UNIVERZA NA PRIMORSKEM

FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN

INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

Izdelava in evalvacija različnih modelov za napovedovanje zvrsti turizma iz besedil

(Development and evaluation of different models for predicting turist category from texts)

Ime in priimek: Andraž Stibilj

Študijski program: Računalništvo in informatika, 2. stopnja

Mentor: doc. dr. Branko Kavšek

Somentor: doc. dr. Jernej Vičič

Koper, september 2017

Ključna dokumentacijska informacija

Ime in PRIIMEK: Andraž STIBILJ

Naslov magistrskega dela: Izdelava in evalvacija različnih modelov za napovedovanje zvrsti turizma iz besedil

Kraj: Koper

Leto: 2017

Število listov: 82 Število slik: 21 Število tabel: 25

Število referenc: 51

Mentor: doc. dr. Branko Kavšek

Somentor: doc. dr. Jernej Vičič

UDK: 004.421(043.2)

Ključne besede: turistični korpus, podatkovno rudarjenje, tekstovno rudarjenje, obdelava naravnih jezikov, klasifikacijski algoritmi, napovedni model, WEKA

Izvleček: Področje tekstovnega rudarjenja je v zadnjih letih zaradi hitrega tehnološkega razvoja in vseprisotne informacijske tehnologije priča eksponenti rasti. Pri odkrivanju uporabnih informacij iz naravnih jezikov se v praksi uspešno uporabljajo klasifikacijski algoritmi v kombinaciji z analizo leksikografskih in lingvističnih vzorcev. Glavni cilj magistrskega dela je predstaviti področje tekstovnega rudarjenja s pripadajočimi metodologijami in tehnikami, v nadaljevanju pa izdelati in evalvirati različne napovedne modele na študiji primera slovenskega turističnega korpusa, ki zajema učno množico 4.688 besedil. Ta besedila so bila predhodno razvrščena v enega izmed 27 razredov s strani strokovnjaka. Pri izgradnji napovednih modelov smo skladno s širše uveljavljeno metodologijo CRISP-DM primerjali uspešnost različnih klasifikacijskih algoritmov za strojno učenje na nelematiziranih in lematiziranih besedilih, ki so bila pred postopkom modeliranja ustrezno prečiščena. Preverili smo vpliv izločanja nepomembnih besed in tistih brez informacijske vrednosti na točnost končnega modela. Uporabljena je bila odprtokodna programska oprema WEKA, ki vključuje širok nabor tehnik in algoritmov za izgradnjo in evalvacijo napovednih modelov. Končna evalvacija potrjuje minimalen vliv dejavnikov, ki naj bi vplivali na točnost napovednih modelov in ob enem predstavlja zadovoljive rezultate za morebitne izboljšave in implementacije modela v praksi.

Key words documentation

Name and SURNAME: Andraž STIBILJ

Title of the master's thesis: Development and evaluation of different models for predicting turist category from texts

Place: Koper

Year: 2017

Number of pages: 82 Number of figures: 21 Number of tables: 25

Number of references: 51

Mentor: Assist. Prof. Branko Kavšek, PhD

Co-Mentor: Assist. Prof. Jernej Vičič, PhD

UDK: 004.421(043.2)

Keywords: Tourist Corpus, Data Mining, Text Mining, Natural Language Processing, Classification Algorithms, Predictive Model, WEKA

Abstract: In recent years the field of text mining has witnessed an exponential growth due to rapid technological development and ubiquitous information technology. When discovering useful information from natural languages, classification algorithms are combined with the analysis of lexicographic and linguistic patterns. The main objective of the master's thesis is to present the field of text mining including all relevant methodologies and techniques. Moreover, we developed and evaluated various predictive models based on the case study of the Slovenian Tourist Corpus which encompasses the learning set of 4.688 texts. These texts have been previously classified in one of the 27 classes by an expert. For constructing predictive models we used the established CRISP-DM methodology. We compared the performance of different machine learning algorithms for classification on the original and lemmatized texts that were properly pre-processed before the modelling process. We examined the effect of eliminating stop words and those without any information gain on final model accuracy. Open source software WEKA was used, which includes a wide collection of techniques and algorithms for the construction and evaluation of predictive models. The final evaluation confirms the minimal impact of the factors that were expected to affect the accuracy of the predictive models.

ZAHVALA

Zahvaljujem se mentorju doc. dr. Branku Kavšku in somentorju doc. dr. Jerneju Vičiču za strokovno pomoč, nasvete in podporo, ki sem jih bil deležen pri izdelavi magistrskega dela. Brez njune konstruktivne kritike in usmeritev magistrsko delo zagotovo ne bi doseglo zastavljenih ciljev.

Posebna zahvala gre tudi moji družini in prijateljem, ki so me skozi vsa leta študija podpirali in spodbujali.

KAZALO VSEBINE

KAZALO TABEL

KAZALO SLIK

KAZALO ENAČB

SEZNAM KRATIC

ARFF	Attribute Relationship File Format = Atributno relacijski datotečni format
CART	Classification and Regression Trees = klasifikacijska in regresijska drevesa
CRISP-DM	Cross Industry Standard for Data Mining = industrijski standard za podatkovno rudarjenje
GNU	GNU's Not Unix! = GNU ni Unix!
IDF	Inverse Document Frequency = inverzna frekvenca dokumenta
KDD	Knowledge Discovery in Database = proces odkrivanja znanja iz podatkov
K-NN	K-Nearest Neighbors = metoda najbližjih sosedov
NLP	Natural Language Processing = Obdelava naravnih jezikov
QP	Quadratic Programming = kvadratično programiranje
SMO	Sequential Minimal Optimization = minimalna sekvenčna optimizacija
SVM	Support Vector Machine = metoda podpornih vektorjev
TF-IDF	Term Frequency–Inverse Document Frequency = frekvenca besede - inverzna frekvenca dokumenta
TISI	Turizem identiteta Slovenije
TURK	Turistični korpus
WEKA	Waikato Environment for Knowledge Analysis = zbirka algoritmov in tehnik za analizo podatkov razvita na Univerzi v Waikatu (Nova Zelandija)

1 UVOD

S prehodom iz industrijske v informacijsko dobo v poznih 70. letih prejšnjega stoletja se je začel hiter tehnološki razvoj, ki se kaže v prisotnosti različnih informacijskih tehnologij praktično na vseh gospodarskih in družbenih področjih [39]. Z digitalizacijo naše družbe se eksponentno povečuje število podatkov zbranih in shranjenih v številnih podatkovnih centrih širom sveta. Ker je zbrana količina podatkov že zdavnaj presegla človeško zmožnost odkrivanja koristnih informacij, je bilo treba razviti nove tehnike in orodja za odkrivanje novih znanj iz teh podatkov. V poznih 80. letih prejšnjega stoletja se je prvič pojavil pojem podatkovno rudarjenje (angl. Data Mining), ki se nanaša na odkrivanje znanja iz podatkov (angl. Knowledge Discovery in Databases). Han, Kamber in Pei ga v svojem delu definirajo kot »proces odkrivanja zanimivih vzorcev in znanja iz velike količine podatkov, dostopnih prek različnih podatkovnih baz/skladišč, spleta in drugih informacijskih odložišč (angl. Repositories)« [9]. Podatkovno rudarjenje je pravzaprav zgolj eden izmed ključnih korakov v procesu pridobivanja znanja iz podatkov, kjer z uporabo inteligentnih metod izločimo podatkovne vzorce za kasnejšo evalvacijo in vizualizacijo pridobljenih znanj [33] [8]. Obstaja več vrst tehnik podatkovnega rudarjenja. Eno izmed njih predstavlja klasifikacija (angl. Classification), ki temelji na strojnem učenju. Cilj klasifikacije je pri danih atributih uspešno napovedati vrednost ciljnega atributa – razreda s pomočjo različnih klasifikatorjev, kot so: odločitvena drevesa, odločitvena pravila, naivni Bayes, Bayesove verjetnostne mreže, metoda najbližjih sosedov, linearne diskriminantne funkcije, logistična regresija, metoda podpornih vektorjev in umetne nevronske mreže [20].

Klasifikacija je uspešno uporabljena pri tekstovnem rudarjenju (angl. Text Mining), ki se nanaša na odkrivanje uporabnih informacij v naravnih jezikih s pomočjo analize leksikografskih in lingvističnih vzorcev. V primeru tekstovne klasifikacije gre za razvrstitev posameznega besedila glede na njegovo vsebino v vnaprej določen razred. Tekstovna klasifikacija je danes glede na vse večjo količino tekstovnih podatkov uporabljena na področjih računalništva, biologije, varnosti, marketinga, menedžmenta in različnih medijev [32]. Magistrsko delo se nanaša na uporabo tekstovne klasifikacije kot tehnike za izgradnjo napovednega modela na študiji primera turističnega korpusa Slovenije [47].

Slovenski turizem je imel v zadnjih 25. letih nadpovprečno rast, čemur pričajo številke izvoza potovanj in tujih nočitev, ki so se v tem času potrojile. Pot do omenjenih rezultatov je povezana s prestrukturiranjem turističnega gospodarstva, lastninjenjem hotelskega sektorja, obnove nastanitvenih zmogljivosti, vzpostavitve novih organizatorjev potovanj, turističnih agencij in društev [48]. Ti so za svoj razvoj pametno izkoristili razvoj digitalizacije. Ob poplavi različnih digitalnih medijev, ki so dobrodošel vir informacij, se potencialni uporabniki turističnih storitev pri izbiri dogodka oziroma destinacije vedno težje znajdejo, zato ideja magistrskega dela predpostavlja izgradnjo modela s pomočjo napovednega modeliranja − tekstovne klasifikacije, s katero bomo razvrstili posamezno

turistično besedilo v eno izmed 26 kategorij. Za izgradnjo modela s pomočjo učne množice imamo na voljo okrog 4.688 besedil, predvsem iz spletnih strani, ki govorijo o različnih tipih turizma. Besedila so ročno pregledana s strani strokovnjaka in ustrezno razvrščena glede na vrsto turizma [47]. Namen magistrskega dela je zgraditi napovedni model, ki bo znal na podlagi besedila napovedati kategorijo − tip turizma. Uporabljena bo odprtokodna programska oprema Weka, ki predstavlja celovito rešitev za izgradnjo našega napovednega modela [51] [50]. Pri izgradnji modela bomo primerjali uspešnost različnih algoritmov za strojno učenje na neprečiščeni in prečiščeni različici turističnega korpusa. Preverili bomo ali lematizacija, izločanje nepomembnih besed in besed brez informacijske vrednosti vplivajo na točnost končnega modela ter kakšen je vpliv nerazvrščenih besedil.

1.1 MOTIVACIJA

Ob pregledu javno dostopne literature [6] [29] smo ugotovili, da je večina raziskav na področju tekstovnega rudarjenja narejena za angleški jezik, zato posledično nikjer nismo zasledili primera evalvacije algoritmov za strojno učenje na študiji primera v slovenskem jeziku. Ker želimo prispevati k razvoju tekstovnega rudarjenja v slovenskem jeziku, smo se odločili, da bomo na študiji primera turističnega korpusa analizirali najpomembnejše algoritme strojnega učenja, ki se uporabljajo pri tekstovni klasifikaciji. S podrobno analizo algoritmov želimo preveriti, kateri izmed algoritmov daje najboljše rezultate pri izgradnji napovednega modela in preučiti ali dejavniki izločanja nepomembnih ali nerazvrščenih besed ter besed brez informacijske vrednosti skupaj s postopkom lematizacije in čiščenja besedil vplivajo na točnost modela. Dodatno motivacijo predstavlja potencialna implementacija izdelanega modela v praksi, s čimer bi bralcem oziroma potencialnim uporabnikom turističnih storitev omogočili enostavnejše, predvsem pa hitrejše iskanje informacij po neurejenih bazah turističnih besedil.

1.2 OPIS PROBLEMA

Problematika magistrskega dela se nanaša na izdelavo napovednega modela z uporabo tekstovne klasifikacije kot ene izmed tehnik tekstovnega rudarjenja. Ročno pregledovanje in razvrščanje velikega števila besedil je v praksi zamudno, ob večji količini dokumentov pa praktično nemogoče, zato želimo z avtomatsko izdelavo čim boljšega napovednega modela olajšati omenjeno problematiko. Za študijo primera bomo uporabili 4.688 besedil turističnega korpusa, ki govorijo o različnih zvrsteh turizma. Besedila so ustrezno razvrščena v posamezno kategorijo s strani strokovnjaka, kar nam omogoča uporabo tehnik nadzorovanega učenja po postopku CRISP-DM [10]. Le-ta vnaprej določa potek analize podatkov, v našem primeru turističnih besedil, s pomočjo podatkovnega rudarjenja. Besedila bomo pred postopkom modeliranja ustrezno pripravili-prečistili, kar naj bi predstavljalo

najobsežnejšo fazo znotraj izbrane metodologije. Pri izgradnji modela bomo analizirali pet najpomembnejših algoritmov ter vpliv izločanja nepomembnih besed, besed brez informacijske vrednosti ter nerazvrščenih besedil v kombinaciji z lematizacijo in čiščenjem besedil.

1.3 PREGLED PODROČJA

Danes vse več ljudi komunicira preko elektronskih sporočil, blogov, forumov, tekstovnih sporočil in drugih socialnih omrežji, ki jih za poznejšo analizo in pridobivanje uporabnih znanj arhiviramo. Področje tekstovnega rudarjenja je v zadnjih letih zaradi eksponentne rasti teh besedil predmet različnih raziskav. Pri pregledu člankov in raziskovalnih del s področja tekstovnega rudarjenja smo se ozirali predvsem na raziskave, ki so vezane na evalvacijo algoritmov za tekstovno klasifikacijo. Opazili smo, da princip našega magistrskega dela sovpada z idejo, kjer so se tuji avtorji osredotočili predvsem na iskanje točnosti posameznih algoritmov s poudarkom na optimizaciji parametrov, ki jih le-ti ponujajo [29] [6].

V strokovnem članku »Comparison of Text Clasification Alogorithms« lahko najdemo primerjavo treh algoritmov za klasifikacijo na dveh podatkovnih zbirkah. Z uporabo Weke so bili analizirani algoritmi Naive Bayes, Support Vector Machine in C4.5 (J48). Podatkovni zbirki sta se razlikovali v številu instanc in atributov. Prva zbirka je vsebovala 768 instanc z 8 atributi, druga pa 40 instanc s 4 atributi. Pri testiranju omenjenih podatkovnih zbirk so avtorji prišli do spoznanja, da najboljše rezultate dosega SVM algoritem pod pogojem, da ima na voljo večje število atributov. Sledita mu Naive Bayes in J48, ki dosegata približno enake rezultate. SVM je v članku označen kot najboljši algoritem za tekstovno klasifikacijo pod pogojem, da so njegovi parametri ustrezno izbrani. Avtorji še dodajajo, da lahko izboljšanje SVM-algoritma dosežemo z združevanjem »Particle Swarm Optimization« in s tem, da problem kvadratičnega programiranja razbijemo na manjše podprobleme s čimer zmanjšamo časovno zahtevnost kar je še posebej uporabno pri večjih podatkovnih zbirkah [29].

Gandhi in Prajapati v članku o primerjavi algoritmov za klasifikacijo besedil primerjata tri algoritme – K-Nearest Neighbors, Naive Bayes in SVM. Pri tem poudarjata, da lahko v nekaterih študijah najdemo podatke o tem, Naive Bayes in K-NN dajeta boljše rezultate kot SVM, vendar pri tem študije ne upoštevajo, da oba algoritma spadata med večrazredne klasifikatorje, medtem ko je SVM v osnovi dvorazredni klasifikator. Pri večrazrednem problemu je treba SVM razširiti s tako imenovano tehniko enega proti vsem (*angl. One proti All*), ki primerja en razred z ostalimi razredi, ki so bili predhodno klasificirani. V primeru prejšnjih študij, katerih cilj je pravilno klasificirati več kot dva razreda, dejansko primerjamo več SVM-klasifikatorjev z enim večrazrednim Naive Bayes/K-NN klasifikatorjem. To je po mnenju avtorjev pomemben dejavnik pri evalvaciji,

ki je bil v preteklosti zanemarjen, zato njihova podatkovna zbirka vključuje le dva razreda – binarno klasifikacijo. Rezultati njune analize potrjujejo domnevo, da je SVM najnatančnejši klasifikator, medtem ko Naive Bayes dosega najslabše rezultate. Dodajata, da je za uspešno tekstovno klasifikacijo premalo zgolj združevanje pozitivnih lastnosti več algoritmov *(angl. Multi Model Process Technique),* ampak se je treba posvetiti predobdelavi in izbiri dobrih učnih besedil [6].

