojavljajo razlike v uspešnosti. Tri-skupinsko razvrščanje predstavlja manjši strošek za uporabnika. Natančnost razvrščanja je večja (0,89). Vrednost priklica je prav tako višja (0,97). Kazalec, ki meri skupno stroškovno razmerje ima višjo vrednost pri tri-skupinskem razvrščanju (2.75). Pri kazalcih Werr, Werrb in Wacc so vrednosti višje pri dvo-skupinskem razvrščanju.

Pri zadnjem uporabniku prihaja do še večjih razlik med načinoma razvrščanja. Strošek razvrščanja pri tri-skupinskem razvrščanju (2,11) je bistveno manjši v primerjavi z dvo-skupinskim razvrščanjem. Kazalec natančnosti (0,91) kaže višjo vrednost pri tri-skupinskem razvrščanju. Tako je tudi pri priklicu, kjer je vrednost kazalca 0,97. Bolj učinkovito razvrščanje se izrazi tudi pri skupnem stroškovnem razmerju, kjer tri-skupinsko razvrščanje ponovno doseže višjo vrednost (1,34). Vrednost utežene natančnosti je višja pri dvo-skupinskem razvrščanju (0,16). Nižjo vrednost utežene stopnje napake ima tri-skupinsko razvrščanje (0,09).

Lahko rečemo, da je tri-skupinsko razvrščanje bolj uspešno od dvo-skupinskega razvrščanja glede na izbrane mere uspešnosti. To se še bolj izrazito kaže v primerih, kjer je uporabnikovo vrednotenje stroška napačne razvrstitve sporočila visoko. Strošek razvrščanja je nižji pri tri-skupinskem razvrščanju, saj z možnostjo odložitve sporočila zmanjša napako razvrščanja. Z večanjem uteži se vrednosti utežene stopnje napake, pri obeh tipih razvrščanja, manjšajo. Podobno je tudi z skupnim stroškovnim razmerjem. Natančnost se z večanjem uteži pri tri-skupinskem razvrščanju veča.

5 SKLEP

Na podlagi rezultatov, ki so prikazani v prejšnjem poglavju ugotavljam, da je razvrščanje v tri skupine je bolj uspešno kot razvrščanje v dve skupini. Pri cenovno občutljivih merah uspešnosti je razlika med načinoma še večja. Višji kot je strošek napačne razvrstitve, boljše se izkaže metoda tri-skupinskega razvrščanja. Rezultati so skladni z dosežki raziskave, ki je služila kot eno od izhodišč tega dela (Zhou in drugi, 2014). Z uvedbo nove skupine razvrščanja se lahko izognemo primerom, ko filter zaželeno pošto zavrne. Za uporabnika je to zelo dobrodošlo, saj je uporaba spletne pošte tako bolj učinkovita. Ugotovil sem tudi, da lahko z uporabo Bayesove teorije odločanja in uporabnikovih stroškovnih preferenc filter avtomatično nastavi vrednosti verjetnostnih pragov. Poleg že omenjenega je pomemben del naloge predstavljala izdelava programa poštnega filtra. S filtrom sem v osnovni obliki prikazal delovanje algoritma za razvrščanje. Z izdelavo lastnega programa sem imel dovolj svobode, da sem lahko izvedel poizkus. Seveda je bil filter omejen, samo na delovanje na lokalnem računalniku in ni bil vgrajen v spletno pošto. Za učenje in razvrščanje sem uporabil sporočila prejeta v lastnem poštnem računu, zato je število le-teh relativno majhno. Za preverjanje uspešnosti algoritmov, se navadno uporablja večji korpus sporočil. Razen implementacije preučevanega algoritma pri izdelavi filtra nisem posvečal velike pozornosti optimizaciji ostalih parametrov, ki lahko vplivajo na bolj uspešno razvrščanje. Eden izmed takih pristopov bi lahko bilo tudi učenje filtra v fazi razvrščanja. V obstoječi obliki se filter ne prilagaja novim sporočilom, saj ne posodablja podatkovne zbirke. Tak filter je potrebno vedno znova učiti, da lahko sledi novim sporočilom. Z sprotnim učenjem, torej med samim razvrščanjem, bi lahko s pomočjo uporabnika filter ostajal svež in pripravljen za nova sporočila. To je ena izmed idej, ki se mi je med izdelavo naloge porodila do te mere, da sem jo celo na hitro preizkusil. Izidi so bili obetavni. Vsekakor ostaja še veliko prostora za nadaljnje raziskave na področju strojnega učenja. Menim, da je raba učinkovitih algoritmov združena z računsko močjo strojev dobra kombinacija za ustvarjanje orodij, ki nam lahko služijo pri reševanju težav.