1.4 KRATEK PREGLED VSEBINE DELA

Magistrsko delo je v osnovi razdeljeno na dva dela – teoretični in empirični del. V prvem poglavju smo v uvodu podali nekaj širših metodoloških informacij o podatkovnem rudarjenju in nato prešli na klasifikacijo kot tehniko podatkovnega rudarjenja, ki smo jo uporabili za reševanja našega problema. V nadaljevanju smo predstavili motivacijo in opis problema ter natančneje pregledali literaturo, ki se nanaša na sorodno problematiko. V drugem poglavju sledi pregled raziskovalnega ozadja, kjer smo natančneje predstavili proces odkrivanja znanja iz podatkov, podatkovno rudarjenje in tehnike podatkovnega rudarjenja, metodologijo CRISP-DM, področje obdelave naravnih jezikov, tekstovno rudarjenje in uporabljeno programsko opremo. Empirični del magistrskega dela se začne s tretjim poglavjem, ki se nanaša na izgradnjo napovednega modela na primeru turističnega korpusa. Predstavljeno je razumevanje problema in podatkov, potek predobdelave podatkov, natančnejši opis klasifikacijskih algoritmov, ki smo jih uporabili pri študiji primera in postopek izgradnje napovednih modelov. V četrtem poglavju evalviramo uspešnost napovednih modelov s pregledom, primerjavo in interpretacijo rezultatov naše analize ter najboljšega napovednega modela. V zadnjem, petem poglavju podamo končne ugotovitve in ideje za nadaljnje delo oziroma izboljšave.

2 RAZISKOVALNO OZADJE

V drugem poglavju sledi pregled raziskovalnega ozadja, kjer smo natančneje predstavili proces odkrivanja znanja iz podatkov, podatkovno rudarjenje in tehnike podatkovnega rudarjenja, metodologijo CRISP-DM, področje obdelave naravnih jezikov, tekstovno rudarjenje in uporabljeno programsko opremo.

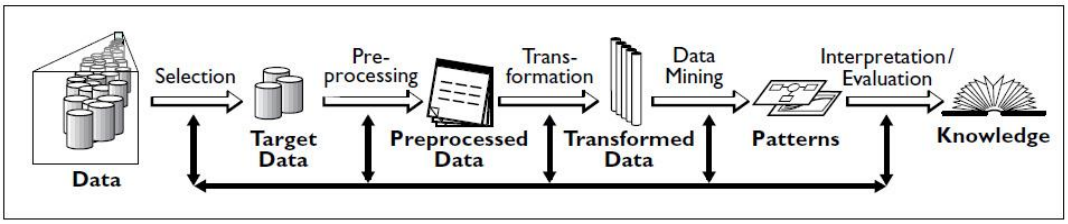
2.1 PROCES ODKRIVANJA ZNANJA IZ PODATKOV

Pojem odkrivanja znanja iz podatkov (angl. Knowledge in Databases) definiramo kot netrivialen proces identifikacije veljavnih, uporabnih in nedvoumno razumljivih vzorcev v podatkih [5]. Proces odkrivanja znanja iz podatkov je sestavljen iz več korakov, ki vključujejo pripravo podatkov, iskanje vzorcev, evalvacijo znanja – vse skozi večkratne iteracije. Z besedo netrivialen proces v definiciji pojma avtorji želijo poudariti, da je v proces vključeno iskanje strukture, modelov, vzorcev in parametrov. Vzorci morajo biti še neodkriti (*angl. Novel*), veljavni na testiranih podatkih in razumljivi z določeno stopnjo natančnosti. S temi pogoji zagotavljamo uporabno vrednost novega znanja. Ker večkrat prihaja do zamenjave pojmov podatkovnega rudarjenja in odkrivanja znanja iz podatkov, je treba poudariti, da podatkovno rudarjenje predstavlja zgolj enega izmed korakov v procesu odkrivanja znanja iz podatkov.

Fayyadov KDD model sestavlja devet korakov, ki so opisani takole [5]:

1. **Razumevanje področja uporabe** − nabor potrebnega predhodnega znanja in seznanitev s cilji končnega uporabnika o novo odkritem znanju.
2. **Določitev ciljnega nabora podatkov** − analitik izbere podmnožico spremenljivk (atributov) in podatkovne točke (primere), ki bodo uporabljeni za opravljanje nalog odkrivanja znanja. Ta korak običajno vključuje poizvedovanje obstoječih podatkov z namenom izbora želene podmnožice.
3. **Čiščenje in predprocesiranje podatkov** − vključuje osnovne operacije, kot so odstranjevanje šuma in osamelcev, zbiranje potrebnih podatkov za modeliranje in izračun šuma, ter določanje strategij za manipulacijo manjkajočih vrednosti.
4. **Reduciranje in projekcija podatkov** − z uporabo dimenzionalne redukcije in transformacijskih metod dosežemo zmanjšanje števila obravnavanih podatkov, ki za cilj naše analize niso uporabni.
5. **Izbira nalog podatkovnega rudarjenja** − povezava cilja KDD, definiranega v koraku ena z eno izmed KDD tehnik (klasifikacija, regresija, razvrščanje v skupine).

6. **Izbira algoritma za podatkovno rudarjenje** − vključuje izbor tehnik, uporabljenih pri iskanju vzorcev v podatkih, ter definiranje njihovih modelov in parametrov.
7. **Podatkovno rudarjenje** − generiranje vzorcev v določeni predstavitveni obliki (klasifikacijska pravila, odločitvena drevesa, regresijski modeli itd.).
8. **Interpretacija odkritih vzorcev** − vizualizacija odkritih vzorcev in modelov ter vizualizacija podatkov na podlagi pridobljenih modelov.
9. **Uporaba odkritega znanja** − vključevanje novega znanja v različne sisteme za nadaljnjo rabo, dokumentiranje in poročanje zainteresiranim stranem, preverjanje in reševanje morebitnih konfliktov s predhodno uveljavljenim znanjem.



Slika 1: Proces odkrivanja znanja iz podatkov (Vir slike: [45])

V procesu odkrivanja znanja ločimo dva tipa ciljev, opredeljenih z namenom uporabe sistema [5]:

1. Verifikacija (*angl. Verification*) – sistem je omejen na preverjanje uporabnikove hipoteze.
2. Odkritje (*angl. Discovery*) – sistem avtonomno išče nove vzorce. Odkritje je v naslednjem koraku razdeljeno še na napoved (*angl. Prediction)*, kjer sistem išče vzorce z namenom predvideti prihodnje ravnanje nekaterih subjektov, in opis (*angl. Description*), kjer sistem najde vzorce z namenom predstaviti jih uporabniku razumljivi obliki.

Večina raziskav se osredotoča predvsem na korak rudarjenja podatkov, vendar se je treba zavedati, da so prav vsi koraki znotraj procesa KDD enako pomembni. V naslednjem poglavju si bomo podrobneje pogledali proces podatkovnega rudarjenja [6] [5].

2.2 PODATKOVNO RUDARJENJE

Dve definiciji podatkovnega rudarjenja povzeti po knjigi »Data Mining: Practical Machine Learning Tools and Techiques« in »Data Mining: Concepts and Techniques se glasita:

1. »Podatkovno rudarjenje je proces odkrivanja vzorcev iz podatkov. Postopek mora biti avtomatski ali (bolj običajno) polavtomatski. Odkriti vzorci morajo biti uporabni do te mere da vodijo do koristi – najpogosteje ekonomske. Podatki so vedno prisotni v znatnih količinah [33].«
2. »Podatkovno rudarjenje, ki ga popularno imenujemo tudi odkrivanje znanja iz podatkov (KDD) je avtomatizirano ali priročno pridobivanje vzorcev. Pridobljeni vzorci predstavljajo implicitno shranjeno ali zajeto znanje v velikih podatkovnih bazah, podatkovnih skladiščih, spletu in drugih množičnih informacijskih repozitorijih ali podatkovnih tokovih [8].«

Zgornji definiciji predstavljata podatkovno rudarjenje kot samostojen proces, vendar smo že v prejšnjem poglavju definirali, da gre pravzaprav le za enega izmed korakov KDD procesa, ki temelji na preizkušenih tehnikah strojnega učenja, razpoznavanja vzorcev in statistike (klasifikacija, razvrščanje v skupine, regresija itd.). Lahko rečemo, da je podatkovno rudarjenje pravzaprav interdisciplinarno področje.

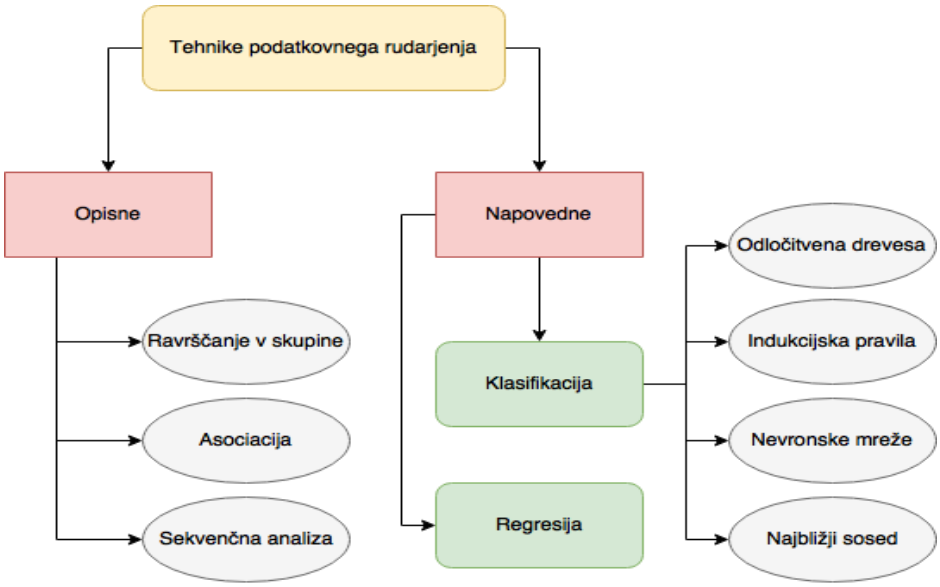
Slika 2: Podatkovno rudarjenje kot interdisciplinarno področje (Vir slike: [36])

2.3 TEHNIKE PODATKOVNEGA RUDARJENJA

Pri odkrivanju znanja iz podatkov se uporabljajo različni tehnike, kot recimo klasifikacija, razvrščanje v skupine, regresija, nevronske mreže, asociacijska pravila, odločitvena drevesa, genetski algoritmi in K-NN-metoda.

V prejšnjem podpoglavju smo že omenili, da poznamo dva tipa ciljev, opredeljenih z namenom uporabe sistema. V nadaljevanju se bomo osredotočili na sistem odkrivanja (*angl. Discovery*), ki avtonomno išče nove vzorce in ga delimo v dve ključni kategoriji [5]:

- napovedno podatkovno rudarjenje,
- opisno podatkovno rudarjenje.



Slika 3: Tehnike podatkovnega rudarjenja

2.3.1 Napovedno podatkovno rudarjenje

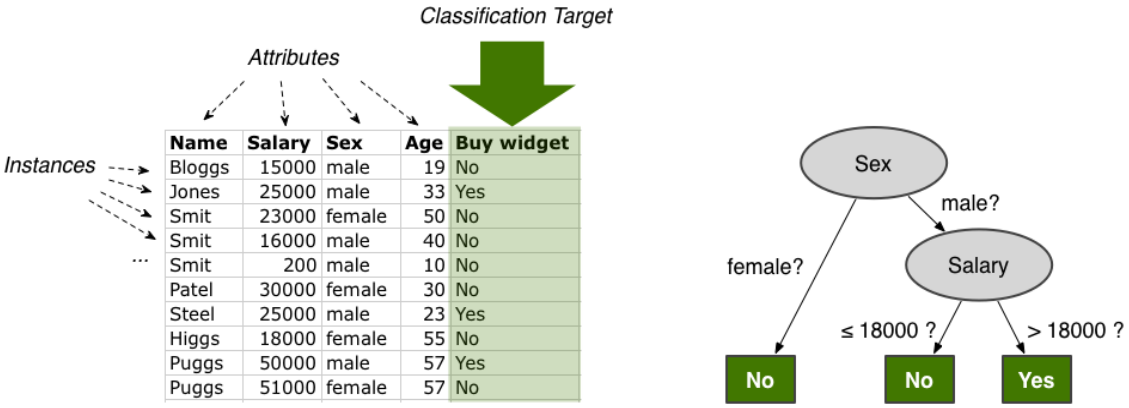
Napovedno podatkovno rudarjenje išče povezave med ciljnimi spremenljivkami ter ostalimi neodvisnimi ali pojasnjevalnimi spremenljivkami [4].

2.3.1.1 Klasifikacija

Klasifikacija je najpogosteje uporabljena tehnika napovednega podatkovnega rudarjenja, ki za razvoj modela uporablja niz predhodno razvrščenih podatkov. Področje odkrivanja goljufij in kreditnega tveganja sta še posebej primerna za to vrsto analize. Najpogosteje uporabljeni pristopi pri klasifikaciji so odločitvena drevesa in algoritmi za klasifikacijo na osnovi nevronskih mrež. Postopek klasifikacije vključuje učenje na učnem primeru in kasnejšo klasifikacijo testnih podatkov. Testni podatki se uporabijo za oceno točnosti pravil razvrščanja. Če je natančnost sprejemljiva, se pravila lahko uporabijo na testnih podatkih. Učni klasifikator uporablja predhodno razvrščene primere za določitev nabora parametrov, potrebnih za ustrezno diskriminacijo. Algoritem nato uporabi parametre na modelu, ki se imenuje klasifikator [21].

Vrste klasifikacijskih modelov:

- razvrščanje z indukcijo odločitvenih dreves,
- Bayesova klasifikacija,
- nevronske mreže,
- metoda podpornih vektorjev (SVM),
- razvrščanje na podlagi asociacij.



Slika 4: Primer klasifikacije z drevesnim indeksom (Vir slike: [37])

2.3.1.2 Regresija

Regresija se uporablja za določitev razmerja med odvisno spremenljivko (ciljno spremenljivko) in eno ali več neodvisnimi spremenljivkami. Odvisna spremenljivka je tista, katere vrednosti želimo napovedati, medtem ko so neodvisne spremenljivke tiste, na katerih temelji naša napoved. Za modeliranje povezave med eno neodvisno in več odvisnimi spremenljivkami uporabimo regresijsko analizo. V vsakodnevnem življenju problematika napovedovanja ni enostavna, ker se lahko nanaša na kompleksne interakcije med napovednimi spremenljivkami, zato je za napovedovanje novih vrednosti potrebno uporabiti naprednejše tehnike (logistična regresija, odločitvena drevesa, nevronske mreže). Ista vrsta modela je pogosto uporabljena tako za regresijo, kot tudi klasifikacijo. CART- napovedni drevesni algoritem je lahko uporabljen za izgradnjo tako klasifikacijskega kot regresijskega drevesa. Isto velja za nevronske mreže [21].

Vrste regresijskih metod:

- linearna regresija,
- multivariatna linearna regresija,

- nelinearna regresija,
- multivariatna nelinearna regresija.

2.3.2 Opisno podatkovno rudarjenje

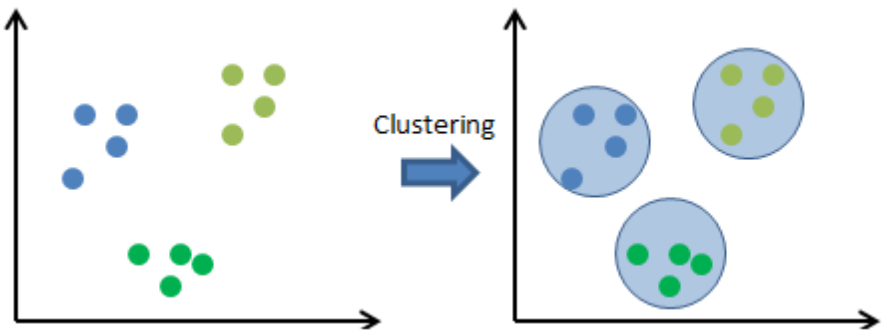
Opisno podatkovno rudarjenje uporablja vrste tehnik, ki znotraj podatkov odkrijejo vzorce (korelacije, trende, skupine, anomalije), te povzemajo (skrite) povezave v podatkih [4].

2.3.2.1 Razvrščanje v skupine

Pri razvrščanju v skupine gre za identificiranje podobnih objektov. Z uporabo metod razvrščanja v skupine lahko prepoznamo skupne regije v prostoru, kjer se objekti nahajajo, in odkrijemo celoten vzorec porazdelitve in korelacije med atributi podatkov. Kot alternativni pristop za identificiranje skupin ali razredov lahko uporabimo klasifikacijo, vendar z večjo količino podatkov ta postane potratna. Razvrščanje v skupine je uporabno v koraku preobdelave podatkov za določanje podmnožice atributov (*angl. Attribute Subset Selection*) in pri klasifikaciji [21].

Vrste metod za razvrščanje v skupine:

- metode razdeljevanja,
- hierarhične aglomerativne (divizijske) metode,
- metode, ki temeljijo na gostoti,
- metode na osnovi mreže,
- metode, ki temeljijo na modelih.



Slika 5: Primer razvrščanja v skupine (Vir slike: [14])

2.3.2.2 Analiza povezanosti

Analiza povezanosti je metoda, ki temelji na podlagi strojnega učenja za odkrivanje zanimivih zvez med spremenljivkami znotraj velikih podatkovnih baz. Njen namen je odkriti močne povezave/pravila znotraj podatkovne baze z uporabo mere zanimivosti. Primer analize povezanosti je analiza nakupovalne košarice, ki odkriva kombinacije dobrin, ki so jih kupile različne stranke. Z združevanjem lahko napovemo dobrine, ki se prodajajo znotraj iste transakcije [21] [4].