6 VIRI

1. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G. in Spyropoulos, C.D. (2000). An evaluation of naive Bayesian anti-spam filtering. G. Potamias, V. Moustakis, M. van Someren (ur.), *Proceedings of the workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning* (str. 9–17). Barcelona: Springer Berlin Heidelberg. Dostopno prek https://arxiv.org/pdf/cs/0006013.pdf

2. Symantec Corporation. (2017, april). *Internet Security Threat Report.* Dostopno prek: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf.

3. Arnes. (b. d.). *Neželena elektronska pošta*. Dostopno prek https://www.varninainternetu.si/2013/spam-mesni-narezek-v-vasem-e-postnem-nabiralniku/

4. Graham, P. (2002, avgust). *A plan for spam*. Dostopno prek http://www.paulgraham.com/spam.html

5. Zhou, B., Yao, Y. & Luo, J. (2014). Cost-sensitive three-way email spam filtering. *Journal of Intelligent Information Systems*, *42*(1), 19–45. https://doi.org/10.1007/s10844-013-0254-7

6. NetZero, Inc. (2018). *Different types of spam.* Dostopno prek https://help.netzero.net/nzhelp/support-home-page/service-help/online-security/security-tips/different-types-of-spam/

7. Pawlak, Z. (1981). Rough sets. *International Journal of Computer & Information Sciences*, *11*(5), 341-356. Dostopno prek https://doi.org/10.1007/BF01001956

8. SI-Cert. (b. d.). *Zakonodaja in spam*. Dostopno prek https://www.cert.si/si/varnostne-groznje/spam/zakonodaja-in-spam/

9. Protocol labs. (b. d.). *History of email spam*. Dostopno prek https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/History_of_email_spam.html

10. Chiarella, J. in O'Brien, J. (2003). *An Analysis of Spam Filters* (delovno poročilo, CS-CEW-0203). Dostopno prek: https://www.researchgate.net/profile/Chaouki_Hannachi/post/Which_are_the_best_spam_datasets/attachment/59d6449679197b807799fd4c/AS:449334262145024@1484141044192/download/spam+filter.pdf.

11. Puri, S., Gosain, D., Ahuja, M., Kathuria, I., Jatana, N. (2013, April). Comparison and Analysis of Spam Detection Algorithms. *International journal of Application or innovation in Engineering & Management*. *2*(4). Dostopno prek http://www.ijaiem.org/Volume2Issue4/IJAIEM-2013-04-02-005.pdf

12. Ferligoj, A. (1997). *Osnove statistike na prosojnicah*. Ljubljana: samozal. Z. Batagelj.

13. Gmvault Gmail backup [Programska oprema]. (2018). Dostopno prek http://gmvault.org/.2018

14. SpamBayes: Bayesian anti-spam classifier written in Python (različica 1.04.) [Programska oprema]. (b. d.). Dostopno prek http://www.spambayes.sourceforge.net/