2.3.2.3 Sekvenčna analiza

Sekvenčna analiza se ukvarja z iskanjem statistično ustreznih vzorcev med primeri podatkov, kjer si vrednosti sledijo v zaporedju. Običajno se predpostavlja, da so vrednosti diskretne, zato je časovna vrsta rudarjenja (*angl. Time Series Mining*) tesno povezana s sekvenčno analizo, vendar običajno velja za drugačno aktivnost. Sekvenčna analiza velja v podatkovnem rudarjenju za poseben primer rudarjenja strukturiranih podatkov [15] [28].

2.4 METODOLOGIJA CRISP-DM

Metodologija CRISP-DM je daleč najbolj priljubljena metodologija za podatkovno rudarjenje, ki definira faze, ki jih raziskovalci uporabljajo pri procesu podatkovnega rudarjenja. Po podatkih iz leta 2014 kar 43 % raziskovalcev na področju podatkovnega rudarjenja uporablja omenjeno metodologijo. Mnogi ji pripisujejo kar »De Facto« standard za razvoj projektov podatkovnega rudarjenja in odkrivanja znanja. Prva verzija metodologije je bila predstavljena na četrti delavnici CRISP-DM SIG v Bruslju leta 1999, leto kasneje pa je bil izdan vodič s šestimi fazami. Zaporedje faz ni striktno definirano vnaprej, zato je mogoč premik med fazami. V spodnjem diagramu so predstavljene najpomembnejše in najpogostejše odvisnosti med posameznimi fazami. Zunanji cikel diagram predstavlja potek splošen potek podatkovnega rudarjenja, medtem ko notranji cikel vsebuje iteracije in prehode med različnimi fazami [10].

Slika 6: Faze metodologije CRISP-DM (Vir slike: [10])

Faze metodologije CRISP-DM zajemajo [26]:

1. **Razumevanje problema** (*angl. Business Understanding*)

Začetna faza se osredotoča na razumevanje projektnih ciljev in zahtev iz poslovnega vidika. To znanje se pretvori v definicijo problema in predhodni načrt za dosego ciljev. Uporabi se lahko model odločanja, ki temelji na modelu odločanja in notacija (model DMN).

2. **Razumevanje podatkov** (*angl. Data Understanding*)

Faza razumevanja podatkov se začne z zbiranjem podatkov in se nadaljuje s podrobnejšim pregledom, identifikacijo morebitnih pomanjkljivosti v podatkih, z iskanjem in reduciranjem podatkov z namenom ugotavljanja skritih vzorcev.

3. **Priprava podatkov** (*angl. Modeling*)

Faza priprave podatkov zajema vse dejavnosti za izdelavo končnega nabora podatkov (podatkov, ki bodo uporabljeni v programu za modeliranje) iz začetnih neobdelanih podatkov. Faza priprave podatkov se izvaja iterativno v naključnem vrstnem redu. Naloge vključujejo izbor tabele, zapisa in atributa ter preoblikovanje in čiščenje podatkov, ki so nato posredovani orodjem za modeliranje.

4. **Modeliranje** (*ang. Modeling*)

V tej fazi se izberejo in uporabijo različne tehnike modeliranja, njihovi parametri pa so kalibrirani do optimalnih vrednosti. Običajno obstaja več tehnik za reševanje iste vrste problema podatkovnega rudarjenja. Nekatere tehnike imajo posebne zahteve glede formata podatkov, zato je pogosto potrebno vračanje k fazi priprave podatkov.

5. **Evalvacija** (*angl. Evaluation*)

V tej fazi projekta imamo model, ki se nam zdi kakovosten iz vidika analize podatkov. Pred končno uporabo je pomembno temeljito ovrednotiti model in pregledati izvedene korake, da bi se prepričali, ali model dosega poslovne cilje. Na koncu evalvacije je treba sprejeti odločitev o tem, ali je model primer za nadaljnjo uporabo ali ne.

6. Uporaba modela (*ang. Deployment*)

V sklepni fazi je treba pridobljeno znanje iz modela organizirati in predstaviti tako, da je razumljivo končnemu uporabniku oziroma stranki. Pomembno je, da stranka razume v osnovi razume princip modela zato, da tega lahko maksimalno izkoristi.

2.5 OBDELAVA NARAVNIH JEZIKOV

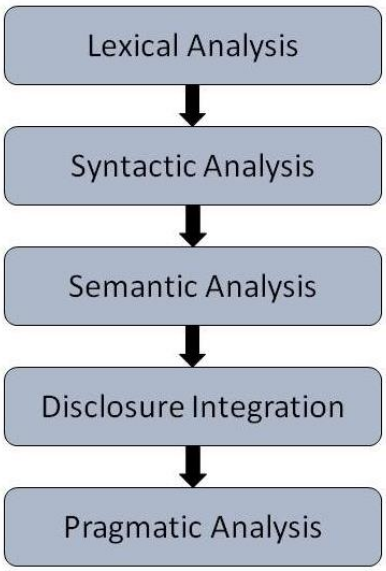
Obdelava naravnih jezikov (NLP) je interdisciplinarno področje računalništva, umetne inteligence in računalniškega jezikoslovja, ki preučuje interakcije med računalniki in človeškimi (naravnimi) jeziki. V osnovi želimo uspešno obdelati velike količine besedil − korpusov naravnega jezika. Izzivi, ki so prisotni pri obdelavi naravnega jezika pogosto vključujejo razumevanje naravnega jezika, ustvarjanje naravnega jezika (pogosto iz formalnih strojno čitljivih logičnih oblik), povezovanje jezika in strojne percepcije, ter pogovornih sistemov [41]. Ker naravni jeziki predstavljajo izjemno bogato obliko in strukturo, prihaja do različnih dvoumnih situacij na področju:

- leksikografske dvoumnosti (primer: angl. beseda board, ki lahko predstavlja samostalnik ali glagol),
- sintaktične dvoumnosti (primer: »He lifted the beetle with red cap.« Je uporabil kapo zato, da je dvignil hrošča ali je dvignil hrošča, ki ima rdečo kapo?),
- referenčne dvoumnosti (primer: »Rima went to Gauri. She said, “I am tired.”« Kdo natančno je tukaj utrujen Rima ali Gauri?)

Pri obdelavi naravnih jezikov ločimo pet splošnih korakov:

1. **Leksikografska analiza** − vključuje prepoznavanje in analizo strukture besed. Leksikon jezika pomeni zbiranje besed in besednih zvez v jeziku. Leksikalna analiza deli celotni del tekstovne datoteke v odstavke, stavke in besede. V splošnem gre za pripisovanje pomena posameznim besedam.
2. **Sintaktična analiza (***angl. Parsing***)** − vključuje analizo besed v stavku za namen preverjanja slovnične pravilnosti in urejanja besed na način, ki prikazuje razmerje med besedami. Stavek, kot je »The school goes to boy«, angleški sintaktični analizator zavrne.

3. **Semantična analiza** − iz besedila črpa natančen pomen oz. slovarski pomen. Preverja se smiselnost besedila. To se naredi tako, da se mapirajo skladenjske strukture in predmeti v domeni nalog. Semantični analizator tako ne upošteva stavka, kot je "vroč sladoled".
4. **Integracija diskurza** − pomen katerega koli stavka je odvisen od pomena stavka neposredno pred njim in za njim. V tej fazi se ustvarja povezava med stavki.
5. **Pragmatična analiza** − pragmatična analiza je preučevanje, kakšen je pravi pomen besedila oz. kako bi si ga bralec moral tolmačiti. Pragmatika se osredotoča na pomene besede, ker lahko ljudje rečejo eno stvar v praksi, mislijo pa nekaj povsem drugega. Gre za uporabo kontekstualnih in situacijskih pomenov iz vsakdanjega življenja [34].



Slika 7: Proces obdelave naravnih jezikov (Vir slike: [34])

Obdelava naravnih jezikov predstavlja eno izmed tehnik transformacije teksta v podatke, ki jih nato v ustrezni obliki uporabimo za nadaljnje korake tekstovnega rudarjenja. Za potrebe obdelave besedil iz turističnega korpusa ne bomo uporabili celotni postopek, opisan v prejšnjem koraku, ampak bomo uporabili tehnike, ki se uporabljajo pri fazi sintaktične analize. Najpogosteje uporabljene tehnike sintaktične analize pri tekstovnem rudarjenju so:

- tokenizacija − razdelitev besedila na besede oz. pojavnice,
- segmentacija − razpredelitev besedila na povedi – ločnice so ločila,

- krnjenje − proces normalizacije besedil, kjer besedam ostranimo predpone in pripone,
- lematizacija − besedi pripišemo njeno osnovno obliko oz. nedoločnik,
- oblikoslovno označevanje − vsaki besedi pripišemo njeno oblikoslovno oznako,
- filtriranje − s pomočjo seznama besed iz dokumentov odstranimo besede, ki za nas nimajo vrednosti (*angl. Stop Words*) [27].

2.6 TEKSTOVNO RUDARJENJE

Podatkovno rudarjenje je področje, ki v zadnjih nekaj letih hitro napreduje predvsem zaradi napredka informacijske tehnologije, ki ustvarja nove vrste podatkov. Trenutno je kar 90 % vseh svetovnih podatkov shranjenih v nestrukturirani obliki. Tekstovni podatki so rezultat širjenja spleta in socialnih omrežij in zaradi svoje nestrukturirane narave zahtevajo uporabo kvantitativni metod in algoritmov, ki lahko učinkovito procesirajo širok nabor tekstovnih podatkov [1] v strukturirano obliko.

Tekstovno rudarjenje, imenovano tudi rudarjenje besedilnih podatkov ali besedilna analiza, je proces pridobivanja visokokakovostnih informacij iz besedil. Visokokakovostne informacije običajno izhajajo iz oblikovanja vzorcev in trendov na podlagi učenja statističnih vzorcev. Tekstovno rudarjenje ponavadi vključuje postopek strukturiranja vhodnega besedila (običajno razčlenjevanje, skupaj z dodajanjem in odstranjevanjem nekaterih jezikovnih značilnosti), pridobivanje vzorcev znotraj strukturiranih podatkov in končno ocenjevanje/razlaga rezultatov. Visoka kakovost se v tekstovnem rudarjenju ponavadi nanaša na določeno kombinacijo relevantnosti, novosti in zanimanja. Tipično tekstovno rudarjenje vključujejo kategorizacijo besedil, razvrščanje besedil v skupine, ekstrakcijo koncepta/entitete, izdelavo zrnatih taksonomij, analizo razpoloženja, povzemanje dokumentov in modeliranje razmerja med entiteto (tj. učni odnosi med imenovanimi subjekti).

Tekstovna analiza vključuje pridobivanje informacij, leksikografsko analizo za preučevanje frekvence besed, prepoznavanje vzorcev, označevanje/notacijo, pridobivanje informacij, tehnike za podatkovno rudarjenje vključno z asociacijsko analizo, vizualizacijo in napovedno analitiko. Glavni cilj je pretvoriti besedilo v podatke za nadaljnjo analizo z uporabo procesiranja naravnega jezika (NLP) in analitskimi metodami [42].

2.6.1 Proces tekstovnega rudarjenja

1. Zbiranje dokumentov

V prvem koraku se zbirajo besedilni dokumenti, ki so prisotni v različnih formatih (pdf, text, html doc, css itd.),

2. Preobdelava dokumentov

V tej fazi se z dokumentov odstranjuje redundantne in neskladne podatke v naslednjem vrstnem redu:

- Tokenizacija (prepoznavanje pojavnic, tj. besed in ločil): dokument je predstavljen kot zbirka povedi. S tokenizacijo želimo celotno besedilo razbiti na posamezne besede. Pri tem so odstranjeni presledki, vejice, ločila itd.
- Odstranjevanje zaustavitvenih besed (*angl. Stop Words Removal*): iz dokumentov odstranimo besede, ki se nanašajo na najpogostejše besede znotraj besedila in nimajo nobene informacijske vrednosti. Ponavadi gre za veznike in predloge. Posebno pozornost moramo nameniti zaustavitvenim besedam, ki so del fraz, ker lahko z izločanjem le-teh spremenimo njihov pomen.
- Lematizacija (*angl. Stemming*): je bodisi ročno bodisi računalniško podprt postopek določanja osnovne (slovarske) oblike posameznim besedam, ki jih najdemo v besedilu. Osnovno obliko besede imenujemo lema [39].

3. Tekstovna transformacija

Tekstovna transformacija se nanaša na pretvorbo dokumenta v vrečo besed (angl. Bag Of Words) oziroma v vektorski model besed, ki je nato uporabljen za nadaljnjo analizo.

4. Izbira atributov

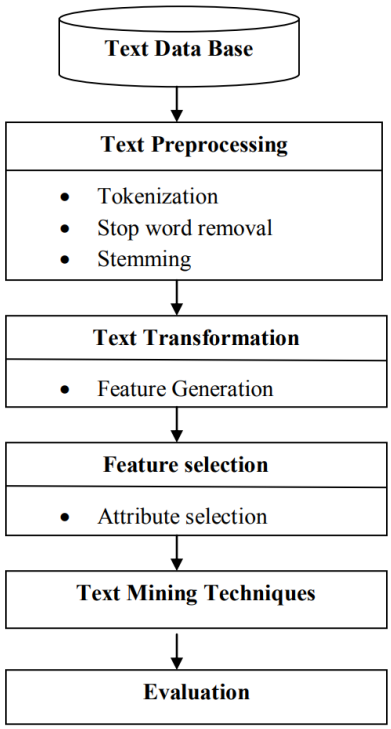
Tu odstranimo atribute, ki jih razumemo kot nepotrebne/nepomembne za postopek tekstovnega rudarjenja. Z reduciranjem števila atributov se zmanjša obseg podatkov, s tem pa tudi časovna in prostorska zahtevnost izgradnje modela.

5. Podatkovno rudarjenje/Izbira vzorcev

V tej fazi se konvencionalni proces podatkovnega rudarjenja združi s procesom tekstovnega rudarjenja. Obstajajo različne tehnike tekstovnega rudarjenja. Te so že bile opisane v poglavju »Tehnike podatkovnega rudarjenja«. Gre za klasifikacijo, razvrščanje v skupine, odkrivanje informacij in tem, povzemanje in ekstrakcijo tem.

6. Evalvacija

Ta faza vključuje vrednotenje in interpretacijo rezultatov v smislu izračuna natančnosti [2] [23].



Slika 8: Proces tekstovnega rudarjenja (Vir slike: [23])

2.7 IZBIRA ORODJA ZA PODATKOVNO RUDARJENJE

Podatkovno rudarjenje ima dolgo zgodovino z močnimi povezavami v statistiki, umetni inteligenci, strojnim učenjem in raziskavami podatkovnih baz. Besedo "podatkovno rudarjenje" lahko najdemo relativno zgodaj − že v osemdesetih letih prejšnjega stoletja. Napredek na tem področju je spremljal razvoj s tem povezanih programskih orodij, začenši z glavnimi programi za statistično analizo v zgodnjih petdesetih letih prejšnjega stoletja, ki je vodil do velikega števila samostojnih, odjemalcev in spletne programske opreme kot današnje servisne rešitve.

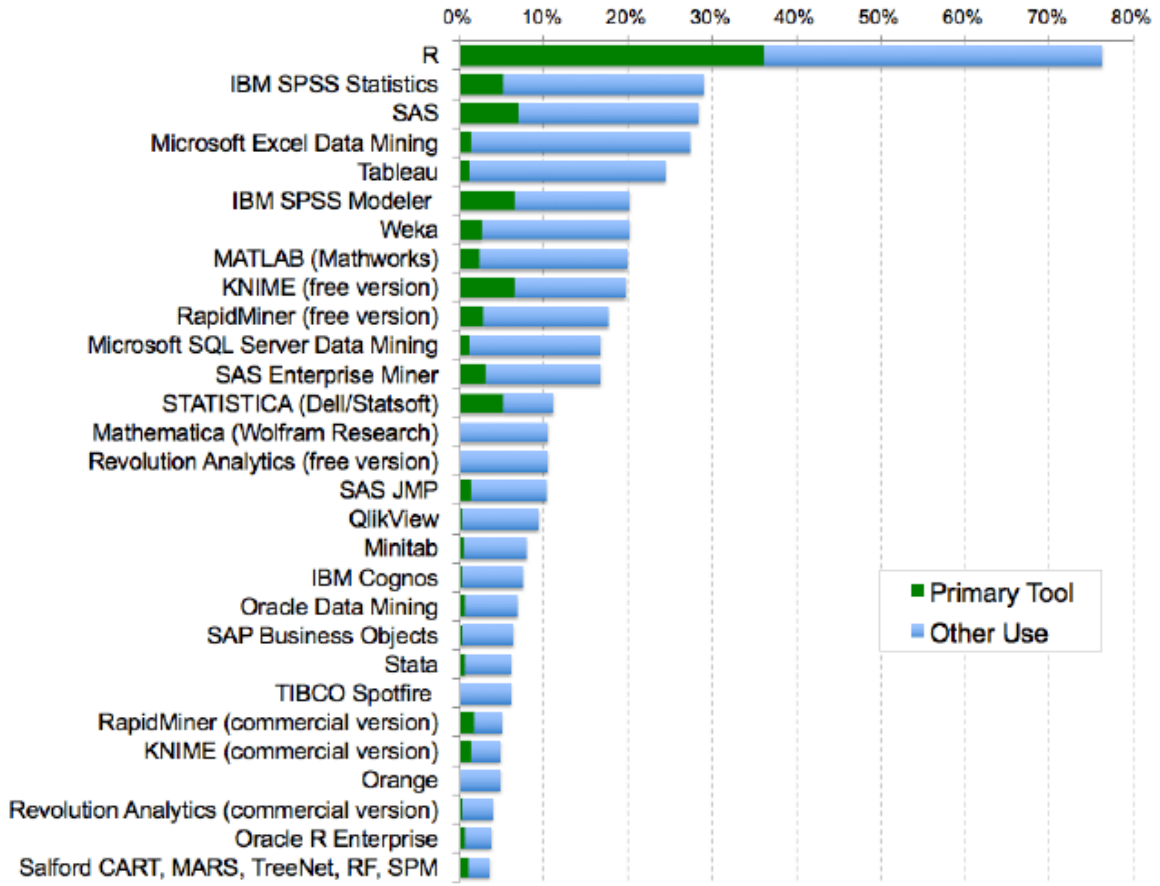
Največje komercialne zgodbe o uspehu so bile rezultat postopnega vključevanja metod podatkovnega rudarjenja v uveljavljena komercialna statistična orodja. Podjetja, kot je SPSS, ustanovljena leta 1975 s predhodniki iz leta 1968, ali SAS, ustavljen leta 1976, ponujajo statistična orodja za računalnike že od sedemdesetih let prejšnjega stoletja. Ta orodja so bila kasneje prilagojena osebnim računalnikom in odjemalcem kot rešitve za večje število uporabnikov. Z vse večjo priljubljenostjo podatkovnega rudarjenja so v glavne produkte vključevali algoritme, kot so umetne nevronske mreže ali odločitvena drevesa na način prevzema podjetij, specializiranih za podatkovno rudarjenje. Primer Integrated Solutions Ltd., prevzet leta 1998 s strani SPSS, je zdaj na voljo pod imenom IBM SPSS Modeler.

Danes so na voljo številna odprtokodna in plačljiva orodja za podatkovno rudarjenje. Pokrivajo širok spekter programskih rešitev, vse od specializiranih rešitev za klasifikacijo, razvrščanje v skupine in segmentacijo, tekstovno rudarjenje, statistične analize, spletno rudarjenje po socialnih omrežjih, zvoku in video zapisih. Na drugi strani imamo na voljo programske pakete, ki omogočajo izvedbo celotnega KDD-procesa, vključno s preobdelavo, evalvacijo in integracijo s podatkovnimi bazami/skladišči [17]. V osnovi se orodja razlikujejo v ciljnih skupinah, podatkovni strukturi, podpori algoritmom in tehnikam, vrsti interakcije z orodjem, kapaciteti in formatu uvoza in izvoza podatkov ter ne nazadnje ceni v primeru uporabe plačljivega orodja.

Za izbor optimalnega orodja, ki smo ga uporabili pri naši analizi, smo upoštevali naslednje kriterije:

- enostavna uporaba prek grafičnega vmesnika,
- dostopna in razumljiva dokumentacija,
- podprte tehnike za izvedbo celovitega KDD-procesa,
- zagotavlja zadovoljiv nivo točnosti,
- podprti različni OS,
- je brezplačen za uporabo.

V spodnjem diagramu je prikazan pregled najbolj uporabljenih orodij za podatkovno rudarjenje na osnovi ankete podjetja Rexer Analytics iz leta 2015. V anketi je sodelovalo 1220 strokovnjakov za podatkovno rudarjenje, ki so v anketi izbrali enega ali več orodij, uporabljenih pri njihovem delu. Anketiranci so pri izbiri orodja morali označiti, ali gre za njihovo primarno (edino) orodje ali pri delu uporabljajo še kako drugo orodje [18].



Slika 9: Pregled najbolj uporabljenih orodij za podatkovno rudarjenje (Vir slike: [18])

Na podlagi prej naštetih meril in pregleda najbolj uporabljenih orodij smo se odločili za odprtokodno rešitev, imenovano WEKA, ki je rezultat večletnega projekta univerze Waikato na Novi Zelandiji na področju raziskovanja podatkovnega rudarjenja in strojnega učenja.

2.7.1 Weka

Weka je priljubljena odprtokodna programska oprema za strojno učenje, napisana v Javi. Nastala je na univerzi Waikato na Novi Zelandiji. Na voljo je pod licenco GNU. Delovno okolje programa je sestavljeno iz nabora orodij in algoritmov za analizo podatkov in modeliranje napovedovanja. Vsebuje grafični vmesnik, ki omogoča preprosto uporabo. Prvotna verzija iz leta 1993 je bila narejena z različnimi programerskimi orodji. Zaradi prenosljivosti na raznoliko strojno opremo so jo v 3. verziji leta 1997 prepisali v programski jezik Java, vključno z implementacijo vseh takratnih algoritmov. Zadnja stabilna različica v uporabi je 3.9.x.

Weka omogoča mnoga standardna opravila na področju podatkovnega rudarjenja (predobdelava, rojenje, razvrščanje, regresija, vizualizacija, izbira značilk) [51] [50].

V glavnem oknu Weke najdemo naslednje aplikacije:

- Raziskovalec (*angl. Explorer*), kjer najdemo vse potrebno za izvedbo KDD-procesa − od preobdelave pa do končne vizualizacije.
- Preizkuševalec (*angl. Experimenter*), ki omogoča sistematično medsebojno primerjavo rezultatov strojnega učenja s podatkovnimi bazami.
- Tok znanja *(ang. KnowledgeFlow)* je funkcijsko podoben Raziskovalcu, vendar tu KDD postopek sestavljamo grafično s pomočjo vlečenja (angl. Drag-and-Drop) posameznih elementov.
- Delavnica (*angl. Workbench*) je skupek vseh zgoraj naštetih grafičnih vmesnikov v enem oknu, zato je navigacija in dostop do njih hitrejša.
- *Simple CLI* omogoča klic metod z uporabo ukazne vrstice.

Slika 10: Weka GUI Chooser 3.8.1

Datoteka ARFF

ARFF je besedilna datoteka za zapis Wekinih podatkovnih zbirk. Preprost primer ARFF-datoteke je v tabeli 1, kjer imamo v glavi najprej definirano ime relacije (@relation), pet atributov (@attribute) različnih podatkovnih tipov in razdelek s podatki – instancami (@data).

Tabela 1: Primer ARFF datoteke v Weki

```
@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}


@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
```

Vrstice, ki se začnejo z znakom @, imenujejo glava datoteke, ostalo pa so podatki − instance. Število atributov posamezne instance se mora ujemati s številom prej definiranih atributov (@attribute). Tip podatkov je lahko numeričen, nominalen, niz, datum in relacija. Weka podpira uvoz ostalih formatov (.csv, .json, .m, .libsvm,.xrff, itd.), vendar je ARFF-format za obdelavo hitrejši, manj prostorsko zahteven in nasploh omogoča boljšo analizo, ker že vnaprej definira metapodatke v atributih [50].

3 IZGRADNJA NAPOVEDNEGA MODELA NA PRIMERU TURISTIČNEGA KORPUSA

Tretje poglavje se nanaša na izgradnjo napovednega modela na primeru turističnega korpusa. Predstavljeno je razumevanje problema in podatkov, potek predobdelave podatkov in natančnejši opis klasifikacijskih algoritmov, ki smo jih uporabili pri študiji primera ter postopek izgradnje napovednih modelov.

3.1 RAZUMEVANJE PROBLEMA

Kot je bilo v uvodnem opisu problema že predstavljeno, se v praktičnem delu lotevamo problema izgradnje napovednega modela z uporabo tekstovne klasifikacije kot ene izmed tehnik tekstovnega rudarjenja. Ob vse večji količini nestrukturiranih podatkov je ročno pregledovanje in razvrščanje velikega števila besedil v praksi zamudno, ob večji količini dokumentov pa praktično nemogoče. Z izdelavo čim boljšega napovednega modela želimo omenjeno problematiko v prihodnje olajšati oziroma odpraviti, ob enem pa analizirati vpliv posameznih dejavnikov, ki vplivajo na točnost modela. Za študijo primera smo uporabili 4.688 besedil (instanc) turističnega korpusa, klasificirani v 27 razredov. Besedila so bila pred izgradnjo napovednega modela razvrščena v posamezen razred s strani strokovnjaka, zato nam predstavljajo široko učno množico.

V nadaljevanju je opisan celoten postopek vse od zajema podatkov pa do končne izgradnje modela z izbranimi tehnikami za podatkovno rudarjenje. Pri izgradnji modela smo analizirali pet najbolj razširjenih algoritmov ter dodatni vpliv dejavnikov lematizacije, nerazvrščenih besed, izločenih besed (*angl.* Stopwords) in besed brez informacijske vrednosti. Za primerjavo modelov smo uporabili neprečiščeno in prečiščeno različico turističnega korpusa v kombinacijah z zgoraj opisanimi dejavniki.

3.2 RAZUMEVANJE PODATKOV

V tem podpoglavju je predstavljen način pridobivanja podatkov/besedil, zbranih v turističnem korpusu ter podroben statistični pregled besedil po kategorijah, v katere so bili ti razvrščeni. Kategorizacija besedil je razvidna iz tabele 2 in 3.

3.2.1 Način pridobivanja podatkov

Gradiva so bila zbrana v dveh delih: v obliki prve različice korpusa TURK [47] ter nadaljevanja TURK2, ki se je izvajal v okviru projekta TISI [46]. Gradiva so bila zbrana ročno − avtorska gradiva smo dobili po elektronski pošti in na medijih (DVD, CD) ter

samodejno s pajkanjem po spletnih straneh (*angl. Crawling*). Gradiva so bila samodejno pretvorjena v enoten format in kodno stran − čisto besedilo v kodni strani UTF-8 [30].

3.2.2 Opis podatkov

Podatke smo črpali iz turističnega korpusa (oz. v jezikoslovnem strokovnem jeziku tudi zgolj korpus), ki v svoji definiciji predstavlja obsežno zbirko besedil v naravnem jeziku, zajetih v določenem obdobju iz množičnih medijev (časopisnega in revijalnega tiska), knjižne produkcije, stripov, interneta, reklamnih besedil, navodil, priloženih izdelkom široke potrošnje, prepisov parlamentarnih razprav ipd., shranjene v strukturirani obliki običajno na digitalnih medijih in s pomočjo jezikovnih tehnologij, pogosto opremljenih z označbami. Med označbe, ki jih vsebujejo nekateri korpusi, spadajo leme, tj. osnovne oblike besed, oblikoskladenjske označbe, skladenjske označbe in še nekatere druge označbe, ki se uporabljajo ne le za namene korpusnega jezikoslovja, ampak tudi za namene prepoznavanje govora in strojnega prevajanja v računalniški lingvistiki [35].

Specializirani jezikovni korpus s področja turizma je zbirka turističnih besedil različnih besedilnih zvrsti in vrst z različnih področij, povezanih s turizmom, s 30.000.000 besed v slovenskem, italijanskem in angleškem jeziku. Nastal je v okviru temeljnega raziskovalnega projekta Večjezični korpus turističnih besedil – informacijski vir in analitična baza slovenske naravne in kulturne dediščine pri UP ZRS (Univerza na Primorskem, Znanstveno-raziskovalno središče) [28].

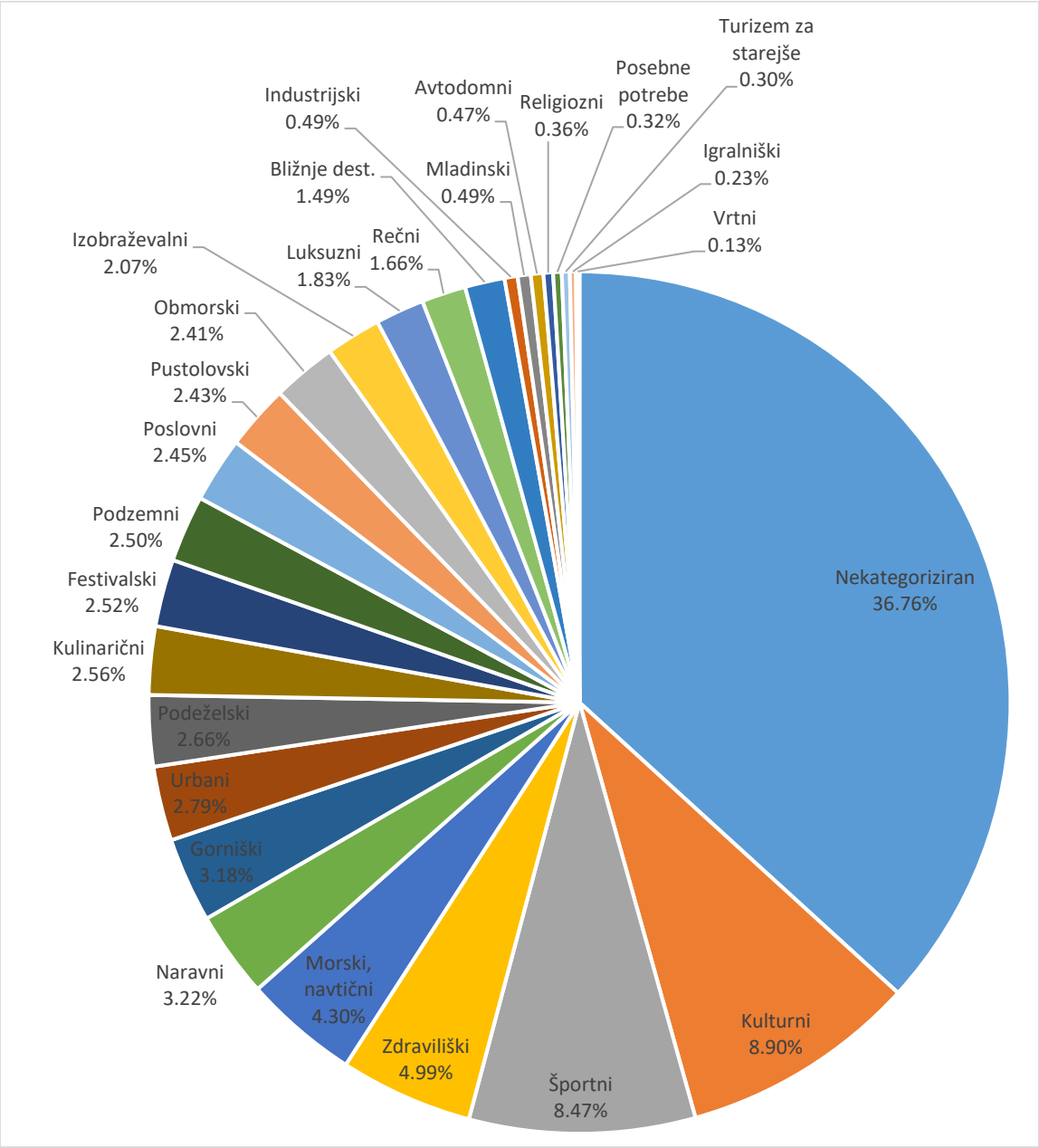
V turističnem korpusu se pred fazo preobdelave nahaja 4.688 besedil (6.212.670 besed), od tega 2965 (3.636.620 besed) tistih, ki so bile s strani strokovnjaka razvrščene v enega izmed 26. razredov. Ostalih 1723 je razvrščenih v razred 0 (2.576.050 besed), ker jih strokovnjak na podlagi besedila ni uspel razvrstiti v točno določeno kategorijo turizma. Na koncu vsakega besedila najdemo označevalec (#TURIZEM#Ft.T + številka turizma), ki predstavlja razred, v katerega je bilo besedilo razvrščeno.

V tabeli 2 je prikazanih vseh 26 tipov turizma, vključno z nekategoriziranim razredom 0, razvrščenih po deležu besedil od največjega proti najmanjšemu. V tabeli 3 najdemo grafični prikaz kategorij turizma po deležih za neprečiščena besedila.