15. Apache SpamAssasin. (različica 3.4.1) [Programska oprema]. (2018). Dostopno prek: https://spamassassin.apache.org/

16. Bogofilter (različica 1.2.4) [Programska oprema]. (b. d.). Dostopno prek: http://bogofilter.sourceforge.net/

17. Yao, Y. (2007) Decision-Theoretic Rough Set Models. J., Yao, P., Lingras, WZ., Wu, M., Szczuka, N.J., Cercone, D., Ślęzak (ur.), *Lecture Notes in Computer Science: zv. 4481, Rough Sets and Knowledge Technology* (str. 1–12). Dostopno prek https://doi.org/10.1007/978-3-540-72458-2_1

18. Kolmogorov, A.N. (1956). *Foundations of the theory of probability*. New York: Chelsea Publishing Company. Dostopno prek https://www.york.ac.uk/depts/maths/histstat/kolmogorov_foundations.pdf

19. Papoulis, A. in Pillai, S. U. (2002). *Probability, random variables, and stochastic processes*. Boston: McGraw-Hill.

20. Duda, R.O., Hart, P. E. in Stork, D. G. (2000). *Pattern Classification*. Wiley Interscience.

21. MySql (b. d.). Dostopno prek https://www.mysql.com/

22. Python (b. d.). Dostop prek https://www.python.org/

PRILOGI

Priloga A: Učenje

```
1.  from __future__ import division
2.  import os
3.  from collections import Counter
4.  import pprint
5.  import itertools
6.  import pandas as pd
7.  import numpy as np
8.  import sqlite3
9.  import math
10. from heapq import nlargest
11. import nltk
12. import re
13. import time
14.
15. start_time = time.time()
16.
17.
18.
19. all_word_probs = 1
20. joint_prob = 1
21. joint_word_prob = 1
22.
23. wordinspamprob = 0
24. wordprob = 0
25. spamprob = 0
26.
27.
28. total_prob = 0
29.
30. words_in_mail = 0
31. number_mails = 1
32. number_spam_mails = 0
33. number_ham_mails = 0
34.
35. aggregate_total_prob_spam = 0
36. aggregate_total_prob_ham = 0
37. lower_dyn_thresh = 0
38. upper_dyn_thresh = 0
39.
40. n = 0
41. nl = 0
42. ns = 0
43. idn = 1
44.
45. mi = 0
46. all_mis = 0
47. inp = 0
48. mi_list = []
49. mi_sorted = []
50. top_mi = 0
51.
52. predval = 0
53. actval = 0
54. countmiss = 0
55. counthit = 0
56. undefined = 0
57. hitspam = 0
```

```
58. hitham = 0
59. missspam = 0
60. missham = 0
61.
62. conn = sqlite3.connect("test.db")
63. c = conn.cursor()
64. count_words = 0
65.
66.
67.
68.
69. c.execute('''''CREATE TABLE IF NOT EXISTS DICTIONARY
70.         (ID INT PRIMARY KEY  NOT NULL,
71.          WORD     TEXT UNIQUE NOT NULL,
72.          N                INT NOT NULL,
73.          NS               INT NOT NULL,
74.          NL               INT NOT NULL,
75.          WM               INT NOT NULL,
76.          WIS              INT NOT NULL,
77.          WP               INT NOT NULL,
78.          MI               INT NOT NULL,
79.          WIH              INT NOT NULL);''')
80.
81. emails = [os.path.join("/path/to/mails/",f) for f in os.listdir("/path/to/mails/train")]
82.
83. for mail in emails:
84.
85.     mailcheck = mail[-5:-4]
86.
87.     print "MAIL NUMBER:", number_mails
88.
89.     if mailcheck == "H":
90.         number_ham_mails = number_ham_mails + 1
91.         actval = 0
92.
93.     elif mailcheck == "S":
94.         number_spam_mails = number_spam_mails + 1
95.         actval = 1
96.     else:
97.         pass
98.
99.     with open(mail) as m:
100.    ##################### WORD PROCESSING ############################################
    #######
101.
102.        for i,line in enumerate(m):
103.            words = line.split()
104.            for word in words:
105.
106.
107.                ############ INCREMENT WORD COUNTER ####################
108.                words_in_mail = words_in_mail + 1
109.                ############################################################
110.                word = word.decode("utf-8")
111.                word = word.lower()
112.
113.
114.                word = re.sub(r'[^\w\s]','',word)
115.                ############ DELIMITER CONDITIONS ########################
116.                if word == ">" or word =="|" or word =="{" or word == "}" or word ==
    ">>" or word == "[" or word == "]" \
117.                or word == "#" or word == "*" or word == "-
    " or word == "<a" or word == "/>" or word == "</tr>"\
```

```
                or word == "<td" or word == "--
    " or word == "<p" or word == "<TR>" or word =="<TD" or word == "<tr>" or word == "="
    :
                    continue


                ##### CHECK IF WORD EXIST AND INSERT IF NOT EXISTS############
                count = c.execute("SELECT count(*) FROM DICTIONARY WHERE WORD = ?",
    (word,))
                for row in count:
                    test = row[0]


                if test == 0:

                    if actval == 0:
                        ns = 1
                        nl = 2
                        n = 2
                        c.execute("INSERT OR IGNORE INTO DICTIONARY (ID,WORD,N,NS,NL
    , WM, WIS, WP, MI, WIH) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", (idn, word, n, ns, n
    l, 1, 0, 0, 0, 0));
                    else:
                        ns = 2
                        nl = 1
                        n = 2
                        c.execute("INSERT OR IGNORE INTO DICTIONARY (ID,WORD,N,NS,NL
    , WM, WIS, WP, MI, WIH) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", (idn, word, n, ns, n
    l, 1, 0, 0, 0, 0));
                    conn.commit()
                    ### INDEX INCREMENT##
                    idn = idn + 1

                    spamprob = float(number_spam_mails / number_mails)  ### PROBABIL
    ITY OF MAIL BEING A SPAM MAIL
                    hamprob = float(number_ham_mails / number_mails)
                    wordinspamprob = float(ns / n) #### PROBABILITY OF WORD BEING IN
     A SPAM MAIL
                    wordinhamprob = float(nl / n) #### PROBABILITY OF WORD BEING IN
    A SPAM MAIL
                    ######################## UPDATE PROB DATA ########################
    #################
                    c.execute("SELECT COUNT (*) FROM DICTIONARY")
                    number_of_words_in_dictionary = c.fetchone()[0]
                    ######################### MI INDEX CALC ######################
                    try:
                        mi = math.log(wordinspamprob/((2/number_of_words_in_dictiona
    ry)*spamprob))
                        #mi = math.log((wordinspamprob)/(2 / number_of_words_in_dict
    ionary))
                    except Exception as e:
                        mi = 0
                    c.execute("UPDATE DICTIONARY SET WIS = ?, WIH = ?, MI = ? WHERE
    WORD=?", (wordinspamprob, wordinhamprob, mi, word))
                    conn.commit()
                else:
                    pass
                ################### mark that word is in message ###################
    #############


                get_data = c.execute("SELECT WM FROM DICTIONARY WHERE WORD = ?", (wo
    rd,))
                foc = 0
                for data in get_data:
                    foc = data[0]
```

```
168.
169.                 ##########################################################################
    ###################
170.                 ########## UPDATE STATS FOR WORD ############################## do
    not increment if word already appeared in mail
171.                 if foc == 0:
172.                     #c.execute("UPDATE DICTIONARY SET WM = ? WHERE WORD=?", (1, word
    ))
173.                     conn.commit()
174.                     #### GET STATS FOR WORD#########################################
    ########################
175.                     get_data  = c.execute("SELECT N, NS, NL FROM DICTIONARY WHERE WO
    RD = ?", (word,))
176.
177.                     word_frequency = 0
178.                     spam_word_frequency = 0
179.                     ham_word_frequency = 0
180.
181.                     for data in get_data:
182.
183.                         word_frequency = data[0]
184.                         spam_word_frequency = data[1]
185.                         ham_word_frequency = data[2]
186.
187.
188.                     ns = spam_word_frequency
189.                     nl = ham_word_frequency
190.                     n = word_frequency + 1
191.
192.
193.                     if actval == 1:
194.
195.                         ns = ns + 1
196.
197.
198.                     elif actval == 0:
199.
200.                         nl = nl + 1
201.                     else:
202.                         pass
203.
204.
205.                     spamprob = float(number_spam_mails / number_mails)  ### PROBABIL
    ITY OF MAIL BEING A SPAM MAIL
206.                     hamprob = float(number_ham_mails / number_mails)
207.
208.                     wordinspamprob = float(ns / n) #### PROBABILITY OF WORD BEING IN
     A SPAM MAIL
209.                     wordinhamprob = float(nl / n) #### PROBABILITY OF WORD BEING IN
    A SPAM MAIL
210.
211.
212.                     c.execute("SELECT COUNT (*) FROM DICTIONARY")
213.                     number_of_words_in_dictionary = c.fetchone()[0]
214.                     ######################## MI INDEX CALC ######################
215.
216.                     try:
217.                         mi = math.log(wordinspamprob/((n/number_of_words_in_dictiona
    ry)*spamprob))
218.                         #mi = math.log((wordinspamprob)/(n/number_of_words_in_dictio
    nary))
219.                     except Exception as e:
220.                         mi = 0
221.
```

```
                    c.execute("UPDATE DICTIONARY SET N = ?, NS = ?, NL = ?, WIS = ?,
    WIH = ?, MI = ?,  WM = ? WHERE WORD=?", (n, ns, nl, wordinspamprob, wordinhamprob,
    mi, 1,  word))
                    ######################################################################

                    conn.commit()
                else:
                    pass

    ######### INCREMENT MAIL COUNTER ##############
    number_mails = number_mails + 1
    ################################################
    c.execute("UPDATE DICTIONARY SET WM = ? WHERE WM = ?", (0,1))
    conn.commit()
################ ALL MAILS CHECKED  ####################################
elapsed_time = time.time() - start_time

print "HAM PROB: ", hamprob
print "SPAM PROB: ", spamprob
print "ELAPSED TIME: ", (elapsed_time/60)
```