Tabela 2: Kategorije turizma po deležih za neprečiščena besedila

Tip turizma	**Abs. frekvenca po razredih**	**Rel. frekvenca po razredih**	**Razred**
Nekategoriziran turizem	1723	36,75 %	0
Kulturni turizem	417	8,9 %	2
Športni turizem	397	8,47 %	23
Zdraviliški turizem	234	4,99 %	13
Morski, navtični	202	4,3 %	20
Naravni turizem	151	3,22 %	11
Gorniški turizem	149	3,18 %	10
Urbani turizem	131	2,79 %	24
Podeželski turizem	125	2,66 %	21
Kulinarični turizem	120	2,56 %	26
Festivalski turizem	118	2,52 %	6
Podzemni turizem	117	2,5 %	22
Poslovni turizem	115	2,45 %	3
Pustolovski turizem	114	2,43 %	4
Obmorski turizem	113	2,41 %	19
Izobraževalni turizem	97	2,07 %	7
Luksuzni turizem	86	1,83 %	9
Rečni turizem	78	1,66 %	18
Turizem bližnjih destinacij	70	1,49 %	12
Industrijski turizem	23	0,49 %	5
Mladinski turizem	23	0,49 %	15
Avtodomni turizem	22	0,47 %	17
Religiozni turizem	17	0,36 %	1
Turizem za osebe s posebnimi potrebami	15	0,32 %	14
Turizem za starejše	14	0,3 %	16
Igralniški turizem	11	0,23 %	25
Vrtni turizem	6	0,13 %	8
SKUPAJ	4.688	100 %	

Tabela 3: Tortni prikaz kategorij turizma po deležih za neprečiščena besedila v %



3.3 PREOBDELAVA PODATKOV

Faza preobdelave podatkov predstavlja enega ključnih korakov za izgradnjo dobrega napovednega modela. Podatki, ki smo jih pridobili iz različnih virov, so shranjeni v besedilnem formatu in vsebujejo znake, fraze, ali celotna besedila, ki jih je treba pred korakom tekstovnega preprocesiranja odstraniti. Za urejanje dokumenta v velikosti 41 MB potrebujemo zmogljiv in zanesljiv tekstovni urejevalnik, podprt z osnovnimi funkcijami za iskanje/zamenjavo in podporo Regex izrazom. Pomemben dejavnik je tudi dejstvo, da je odprtokoden in brezplačen za uporabo. Izbirali smo med dvema najbolj priljubljenima

tekstovnima urejevalnikoma Sublime 3 in Notepad++ po raziskavi spletne strani g2crowd. Primerjali smo zmogljivosti omenjenih dveh tekstovnih urejevalnikov na področju:

- hitrosti odpiranja dokumenta,
- hitrost iskanja/zamenjave osnovnih in Regex izrazov,
- uporabniškega vmesnika (vizualizacija najdenih/zamenjanih besed).

Primerjava tekstovnih urejevalnikov na podatkih turističnega korpusa je pokazala, da povprečni čas odpiranja 41 MB velike tekstovne datoteke znaša 1 sekundo proti 30 sekundam, čas zamenjave 9202 znakov pa 3 sekunde proti 9,45 minutam v korist Notepad-a++. Tudi uporabniški vmesnik in prikaz iskanja/zamenjave je bil precej bolj tekoč in intuitiven kot pri Sublime 3, zato smo se odločili za uporabo Notepad++.

Ker je bila količina tekstovne datoteke za ročno čiščenje izjemno obsežna, smo želeli postopek čiščenja, kar se da avtomatizirati. Slednje nam je do neke mere uspelo realizirati z uporabo Regex izrazov, če smo iz besedila lahko razbrali ponavljajoč se in logičen vzorec.

S pomočjo Regex izrazov smo odstranili, zamenjali ali prevedli:

- HTML-elemente,
- nabor znakov oz. simbolov, ki so bili brez pomena ali niso bili podprti v UTF-8 kodiranju,
- posamezne ponavljajoče se fraze brez pomena,
- tuje izraze znotraj slovenskih besedil.

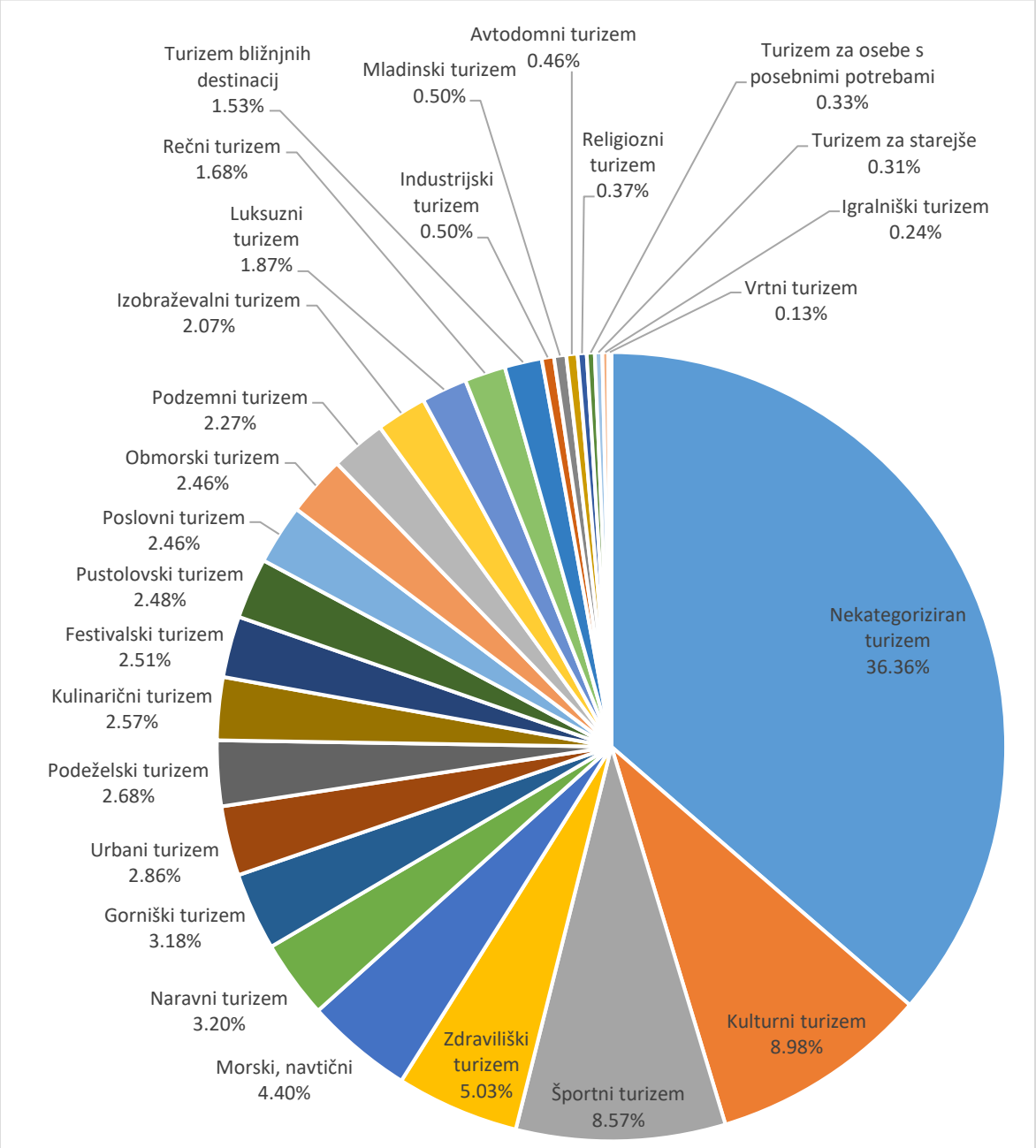
Nekatere nepravilnosti je bilo treba odpraviti skozi ročno kontrolo besedila. Na ročno čiščenje se nanaša predvsem rekonstrukcija teksta (manjkajoči šumniki, slovnične in pravopisne napake), odstranjevanje tujih besedil (italijanščina, nemščina, hrvaščina, francoščina), odstranjevanje besedil, ki jih ni bilo zmožno rekonstruirati. Od prvotnih 4.688 besedil je po koncu čiščenja ostalo 4588 besedil, ki jih je bilo treba pripraviti na uvoz v WEKO. V tabeli 4 so predstavljene spremembe v frekvencah po končanem postopku čiščenja. Razredi, iz katerih smo odstranjevali besedila, so v tabeli 4 označeni z rumeno barvo. V stanju +/- je zabeležena sprememba v relativni in absolutni frekvenci števila besedil po čiščenju. Iz začetnega neprečiščenega korpusa smo odstranili 100 besedil.

Tabela 4: Kategorije turizma po deležih za prečiščena besedila

Tip turizma	Abs. frekvenca po razredih	Stanje +/-	Rel. frekvenca po razredih	Stanje +/-	Razred
Nekategoriziran turizem	1668	-55	36,36 %	-0,40 %	0
Kulturni turizem	412	-5	8,98 %	+0,08 %	2
Športni turizem	393	-4	8,57 %	+0,10 %	23
Zdraviliški turizem	231	-3	5,03 %	+0,04 %	13
Morski, navtični	202	0	4,40 %	-0,09 %	20
Naravni turizem	147	-4	3,20 %	-0,02 %	11
Gorniški turizem	146	-3	3,18 %	0,00 %	10
Urbani turizem	131	0	2,86 %	+0,06 %	24
Podeželski turizem	123	-2	2,68 %	+0,01 %	21
Kulinarični turizem	118	-2	2,57 %	+0,01 %	26
Festivalski turizem	115	-3	2,51 %	-0,01 %	6
Pustolovski turizem	114	-3	2,48 %	-0,01 %	4
Poslovni turizem	113	-2	2,46 %	+0,01 %	3
Obmorski turizem	113	-1	2,46 %	+0,03 %	19
Podzemni turizem	104	-9	2,27 %	-0,14 %	22
Izobraževalni turizem	95	-2	2,07 %	0,00 %	7
Luksuzni turizem	86	0	1,87 %	+0,04 %	9
Rečni turizem	77	-1	1,68 %	+0,01 %	18
Turizem bližnjih destinacij	70	0	1,53 %	+0,03 %	12
Industrijski turizem	23	0	0,50 %	+0,01 %	5
Mladinski turizem	23	0	0,50 %	+0,01 %	15
Avtodomni turizem	21	-1	0,46 %	-0,01 %	17
Religiozni turizem	17	0	0,37 %	+0,01 %	1
Turizem za osebe s posebnimi potrebami	15	0	0,33 %	+0,01 %	14
Turizem za starejše	14	0	0,31 %	+0,01 %	16
Igralniški turizem	11	0	0,24 %	+0,01 %	25
Vrtni turizem	6	0	0,13 %	0,00 %	8
SKUPAJ	4588	-100	100 %		

V tabeli 5 najdemo grafični prikaz kategorij turizma po deležih za neprečiščena besedila.

Tabela 5: Tortni prikaz kategorij turizma po deležih za prečiščena besedila v %



3.3.1 Priprava podatkov na uvoz v WEKO

Prečiščene podatke smo morali skladno z Wekinim ARFF-formatom pripraviti na uvoz. Ker WEKA za označevanje začetka tipa vrste niz uporablja enojno navednico »'«, je bilo potrebno vsem enojnim navednicam, ki se niso nahajala na prvem mestu besedila dodati izhodno sekvenco v obliki poševnice »\'«. Vsako besedilo je v osnovi že vsebovalo

označevalec za začetek in konec besedila. Začetek je bil označen z enojno navednico »'«, konec pa z nizom »#TURIZEM#Ft.T« v primeru nekvalificiranega besedila (razred 0), ali #TURIZEM#Ft.T. + število od 1−26, ki je razvrščalo članek v točno določeno kategorijo/razred. Označevalec konca besedila smo zamenjali z »',« tako, da smo ločili atributa *Article* in *Class*, ki smo ju kasneje definirali v glavi ARFF-datoteke.

Primer besedila pred in po pripravi na uvoz:

1. 'Napev je slovenski in nam je vsem zelo domač, a gre za 'tuji' oziroma neslovenski izdelek.#TURIZEM#Ft.T.2.
2. 'Napev je slovenski in nam je vsem zelo domač, a gre za \'tuji\' oziroma neslovenski izdelek.',2.

V tabeli 6 je prikazana izdelana glava ARFF-formata. V @RELATION smo najprej smiselno poimenovali datoteko. Sledita atributa *Article* tipa niz in *Class* tipa number, v katerem smo definirali vseh 26 razredov, vključno z razredom 0. Za oznako @DATA si sledijo preobdelani podatki v formatu: *'Article',Class* (glej primer besedila).

Tabela 6: Glava ARFF-datoteke, pripravljena na uvoz

```
@RELATION TouristCampusData
@ATTRIBUTE Article String
@ATTRIBUTE Class
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26}


@DATA
```

3.3.2 Predstavitev besedilnih dokumentov

Ker večina klasičnih tehnik strojnega učenja za tip vhodnih podatkov ne podpira besedilnih podatkov, je te podatke treba predstaviti v ustrezni obliki. Besedila so običajno predstavljena v obliki atributov oziroma matrike, kjer vrstice predstavljajo instance – v našem primeru posamezna besedila, stolpci pa atribute. Takšno predstavitev pogosto imenujemo model vektorskega prostora (*angl. Vector-Space-Model*) ali tudi vreča besed (*angl. Bag Of Words*), kjer je posamezen dokument predstavljen z vektorjem. Potem ko smo besedilo predstavili kot vrečo besed, lahko začnemo šteti tekstovne elemente. V WEKI je vreča besed, implementirana pod imenom *StringToWordVector,* in ponuja več nastavljivih parametrov.

Vidonij za ponazoritev zgoraj opisanega postopka uporabi primer dveh vzorčnih besedil:

- Ker danes sije toplo sonce, je zunaj toplo.
- Ker je danes zunaj toplo, bom zunaj igral košarko.

Kot lahko opazimo, je avtor za lažje razumevanje in skladnost z uporabljenim postopkom iz besedila odstranil vsa ločila in velike začetnice. V vreči besed pridobimo unijo vseh besed, ki se pojavijo v obeh besedilih, torej le enkrat brez ponovitev. Gre za množico besed: *ker, danes, sije, sonce, je, zunaj, toplo, bom, igral, košarko.* Množica predstavlja naš vektor besed, za katere nas zanima, kakšna je njihova pojavitev znotraj besedila. Pojavitev obeh besedil smo predstavili v tabeli 7.

Tabela 7: Primer štetja pojavitev besed v besedilu (Vir tabele: [31])

	ker	danes	sije	sonce	je	zunaj	toplo	bom	igral	košarko
B_1	1	1	1	1	1	1	2	0	0	0
B_2	1	1	0	0	1	2	1	1	1	1

Končna matrika M, kjer vsaka vrstica predstavlja eno besedilo, stolpci pa posamezne atribute oziroma besede, bi izgledala takole:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Enačba 1: Matrika pojavitve besed (Vir enačbe: [31])

Na poziciji matrike $M_{v,s}$ se nahaja število pojavitev besede, ki je na s-ti poziciji v vektorju vseh besed in v v-tem besedilu po vrsti. Če to predstavimo na primeru vzorčnih besedil, je vrednosti $M_{2,6}$ enaka 2, ker se v drugem besedilu na šestem mestu vektorja B_2 pojavi beseda *zunaj* v dveh primerih.

Pristop štetja posameznih besedil lahko razširimo tako, da štejemo pojavitev večjega skupka zaporednih besed, imenovanega *terka*. Na tak način pridobimo znanje o celotni strukturi stavkov in ne le o številu besed, ki so v besedilu. Poglejmo si opisan pristop na primeru prejšnjih dveh besedil. Tokrat bomo šteli pojavitve za dve besedi (*dvoterke*), ki v besedilu stojita skupaj. Dobimo pare zaporednih elementov, ki jih sestavimo v vektor: *(ker,*

danes), (danes, sije), (sije, toplo), (toplo, sonce), (sonce, je), (je, zunaj), (zunaj, toplo), (ker, je), (je, danes), (danes, zunaj), (toplo, bom), (bom, zunaj), (zunaj, igral), (igral, košarko). Preverimo pojavnost določenega para besed znotraj dveh zaporednih besed in na podlagi vektorjev zgradimo matriko M [31].

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Enačba 2: Matrika pojavitve dvoterk (Vir enačbe: [31])

Vrednost matrike $M_{2,7}$ je v tem primeru 1, kar pomeni, da v drugem besedilu obstaja ena terka – (*je, zunaj*) na sedmi poziciji vektorja B_2.

3.3.2.1 IDF

Poleg zgornjega načina ugotavljanja pojavitev (frekvenc) elementov znotraj posameznih besedil obstaja tudi ugotavljanje inverznega števila besedil. Ker prej opisani metodi ne upoštevata pogostosti nekega elementa v besedilu, je treba uporabiti postopek inverznega števila besedil, ki ugotavlja, kako razširjen je nek element znotraj vseh besedil (korpusa). Vrednost IDF ugotavljamo po enačbi 3, kjer nam N predstavlja število vseh besedil v naši zbirki (korpusu), b_e pa število besedil iz naše zbirke, ki vsebujejo eno pojavitev elementa *e* [31].

$$idf_e = log\left(\frac{N}{b_e}\right)$$

Enačba 3: IDF-formula (Vir enačbe: [31])

3.3.2.2 TF-IDF

Če metodo štetja pojavitev elementov združimo z inverznim številom dokumentov, dobimo združeni postopek, ki se imenuje TF-IDF. Nasprotno kot pri metodi štetja pojavitev elementov v posameznem besedilu, tukaj odpravimo problem, da so vsi elementi enakovredni, saj dobijo večjo težo tisti elementi, ki se velikokrat pojavljajo v manjšem številu besedil in ne tisti, ki se pojavljajo v velikem številu besedil [10]. Na ta način upoštevamo dodatno tudi dejstvo, da je nek element (recimo beseda) značilen samo za neko besedilo, v katerem se velikokrat pojavi, v preostalih besedilih pa se ta element ne pojavlja. Nasprotno pa je tudi primer, ko se nek element pojavlja recimo v vseh besedilih. Elementi, ki so pogosti za vsa besedila, nam namreč pri naši napovedi ne pomagajo preveč, saj nam ne

pripomorejo pri razločitvi med besedili. Po drugi strani pa nam pri razločitvi med različnimi besedili pomagajo elementi, ki so značilni samo za nekatera besedila iz naše množice besedil. Zato tudi ta metoda da takšnim elementom večjo težo. Formalno se po opisani metodi za vsak element vrednost izračuna po enačbi 4, kjer pomnožimo število pojavitev elementa *e* v dotičnem besedilu *b* z inverznim številom dokumentov (idf_e), v katerih se pojavi vsaj enkrat naš trenutni element *e* [31].

$$tf - idf_{e,b} = tf_{e,b} \times idf_e$$

Enačba 4: TF-IDF-formula (Vir enačbe: [31])

Vse zgoraj naštete metode predstavitve besedil pozabljajo na dejstvo, da je pomen besed odvisen od konteksta, v katerem se določena beseda nahaja, in da ima lahko ista beseda različen pomen. Prav tako nam taka predstavitev ne bo razrešila problema podpomenk in drugačnih povezav med različnimi izrazi. Zanemarita se tudi vrstni red besed in struktura besedila. Dokument je torej le seznam besed v poljubnem vrstnem redu [11].

3.3.3 Redukcija podatkov

V naši študiji primera imamo opravka z veliko količino besedil, ki obsega preko 41 MB tekstovnih podatkov. Potem ko smo uspešno aplicirali IDF- in TF-IDF-metodi na naše besedilo, je treba število dobljenih atributov kar se da zreducirati. Za ta namen smo uporabili postopek lematizacije, odstranjevanja nepomembnih besed in izbire atributov s pozitivno informacijsko vrednostjo.

S tem smo dosegli:

- poenostavitev modela in razumevanja s strani uporabnikov/raziskovalcev,
- krajši učni proces modela,
- reševanje prekletstva dimenzionalnosti (*angl. Course of Dimensionality*), ko zaradi kombinacije povečevanje dimenzionalnost in pomanjkanja atributov klasifikator izgublja na točnosti,
- izboljšanje generalizacije z zmanjšanjem prekomernega prileganja (*angl. Overfitting*).

3.3.3.1 Lematizacija

Vidoni v svojem delu ugotavlja, da se v primeru, ko imamo osnovno neobdelano obliko besedil, pojavljajo besede z istim ali podobnim pomenom, ki so podobno zapisane. Ljudje z

razpoznavo pomena sicer nimamo težav, ker se opiramo tudi na kontekst, v katerem se beseda nahaja. Problem nastane pri računalnikih, ki z uporabo postopkov, opisanih v poglavju »Predstavitev besedilnih podatkov«, tretirajo vsako različno zapisano besedo različno, pa najsi bo to samo razlika v veliki začetnici.

Problem je prisoten v mnogih jezikih v slovenskem jeziku, pa še toliko bolj, ker imajo osnovne besede še več možnih izpeljank na račun dvojine, množine, sklonov in spregane oblike besed. Pred začetkom tekstovnega rudarjenja je zato smiselno besedilo normalizirati oziroma ga transformirati v osnovne leme. Ročni in računalniški postopek lematizacije zahteva uporabo različnih pravil in slovarjev. Spodaj je prikazan primer osnovne in lematizirane oblike besedila:

1. Osnovna oblika: Ker sem zbolel, bom danes ostal doma.
2. Lematizirana oblika: Ker biti zboleti, biti danes ostati doma.

S primera je razvidno, da so besede: *sem, zbolel, bom ostal* sedaj v svoji normalizirani obliki in da se na primeru osnovnih besed *sem* in *bom* že vidi rezultat zmanjševanja atributov − besed v skupni besedi *biti*. Lematizacija se pogosto uporablja v rudarjenju besedilnih dokumentov. Na ta način zmanjšamo število različnih besed v korpusu, vendar pa s tem izgubimo dodatne informacije, kot sta sklon in slovnični čas. V večini primerov nas zanima le pomen besed, zato je ta postopek zelo razširjen [31].

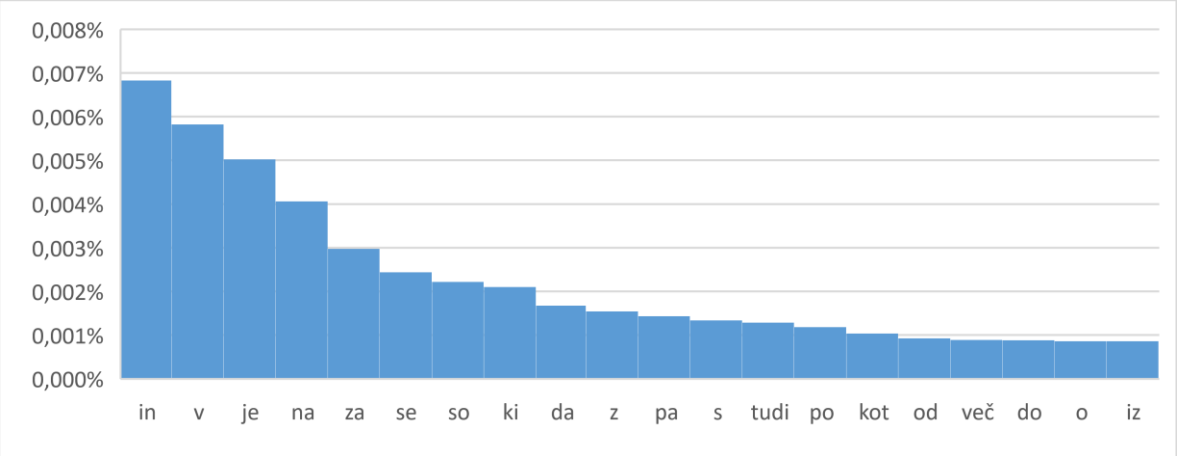
3.3.3.2 Odstranjevanje nepomembnih besed

Funkcijske besede, med katere spadajo vezniki, zaimki, členki, predlogi in pomožni glagoli, spadajo med nepomembne besede, nosijo majhno količino informacij, po drugi strani pa zaradi svoje visoke frekvence slabšajo točnost učnega modela in hitrost njegove izdelave. Takšne besede je smiselno izločiti iz procesa tekstovnega rudarjenja. Za tuje jezike obstaja kar nekaj seznamov z nepomembnimi besedami, na žalost slovenskega ni med njimi. Seznam besed, ki smo jih izločili, smo pripravili na podlagi frekvence najpogostejših besed znotraj prečiščene različice besedil turističnega korpusa. Izmed 100 najpogostejših besed smo se odločili za izločanje prvih 20 besed nelematizirane in lematizirane različice korpusa v kombinaciji z in brez razreda 0. Pri odstranjevanju besed je treba biti posebej pozoren na besede, ki so del fraz in bi v primeru odstranjevanja imele negativni učinek na točnost modela.

Nelematiziran korpus

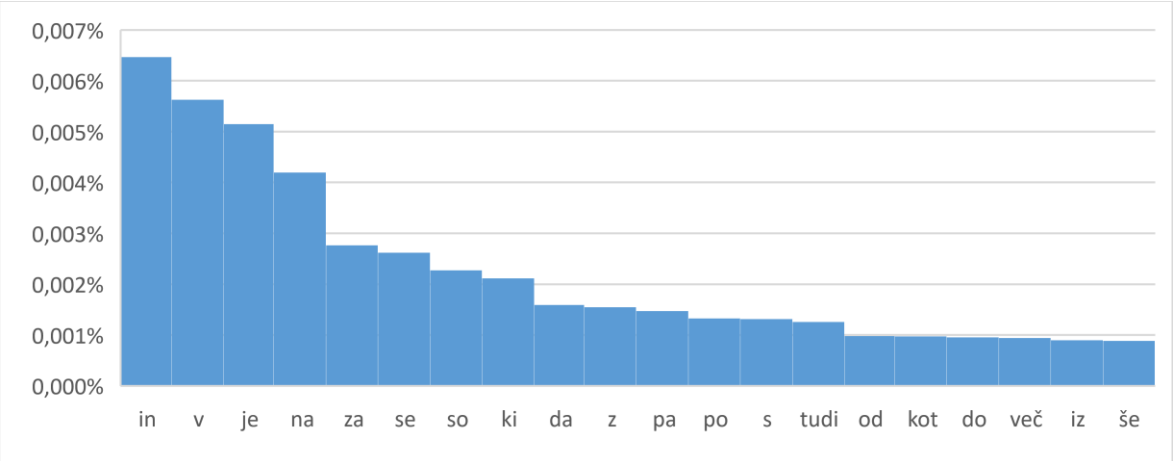
Prvih 20 izmed 480 besed v tabeli 8 predstavlja 45,45 % vseh besed v prečiščenem nelematiziranem korpusu z vsemi razredi. V primeru uporabe 100 besed pokrijemo 74,8 % vseh besed.

Tabela 8: Frekvenca nepomembnih nelematiziranih besed – vsi razredi



Prvih 20 izmed 476 besed v tabeli 9 predstavlja 45,39 % vseh besed v prečiščenem nelematiziranem korpusu brez razreda 0. V primeru uporabe 100 besed pokrijemo 74,7 % vseh besed. Razlika med prejšnjo različico z vključenim razredom 0 je v pojavu besede »še«.

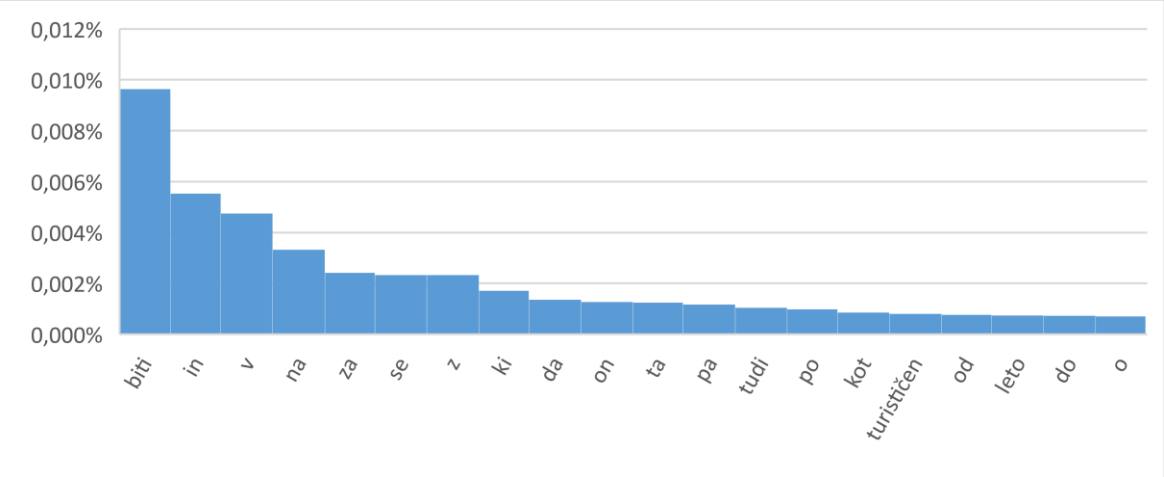
Tabela 9: Frekvenca nepomembnih nelematiziranih besed – brez 0



Lematiziran korpus

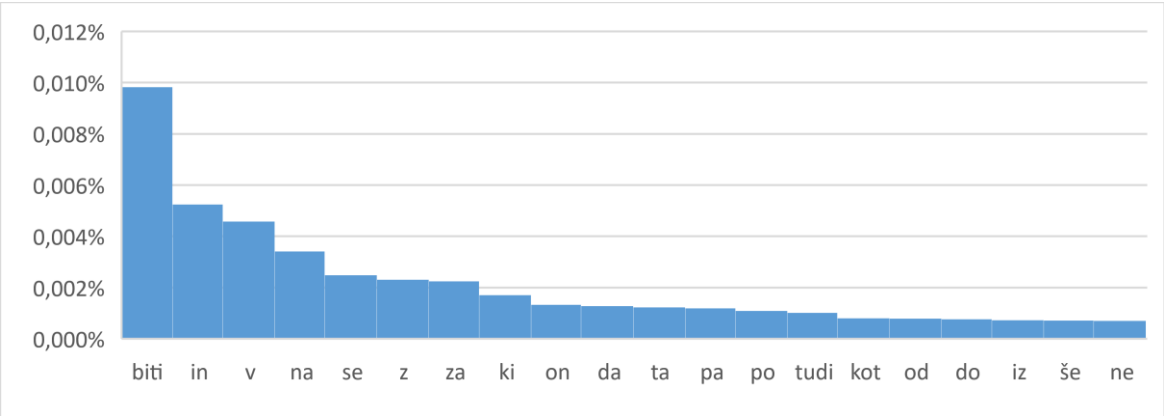
Pri lematiziranem korpusu prihaja do večje spremembe v frekvenci besed s pojavom nedovršnih glagolov (*biti*), zaimkov (*on, ta*) in pridevnikov (*turističen*) kot rezultat lematizacije besed. Prvih 20 izmed 524 besed v tabeli 10 predstavlja 43,75 % vseh besed v prečiščenem lematiziranem korpusu z vsemi razredi. V primeru uporabe 100 besed pokrijemo 69,72 % vseh besed.

Tabela 10: Frekvenca nepomembnih lematiziranih besed – vsi razredi



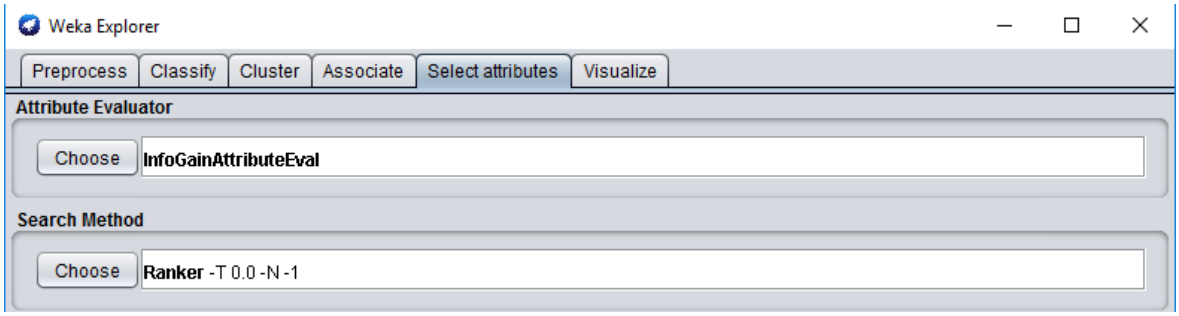
Prvih 20 izmed 524 besed v tabeli 11 predstavlja 43,55 % vseh besed v prečiščenem lematiziranem korpusu brez razreda 0. V primeru uporabe 100 besed pokrijemo 69,04 % vseh besed.

Tabela 11: Frekvenca nepomembnih lematiziranih besed – brez 0



3.3.3.3 Izbira atributov s pozitivno informacijsko vrednostjo

Za dodatno zmanjšanje števila atributov smo v WEKI-tehniki uporabili izbor atributov, poimenovanih *InfoGainAttributeEval*, ki ocenjuje vrednost atributa z merjenjem informacijskega prispevka glede na razred. Pri tem uporablja metodo *Ranker* za razvrščanje atributov na podlagi njihove individualne evalvacije. *Ranker* ponuja možnost nastavitve meje (*angl. Treshold*), s katero določamo, kateri atributi so za nas sprejemljivi. Mejo smo nastavili na 0, kar pomeni, da sprejemamo le atribute, ki imajo pozitiven informacijski prispevek [7].



Slika 11: Nastavitve tehnike InfoGainAttributeEval

3.4 PREDSTAVITEV TESTIRANIH ALGORITMOV

Pri izgradnji napovednih modelov smo uporabili tehniko tekstovne klasifikacije, s katero smo med seboj primerjali več različic turističnih korpusov v kombinaciji z zgoraj opisanimi postopki odstranjevanja in zmanjševanja atributov. V nadaljevanju so opisani algoritmi, ki smo jih uporabili pri izgradnji napovednih modelov.

3.4.1 ZeroR

ZeroR je najpreprostejša klasifikacijska metoda, ki temelji na frekvenčni tabeli. ZeroR klasifikator preprosto napoveduje večinsko kategorijo (razred). Čeprav ZeroR nima napovedne moči, ga je koristno uporabiti kot izhodišče za primerjavo zmogljivost drugih klasifikacijskih algoritmov.

Če algoritem ZeroR apliciramo na nelematiziran neprečiščen korpus z vsemi razredi, bomo dobili napoved večinskega razreda tako, da bomo delili število instanc največjega razreda s številom vseh instanc. ZeroR bo pravilno napovedal večinski razred z natančnostjo 36,75 %, kar je dejansko enako deležu besedil, ki pripadajo razredu 0 [25].

3.4.2 OneR

OneR na kratko ali »One Rule« je preprost, vendar točen klasifikacijski algoritem, katerega naloga je najti atribut – predikator z najmanjšo napovedno napako. Če želimo ustvariti pravilo za predikator, moramo konstruirati tabelo frekvenc za vsak predikator. Natančnost OneR algoritma se lahko primerja z natančnostjo najnaprednejših klasifikacijskih algoritmov, pri čemer so konstruirana pravila veliko lažje razumljiva. Poglejmo si koncept OneR algoritma na primeru s slike 12 [24].