Priloga B: Razvrščanje

```
from __future__ import division
import os
from collections import Counter
import pprint
import itertools
import pandas as pd
import numpy as np
import sqlite3
import math
from heapq import nlargest
import nltk
import re
import time

start_time = time.time()

wacc = 0
werr = 0
werb = 0
tcr = 0
recall = 0
precision = 0

counth = 0
counts = 0

all_word_probs = 1
joint_prob = 1
joint_word_prob = 1

wordinspamprob = 0
wordprob = 0
spamprob = 0


total_prob = 0

words_in_mail = 0
```

```
number_mails = 1
number_spam_mails = 0
number_ham_mails = 0

aggregate_total_prob_spam = 0
aggregate_total_prob_ham = 0
lower_dyn_thresh = 0
upper_dyn_thresh = 0

n = 0
nl = 0
ns = 0
idn = 173378   #last id number from train db#

mi = 0
all_mis = 0
inp = 0
mi_list = []
mi_sorted = []
top_mi = 0

predval = 0
actval = 0
countmiss = 0
counthit = 0
undefined = 0

nss = 0
nll = 0
nsl = 0
nls = 0
nm = 0

conn = sqlite3.connect("test.db")
c = conn.cursor()
count_words = 0

hamprob = 0.467128027682  #probs from train#
spamprob = 0.532871972318


#file1 = open("train_ham.txt", "a")
#file2 = open("train_spam.txt", "a")

file1 = open("wacc.txt", "a")
file2 = open("werr.txt", "a")
file3 = open("werrb.txt", "a")
file4 = open("tcr.txt", "a")
file5 = open("recall.txt", "a")
file6 = open("precision.txt", "a")
file7 = open("nll.txt", "a")
file8 = open("nss.txt", "a")
file9 = open("nls.txt", "a")
file10 = open("nsl.txt", "a")
file11 = open("nm.txt", "a")
file12 = open("all.txt", "a")

c.execute('''''CREATE TABLE IF NOT EXISTS DICTIONARY
       (ID INT PRIMARY KEY  NOT NULL,
        WORD     TEXT UNIQUE NOT NULL,
        N                INT NOT NULL,
        NS               INT NOT NULL,
        NL               INT NOT NULL,
        WM               INT NOT NULL,
        WIS              INT NOT NULL,
        WP               INT NOT NULL,
```

```
        MI              INT NOT NULL,
        WIH             INT NOT NULL);''')

emails = [os.path.join("/path/to/mails/test",f) for f in os.listdir("/path/to/mails/test")]

for mail in emails:

    mailcheck = mail[-5:-4]

    print "MAIL NUMBER:", number_mails


    if mailcheck == "H":
        number_ham_mails = number_ham_mails + 1
        actval = 0

    elif mailcheck == "S":
        number_spam_mails = number_spam_mails + 1
        actval = 1
    else:
        pass



    with open(mail) as m:
    ##################### WORD PROCESSING ##############################################
    #######

        for i,line in enumerate(m):
            words = line.split()
            for word in words:

                word = word.decode("utf-8")
                word = word.lower()


                word = re.sub(r'[^\w\s]','',word)


                count = c.execute("SELECT count(*) FROM DICTIONARY WHERE WORD = ?", (word,))

                for row in count:
                    test = row[0]


                if test == 1:

                    get_data = c.execute("SELECT WM FROM DICTIONARY WHERE WORD = ?", (word,))
                    foc = 0
                    for data in get_data:
                        foc = data[0]

                    if foc == 0:
                        c.execute("UPDATE DICTIONARY SET WM = ? WHERE WORD=?", (1, word))
                        conn.commit()
                    else:
                        pass


                else:
                    pass