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

Slika 12: OneR primer izračuna najmanjše napovedne napake (Vir slike: [25])

Na voljo imamo 4 atribute oz. predikatorje, katerih kombinacija odloča o tem, ali bomo igrali golf ali ne. Če pogledamo frekvenčno tabelo za predikator vremenska napoved (*ang. Outlook*), vidimo, da imamo 2 napaki v 5 zapisih za sončno vreme, 0 napak od 4 zapisov za oblačno vreme in 2 napaki od 5 zapisov za deževno vreme. Skupaj to znese 4 napake od 14 zapisov, kar predstavlja 28,57-odstotni delež napake. Za atribut vlažnost (*angl. Humidity*) imamo isti delež napake, pri temperaturi (*angl. Temperature*) in vetrovnosti (*angl. Windy*) pa najdemo 5 napak od 14 zapisov, kar predstavlja 35,71-odstotni delež napake. Ker iščemo atribut z najmanjšo napovedno napako in ker imata vremenska napoved in vlažnost enak delež napake, se OneR med omenjenima atributoma odloči za naključno izbiro [22].

3.4.3 Naivni Bayes

Gre za klasifikacijsko tehniko, ki temelji na Bayesovem teoremu z domnevo neodvisnosti med prediktorji. Poenostavljeno rečeno, klasifikator Naivni Bayes predpostavlja, da je prisotnost določene lastnosti v razredu nepovezana s prisotnostjo katerekoli druge lastnosti. Zaradi domneve o neodvisnosti je Naivni Bayes lahko prilagodljiv in zmožen uporabe visoko dimenzijskih lastnosti z omejenimi učnimi podatki. To je uporabno za mnoge zbirke

podatkov iz realnega sveta, kjer je količina podatkov majhna v primerjavi s številom lastnosti posameznega podatka [16].

Primer: Jabolko se lahko šteje za sadje, če je rdeče, okroglo in ima 3 cm premera. Tudi če so te lastnosti odvisne druga od druge ali znotraj razreda obstajajo še kake druge povezane lastnosti, pa vse neodvisno prispevajo k verjetnosti, da je to sadje jabolko. Pravimo, da gre za naivno klasifikacijsko tehniko, saj je v praksi praktično nemogoče dobiti primer, kjer so lastnosti neodvisne druga od druge. Naivni Bayes je enostaven za gradnjo in še posebej uporaben za zelo velike zbirke podatkov. Kljub svoji preprostosti prekaša bolj sofisticirane klasifikacijske metode. Primeri sodobnih aplikacij, ki uporabljajo Naivni Bayes, vključujejo filtriranje neželene pošte, avtomatske medicinske diagnoze, obdelavo medicinskih slik in prepoznavanje čustev.

Naloga Bayesovega klasifikatorja je izračunati pogojne verjetnosti za vsak razred pri danih vrednostih atributov za novi primer, ki ga želimo klasificirati. Naivni Bayesov klasifikator predpostavlja pogojno neodvisnost vrednosti atributov pri danem razredu:

$$p(v_1, v_2, \ldots v_n|c) = \prod_i p(c|v_i)$$

Enačba 5: Pogojna neodvisnost atributov pri danem razredu (Vir enačbe: [12])

$$p(c|v_1, v_2, \ldots v_n) = p(c) * \prod_i \frac{p(c|v_i)}{p(c)}$$

Enačba 6: Naivna Bayesova formula (Vir enačbe: [12])

Naivni Bayesov klasifikator nov primer klasificira tako, da za vsak možni razred ci po naivni Bayesovi formuli izračuna verjetnost, da primer (v1, v2, ...vn) pripada razredu ci, kar zapišemo p(ci|v1, v2, ...vn). Primer klasificira v razred z največjo verjetnostjo [12].

3.4.4 J48

J48 je implementacija C4.5-algoritma za gradnjo odločitvenega drevesa, ki ga je razvil Ross Quinlan. Gre za izboljšano verzijo ID3-algoritma. C4.5 izbere atribut, ki najbolje loči množico primerov v podmnožice glede na razred. Merilo za izbiro atributa je razmerje informacijskega prispevka, tako da je izbran atribut z največjim razmerjem informacijskega prispevka. Na podlagi izbranega atributa se učni primeri razdelijo na podmnožice. Algoritem lahko obravnava tako zvezdne kot diskretne atribute. Da lahko obravnava zvezdne atribute, ustvari mejno vrednost in nato razdeli trenutno vozlišče v dve kategoriji – eno za vrednosti atributov nad zastavljenim pragom in drugo za manjšo ali enako vrednost atributov. Podobno

kot v ID3 algoritmu C4.5 razvrsti atribute v vsakem vozlišču z namenom iskanja najboljšega razdelitvenega atributa. Razdelitev je končana, ko dosežemo enega izmed spodnjih pogojev:

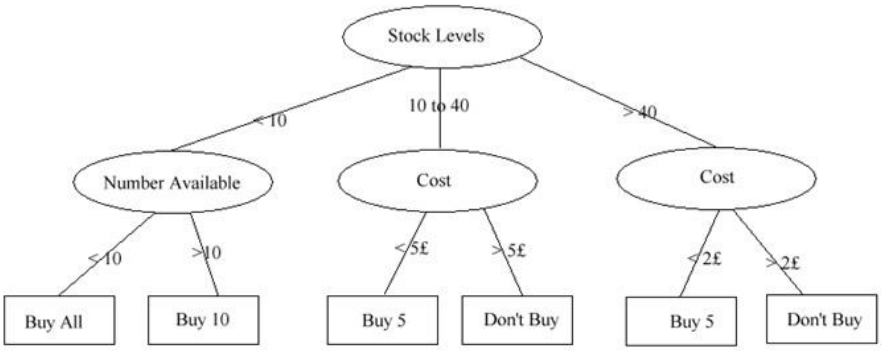
- vsi vzorci v listu pripadajo istemu razredi,
- nobena izmed lastnosti ne zagotavlja informacijskega prispevka,
- število vzorcev, ki jih želimo razdeliti, pade pod definirano mejo.

Prednosti C4.5-algoritma so lažja interpretacija in implementacija algoritma, uporaba diskretnih in zvezdnih vrednosti, odpornost na šum in majhna časovna zahtevnost. Slabosti pa ta, da lahko že manjša variacija v podatkih vodi do različnih odločitvenih dreves v primeru, ko so spremenljivke blizu ena drugi. Ob enem ne daje dobrih rezultatov na majhni učni množici. Primeren je za reševanje različne problematike, ker lahko obravnava numerične in manjkajoče vrednosti. Vključuje metodo za obrezovanje dreves (*angl. Tree Prunning*), ki zmanjšuje napake napačne klasifikacije. Te so vzrok šuma ali prevelikega prilagajanja znotraj učne množice. V primerjavi z ID3-algoritmom proces obrezovanja dreves vodi k manjšim drevesom, preprostejšim pravilom in lažji interpretaciji.

Psevdokoda:

- Poišči atribut z največjim razmerjem informacijskega prispevka.
- Kreiraj odločitveno vozlišče, ki se razdeli glede na izbrani atribut.
- Rekurzivno ponavljaj postopek na podmnožicah in dodajaj nova vozlišča.
- Ustavi se, ko prideš do enega izmed zgornjih zaustavitvenih pogojev.

Komercialni naslednik algoritma C4.5 je C5.0/See5, ki je veliko hitrejši kot C4.5, pomnilniško učinkovitejši in pri izgradnji kreira manjša odločitvena drevesa.

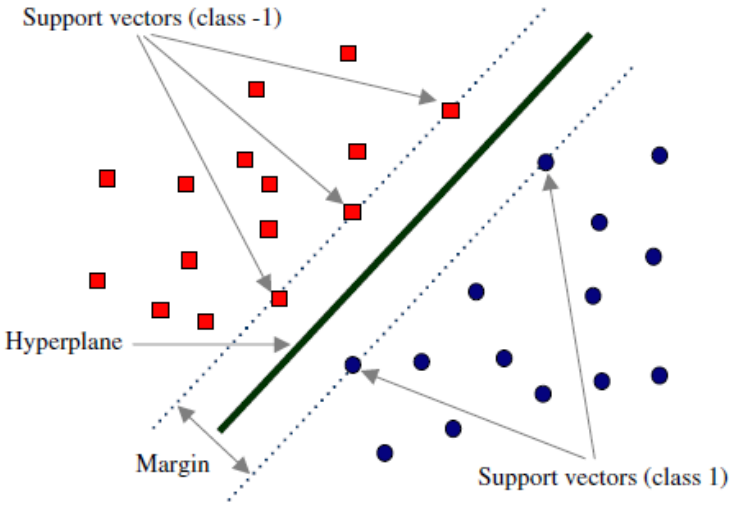


Slika 13: Primer C4.5 algoritma (Vir slike: [38])

3.4.5 SMO

Poglejmo si naprej metodo podpornih vektorjev (*angl. SVM*), ki velja za eno najbolj uspešnih med za klasifikacijo in regresijo. SVM za izgradnjo modela uporablja vse razpoložljive atribute, tudi tiste manj pomembne in jih z linearno kombinacijo uporabi za napovedovanje odvisne spremenljivke. Pomemben je predvsem način kombiniranja atributov, manj pa izbira atributov saj bo metoda z ustrezno kombinacijo izluščila želeno informacijo. Primerna je za učenje na velikih množicah primerov. Z njo dosegamo visoko točnost, vendar jo težko interpretiramo.

Ideja SVM je znotraj atributov postaviti optimalno hiperravnino, ki je definirana kot tista, ki je enako in najbolj oddaljena od dveh najbližjih primerov dveh razredov. Če imamo več razredov, postopek ponovimo za vsak razred, ki ga ločujemo od ostalih. Najbližjim primerom optimalne hiperravnine pravimo podporni vektorji, razdalji hiperravnine od podpornih vektorjev pa rob. Optimalna hiperravnina je torej tista, ki ima maksimalni rob [19].



Slika 14: Metoda podpornih vektorjev (Vir slike: [42])

Sekvenčna minimalna optimizacija (*ang. Sequential Minimal Optimization*) je algoritem za reševanje problema kvadratičnega programiranja (krajše QP), ki se pojavlja med učenjem podpornih vektorjev. Algoritem je leta 1998 iznašel John Platt iz podjetja Microsoft Research in je široko uporabljen pri učenju metode podpornih vektorjev. Ker učenje z metodo podpornih vektorjev zahteva reševanje obsežnega optimizacijskega problema, QP SMO razbije QP v najmanjšo možno serijo QP. Ti manjši QP se nato rešujejo analitično, s čimer zmanjšamo časovno potratnost numerične optimizacije QP kot notranje zanke. Količina pomnilnika, ki jo SMO potrebuje za učenje, je linearna z učno množico, kar

omogoča uporabo veliki učnih množic. Prejšnje metode za reševanje QP se bile veliko bolj kompleksne [19].

3.4.6 Random forest

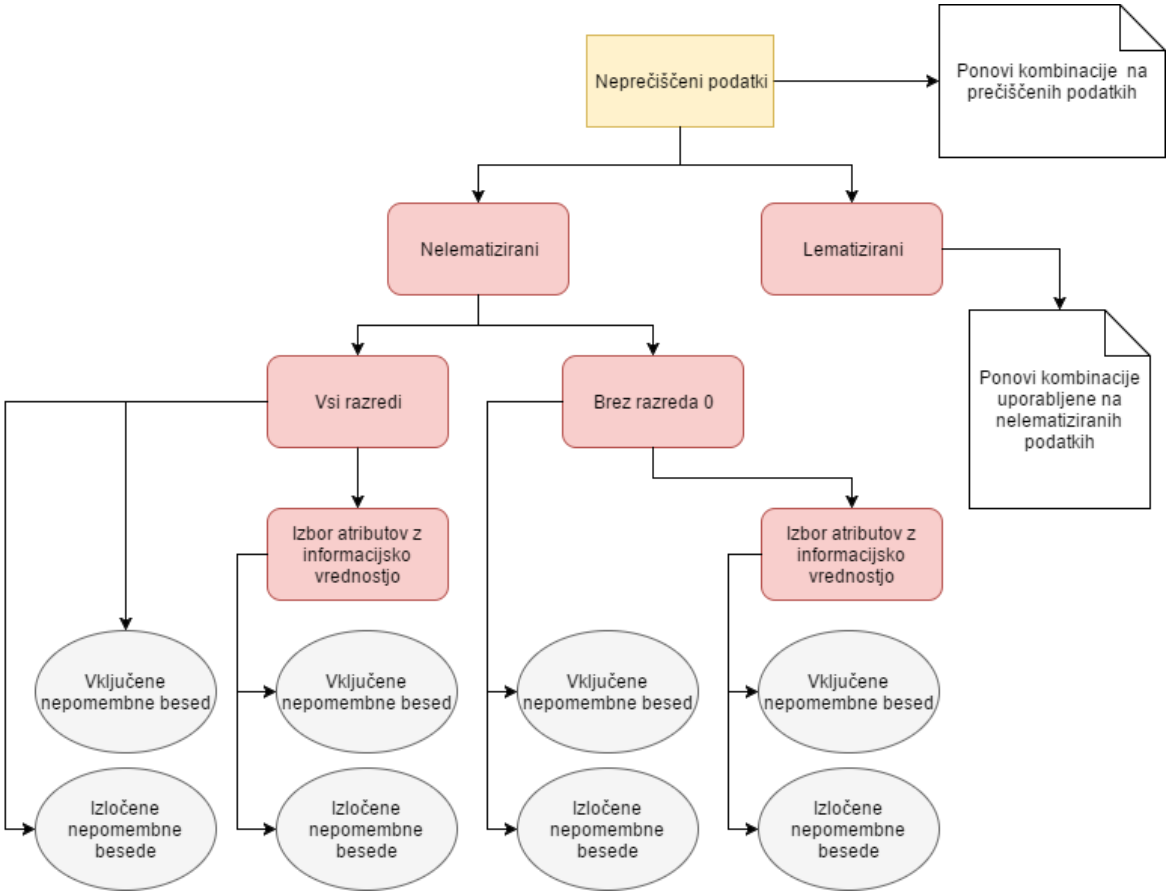
Naključni gozd je ansambelska učna metoda (Algoritmi, ki se modeliranja podatkov lotevajo z združevanjem več različnih modelov.), ki se uporablja za klasifikacijo, regresijo in druge naloge. Najprej ga je predlagal Tin Kam Ho, ki so jo nadalje razvili Leo Breiman (Breiman, 2001) in Adele Cutler. Random Forest zgradi niz odločitvenih dreves. Metoda naključnih gozdov je namenjena izboljšanju napovedne točnosti drevesnih modelov. Ideja je generirati zaporedje odločitvenih dreves, tako da se pri izbiri najboljšega atributa v vsakem vozlišču naključno izbere majhno število atributov, ki vstopajo v izbor za najboljši atribut. Breiman je predlagal naključno izbiro toliko atributov v vsakem vozlišču, kolikor znaša logaritem števila atributov plus 1. To število je lahko tudi 1, kar pomeni popolnoma naključno izbiro atributa v vsakem vozlišču vsakega drevesa.

Vsako drevo se uporabi pri klasifikaciji novega primera po metodi glasovanja – vsako drevo ima en glas, ki ga nameni razredu v, katerega bi klasificiralo nov primer. Število zgrajenih dreves je ponavadi 100 ali več. Iz vseh glasov dobimo verjetnost porazdelitev po vseh razredih.

Metoda zaradi robustnosti dosega točnost, ki je primerljiva z najboljšimi algoritmi, slabo stran pa predstavlja otežena razlaga odločitve, saj je množica 100 ali več dreves nepregledna in nerazumljiva za uporabnika. Končni model temelji na osnovi večinskega glasovanja posamično razvitih dreves v gozdu [12].

3.5 IZGRADNJA NAPOVEDNIH MODELOV

Potem ko so bila besedila ustrezno prečiščena, lematizirana in ločena na tista z vsemi in brez razreda 0, se v tem poglavju lotevamo izgradnje napovednih modelov za vsako izmed naštetih različic turističnega korpusa. Skupaj smo modelirali 32 kombinacij, kar za 5 algoritmov z dvema načinoma evalvacije znaša 320 zgrajenih modelov. Diagram modeliranja je razviden iz slike 15, kjer smo zaradi lažje vizualizacije predstavili kombinacije, ki so bile izvedene nad neprečiščenimi podatki turističnega korpusa. Isti postopek smo ponovili tudi na prečiščenih podatkih.