```

```
166.
167.    ################### ALL WORDS CHECKED ######################
168.    ################### CALCULATE PROBS ########################
169.    ############################################################
170.    ###############  ACTUAL PROBABILITY CALCULATION #############
171.    ############################################################
172.    get_data  = c.execute("SELECT WIS, WIH, WORD FROM DICTIONARY  WHERE WM = ? ORDER
     BY MI DESC LIMIT 15 ", (1,))
173.    #get_data  = c.execute("SELECT WIS, WIH FROM DICTIONARY  WHERE WM = ?", (1,))

174.
175.    all_words_in_spam_probs = 1
176.    all_words_in_ham_probs = 1
177.    all_words_list = 1
178.
179.    mi_list_prob_spam = []
180.    mi_list_prob_ham = []
181.    word_list = []
182.
183.    for data in get_data:
184.
185.        all_words_in_spam_probs = data[0]
186.        all_words_in_ham_probs = data[1]
187.        all_words_list = data[2]
188.
189.        if all_words_in_spam_probs == 0:
190.            all_word_in_spam_probs = 1
191.        else:
192.            pass
193.
194.        if all_words_in_ham_probs == 0:
195.            all_words_in_ham_probs = 1
196.        else:
197.            pass
198.
199.        mi_list_prob_spam.append(all_words_in_spam_probs)
200.        mi_list_prob_ham.append(all_words_in_ham_probs)
201.        word_list.append(all_words_list)
202.
203.
204.    result_joint_prob_spam = np.prod(np.array(mi_list_prob_spam))
205.    result_joint_prob_ham = np.prod(np.array(mi_list_prob_ham))
206.
207.
208.
209.    total_prob = (hamprob * result_joint_prob_ham) / ((result_joint_prob_spam * spam
    prob) + (result_joint_prob_ham * hamprob))
210.
211.    ######## TWO WAY ####################
212.
213.    if total_prob >= 0.8889:
214.        predval = 0
215.
216.    elif total_prob < 0.8889:
217.        predval = 1
218.
219.    else:
220.        pass
221.
222.
223.    ######## THREE_WAY ##################
224.
225.    #if total_prob > 0.0123 and total_prob < 0.8889:
226.    #   predval = 2
227.    #elif total_prob <= 0.0123:
228.    #   predval = 1
```

```
    #elif total_prob >= 0.8889:
    #   predval = 0
    #else:
    #   pass


    #if actval == 0:
    #    #file1.write("%s\n" %(total_prob))
    #    counth = counth +  1

    #elif actval == 1:
    #    #file2.write("%s %s\n" %(total_prob, number_mails))
    #    counts = counts +  1

    #### PREDICTION HIT #################

    if predval == 1 and actval == 1:            ### HIT SPAM ###
        nss = nss + 1
        counthit = counthit + 1


    elif predval == 0 and actval == 0:          ### HIT HAM ###
        nll = nll + 1
        counthit = counthit + 1


    elif predval == 0 and actval == 1:          ### MISS SPAM ###
        nsl = nsl + 1
        countmiss = countmiss + 1


    elif predval == 1 and actval == 0:          ### MISS HAM ###
        nls = nls + 1
        countmiss = countmiss + 1

    elif predval == 2:
        nm = nm + 1
    else:
        pass


    w = 9

    try:
        wacc = ((nll*w) + nss) / ((number_ham_mails* w) + number_spam_mails)
    except Exception as e:
        print "division by zero1"

    try:
        werr = ((nls*w) + nsl) / ((number_ham_mails* w) + number_spam_mails)
    except Exception as e:
        print "division by zero2"
    try:
        werrb = (number_spam_mails) / ((number_ham_mails* w) + number_spam_mails)
    except Exception as e:
        print "division by zero3"
    try:
        tcr = werrb / werr
    except Exception as e:
        print "division by zero4"
    try:
        recall = (nss)/ (nss + nsl)
    except Exception as e:
        print "division by zero5"
```

```
    try:
        precision = (nss) / (nss + nls)
    except Exception as e:
        print "division by zero6"


    file1.write("%s\n" %wacc)
    file2.write("%s\n" %werr)
    file3.write("%s\n" %werrb)
    file4.write("%s\n" %tcr)
    file5.write("%s\n" %recall)
    file6.write("%s\n" %precision)
    file12.write("%s %s\n" %(actval, total_prob))
    print "total_prob:", total_prob
    print "predval:", predval
    print "actval:", actval
    print "##############"

    #### PREDICTION MISS ################

    #print raw_input("SPACE TO CONTINUE")


    ######### INCREMENT MAIL COUNTER ###############
    number_mails = number_mails + 1
    ################################################


    c.execute("UPDATE DICTIONARY SET WM = ? WHERE WM = ?", (0,1))
    conn.commit()


################ ALL MAILS CHECKED  ######################################

elapsed_time = time.time() - start_time

print "HIT ACCURACY: ",  counthit/number_mails
print "MISS ACCURACY: ",  countmiss/number_mails
print "UNDECIDED ACCURACY: ",  undefined/number_mails
print "###################"
print "# HIT SPAM ACC: ", nss/number_spam_mails
print "# HIT HAM ACC: ", nll/number_ham_mails
print "# MISS SPAM ACC: ", nsl/number_spam_mails
print "# MISS HAM ACC: ", nls/number_ham_mails

print "TIME ELAPSED: ", (elapsed_time/60)
print "############################"
print "# HIT ACCURACY: ",  counthit
print "# MISS ACCURACY: ",  countmiss
print "# UNDECIDED ACCURACY: ",  undefined
print "###################"
print "# HIT SPAM: ", nss
print "# HIT HAM: ", nll
print "# MISS SPAM: ", nsl
print "# MISS HAM: ", nls
print "NUMBER HAM MAILS : ", number_ham_mails
print "NUMBER SPAM MAILS : ", number_spam_mails
print "count h: ", counth
print "count s: ", counts

file7.write("%s\n" %nll)
file8.write("%s\n" %nss)
file9.write("%s\n" %nls)
file10.write("%s\n" %nsl)
file11.write("%s\n" %nm)
```