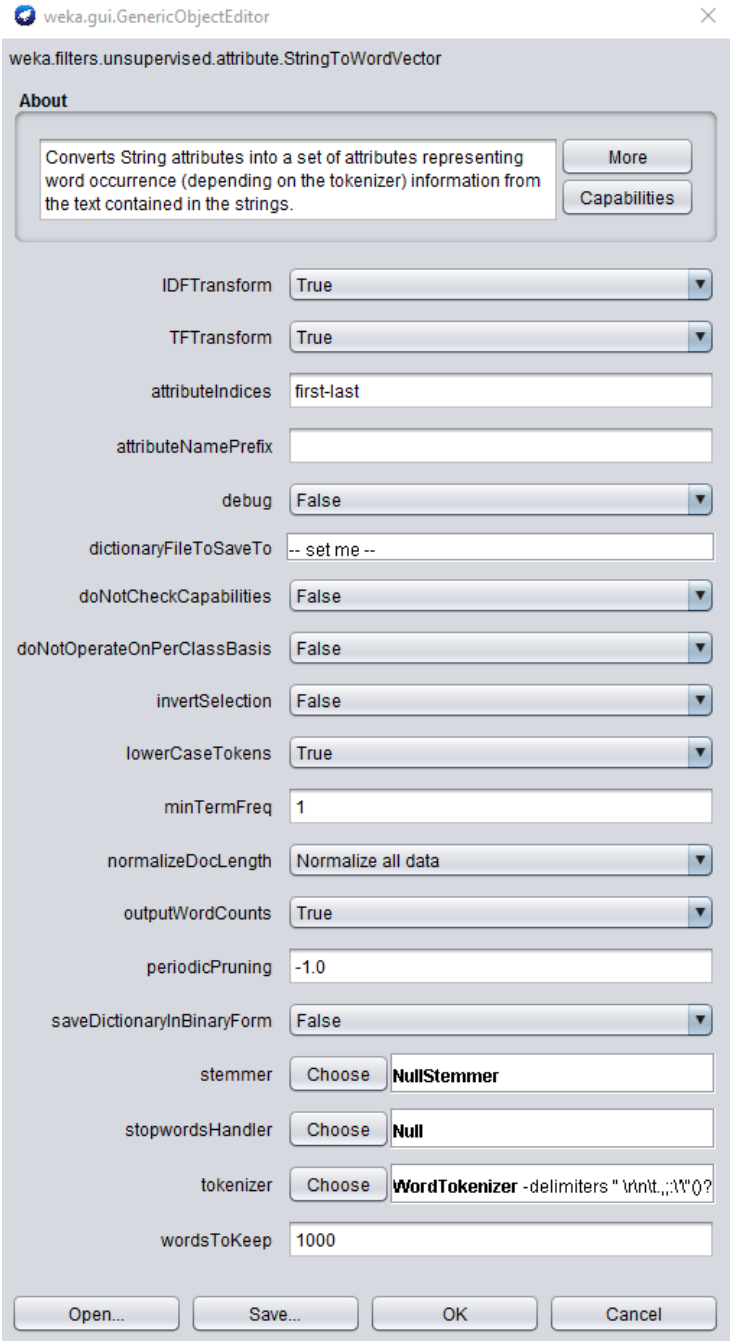


Slika 15: Diagram kombinacij modeliranja

3.5.1 Nastavitev parametrov StringToWordVector filtra

Na zavihku Preprocesiranje (*angl. Preprocess*) znotraj okna Raziskovalec (*angl. Explorer*) smo najprej izbrali želeno predpripravljeno različico turističnega korpusa, ki jo je treba z nenadzorovanim filtrom za atribute *StringToWordVector* spraviti v ustrezno obliko za nadaljnjo obdelavo. Filter smo konfigurirali tako, da smo omogočili (*angl. True*) parametre IDFTransform, TFTransform in outputWordCounts. Dolžino dokumenta normaliziramo (*angl. NormalizeDocLenght*) z nastavitvijo »Normaliziraj vse podatke« (*angl. Normalize All Data*). Ostale parametre smo pustili nastavljene na privzeto vrednost.

V primeru, ko izločamo nepomembne besede, je treba v parametru *stopwordsHandler* uporabiti metodo *WordsFromFile*, kjer pokažemo na tekstovno datoteko, v kateri so definirane nepomembne besede. Kot privzeta nastavitev *stopwordsHandler* nepomembne besede iz modela niso izključene.



Slika 16: Konfiguracija filtra StringToWordVector

Ko zaključimo z nastavitvijo parametrov, poženemo filter nad vhodnimi podatki. V trenutni relaciji (*angl. Current Relation*) se nam izpišejo podatki o imenu, številu atributov, instancah in seštevku uteži. V razdelku z atributi (*angl. Attributes*) lahko opazimo, da je atribut *class* na prvem mestu, kar pa za nadaljnjo aplikacijo algoritmov nad podatki ni pravilno, saj se v Weki razred vedno nahaja na zadnjem mestu med atributi. V načinu urejanja (*angl. Edit*) z desnimi klikom izberemo *class* in izberemo možnost »Atribut kot

razred« (*angl. Attribute As Class*). Podatki so s tem pripravljeni na korak podatkovnega rudarjenja z izbranimi algoritmi.

3.5.2 Uporaba AttributeSelection filtra

Z uporabo nadzorovanega *AttributeSelection* filtra in evalvatorja *InfoGainAttributeEval* želimo doseči zmanjšanja števila atributov in povečati točnost napovednega modela. Pri tem z iskalno metodo *Ranker* izločamo atribute, ki imajo negativno informacijsko vrednost. Bistveno je, da omenjen filter apliciramo po vektorizaciji besedil, saj ne podpira atributov tipa niz. Po končam postopku apliciramo nad podatke izbrani algoritem, s čimer začnemo s fazo podatkovnega rudarjenja.

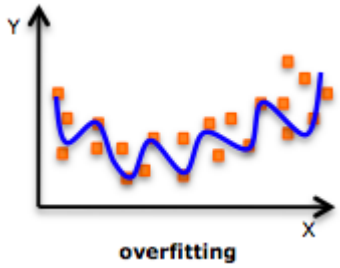
Slika 17: Konfiguracija AttributeSelection filtra

4 EVALVACIJA USPEŠNOSTI NAPOVEDNIH MODELOV

Namen napovednega modeliranja je zgraditi modele, ki imajo dobre rezultate pri napovedovanju novih, še nepoznanih podatkov, zato je pomembno, da uporabimo robustne tehnike za učenje in evalvacijo modelov na učnih množicah podatkov. Zanesljivejša kot je ocena zmogljivosti našega modela, večja je uporabnost in možnost implementacije modela v praksi.

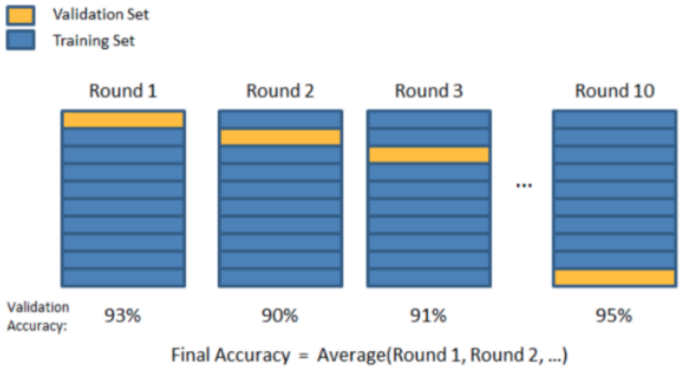
Izgradnje modelov smo se lotili s šestimi izbranimi algoritmi, od katerih *ZeroR* predstavlja začetno izhodišče (*angl. Baseline*) za primerjavo zgrajenega modela. Ostale klasifikacijske algoritme smo testirali v naslednjem vrstnem redu: *OneR, Naive Bayes, J48, SMO in Random Forest*. Znotraj posameznega algoritma so bili uporabljeni privzeti parametri, ki jih je ponujala Weka. Evalvacija modelov je potekala z:

- **Učno množico** (*angl. Training Set*): testiranje se izvede nad celotno učno množico. Obstaja velika verjetno prevelikega prilagajanja (*angl. Overfitting*) podatkom iz učne množice, zaradi česar prihaja do pretirane točnosti modela. Ta način evalvacije smo uporabili samo pri izhodiščnem *ZeroR* algoritmu.



Slika 18: Primer pretiranega prilagajanja modela podatkom (Vir slike: [3])

- **Prečno preverjanje** (*angl. Cross-Validation*): Učno množico razdelimo na *n* enakih delov in *n-1* uporabimo za učno množico preostalega enega pa za validacijo. Postopek ponovimo n-krat, pri tem vsakič spremenimo učno množico za validacijo. Končna natančnost modela je rezultat povprečja validacij n-krogov. Velja za zlati standard ocenjevanja uspešnosti modela, vendar je časovno in prostorsko zahtevna.



Slika 19: Primer prečne validacije na 10 vzorcih (Vir slike: [3])

- **Odstotkovna delitev** (*angl. Percentage Split*): Podatke naključno razdelimo na učno, preostanek pa na validacijsko množico. Nato izvedemo evalvacijo modela. V primeru Percentage Split 66 % gre za razmerje 66 % : 34 % [3].

Pri evalvaciji klasifikacijskih algoritmov za strojno učenje v Weki lahko v končnem povzetku uspešnosti najdemo veliko informacij o točnosti našega modela. Izpostaviti je treba tri pomembne razdelke z informacijami, ki nudijo vpogled v uspešnost klasifikacijskega algoritma:

- **Točnost klasifikacije** (*angl. Classification Accuracy*)**.**
    - Pravilno in nepravilno klasificirane instance. Gre za delež pravilnih ali nepravilnih napovedi, ki je predstavljen v absolutnem formatu, pri čemer je 100 % najboljši rezultat, ki ga algoritem lahko doseže.
    - Kappa statistika je pomembna mera za uspešnost klasifikatorjev, še posebno takrat, ko imamo večrazredne in neuravnotežene podatke. Pove nam, koliko je klasifikator boljši od tistega, ki naključno razvršča na podlagi frekvence posameznega razreda [13].
- **Točnost po razredih** (*angl. Accuracy by Class*)**.** Statistični pregled rezultatov po razredih, ki nudi naslednje informacije:
    - Pravilno pozitivna ocena (*angl. TP Rate*) predstavlja delež vseh primerkov, ki so bili razvrščeni v pravilni razred x. V matriki te elemente najdemo na diagonali. *Recall* predstavlja isto izračun kot *TP Rate*.
    - Nepravilno pozitivna ocena (*angl. FP Rate*) predstavlja delež primerkov, ki pripadajo razredu x, vendar so bili razvrščeni v preostale razrede. To so vsi primerki, ki se ne nahajajo na diagonali matrike.

- Natančnost (*angl. Precision*) predstavlja delež primerkov, ki resnično pripadajo razredu x med vsemi tistimi, ki so bili klasificirani kot pripadniki razreda x.
- F-mera (*angl. F-Measure*) predstavlja kombinirano mero *natančnosti* in *recall-a* uporabno pri primerjavi različnih klasifikatorjev [49].
- ROC-krivulja (*angl. ROC Area*) predstavlja graf resnično pozitivnega deleža (y-os) proti lažno pozitivnem deležu (x-os) za vsak posamezen možen klasifikacijski prag. Krivulja nam pove uspešnost klasifikacijskih metod. Z njeno pomočjo lahko določimo klasifikacijski prag, ki ustreza našim zahtevam [44].

- **Matrika pravilnih in napačnih razvrstitev** (*angl. Confusion Matrix*)**.** Tabela, ki omogoča pregled nad klasifikacijo podatkov po razredih. Uporabno za analizo napak algoritma.

4.1 PREGLED REZULTATOV

Za potrebe končnega odgovora o tem, kateri izmed algoritmov daje najboljše rezultate na študiji našega primera in ali dejavniki, kot so lematizacija, nepomembne besede in besede brez informacijske vrednosti, vplivajo na točnost napovednega modela, je bilo treba skupaj evalvirati 320 modelov. V nadaljevanju je prvotni korpus zaradi sistematičnosti razdeljen na podpoglavja:

- neprečiščeni nelematizirani podatki,
- neprečiščeni lematizirani podatki,
- prečiščeni nelematizirani podatki,
- prečiščeni lematizirani podatki.

Vsaka izmed naštetih različic vključuje testiranje z vključenim razredom nič in brez razreda nič. V vsaki tabeli najdemo verzijo z izločenimi nepomembnimi besedami in verzijo z vsemi besedami. Isti princip je uporabljen za testiranje različice, kjer so v modeliranje vključene zgolj besede s pozitivno informacijsko vrednostjo. Točnost napovednega modela, ki obenem predstavlja število pravilno razvrščenih instanc, je podana v odstotkih. Z zeleno barvo je v tabelah označen najboljši rezultat algoritma, ki ne vključuje izločanja besed z negativno informacijsko vrednostjo, z rumeno pa najboljši rezultat, ki to opcijo vključuje.

4.1.1 Neprečiščeni nelematizirani podatki

Model z vključenim razredom nič

Najboljši rezultat dosega algoritem SMO v CV10 načinu evalvacije z rezultatom 51,56 %. Sledi mu različica z izločenimi nepomembnimi besedami z 51,11 %. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, OneR, J48 in Naive Bayes.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 966 atributov v osnovnem načinu in 960 atributov v načinu brez nepomembnih besed. To predstavlja 11,89 % (8127 proti 966) začetnih atributov v prvem primeru oz. 11,73 % (8182 proti 960) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 51,12 %, je dosežen pri algoritmu SMO v Percentage Split 66 % načinu evalvacije. Sledi mu SMO različica z izločenimi nepomembnimi besedami v CV10 načinu evalvacije – 51,11 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 14,81 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute z pozitivno informacijsko vrednostjo znaša 0,37 % v prid osnovnemu.

Tabela 12: Primerjava algoritmov neprečiščenih nelematiziranih podatkov z vključenim razredom nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
	Training set	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	36.75	38.20	39.08	13.59	14.12	38.74	39.71	51.56	49.75	43.05	43.91
Stowords	/	38.33	39.59	13.67	14.24	39.29	37.95	51.11	49.94	43.13	43.79
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	37.99	39.9	11.28	11.12	42.64	42.60	50.90	51.19	46.84	46.42
Stopwords	/	37.99	37.64	11.28	11.61	41.98	42.47	51.11	50.56	47.21	47.3

Model brez razreda nič

Najboljši rezultat dosega algoritem SMO z izločenimi nepomembnimi besedami v CV10 načinu evalvacije z rezultatom 56,39 %. Sledi mu osnovna različica s 56,25 % v CV10 načinu evalvacije. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, Naive Bayes in OneR.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 953 atributov v osnovnem načinu in 946 atributov v načinu brez nepomembnih besed. To predstavlja 11,76 % (8100 vs, 953) začetnih atributov v prvem primeru oz. 11,60 % (8154 proti 946) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 53,63 %, je dosežen pri algoritmu SMO v CV10 načinu evalvacije. Sledi mu SMO različica z izločenimi nepomembnimi besedami v CV10 načinu evalvacije – 53,59 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO-algoritma znaša 42,33 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 2,76 % v korist osnovnemu.

Tabela 13: Primerjava algoritmov neprečiščenih nelematiziranih podatkov brez razreda nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	14.06	14.50	14.29	18.52	20.04	40.91	38.79	56.26	54.76	42.26	40.87
Stowords	/	14.81	14.09	18.82	18.95	40.57	38.09	56.39	54.76	42.33	40.97
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	14.50	15.48	17.27	18.85	43.51	42.76	53.63	52.08	49.88	48.81
Stopwords	/	15.35	15.58	17.23	18.65	43.85	40.97	53.59	52.38	50.79	48.31

4.1.2 Neprečiščeni lematizirani podatki

Model z vključenim razredom nič

Najboljši rezultat dosega algoritem SMO z izločenimi nepomembnimi besedami v CV10 načinu evalvacije z rezultatom 51,74 %. Sledi mu osnovna različica z 51,25 % v CV10

načinu evalvacije. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, OneR in Naive Bayes.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 784 atributov v osnovnem načinu in 775 atributov v načinu brez nepomembnih besed. To predstavlja 13,17 % (5954 proti 784) začetnih atributov v prvem primeru oz. 11,60 % (5978 proti 775) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi − J48 in Random Forest. Najboljši rezultat, ki znaša 52,06 %, je dosežen pri algoritmu SMO v CV10 načinu evalvacije. Sledi mu SMO različica z izločenimi nepomembnimi besedami v CV10 načinu evalvacije – 51,87 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 15,34 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 0,32 % v korist drugemu. Različica z atributi, ki imajo pozitivno informacijsko vrednost, je bila tu edino boljša od različice z vsemi besedami med vsemi testiranimi modeli.

Tabela 14: Primerjava algoritmov neprečiščenih lematiziranih podatkov z vključenim razredom nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	36.72	38.17	38.9	20.97	21.83	41.01	35.7	51.25	50.38	43.27	43.1
Stowords	/	38.21	38.9	20.7	22.08	40.07	35.45	51.74	50.50	43.38	43.1
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	38.49	38.96	15.58	14.93	41.37	41.28	52.06	50.56	45.96	46.30
Stopwords	/	38.32	38.39	15.49	15.06	41.52	40.28	51.87	51.44	47.34	46.74

Model brez razreda nič

Najboljši rezultat dosega algoritem SMO z izločenimi nepomembnimi besedami v CV10 načinu evalvacije z rezultatom 58,39 %. Sledi mu osnovna različica z 58,12 % v CV10 načinu evalvacije. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, Naive Bayes in OneR.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 749 atributov v osnovnem načinu in 743 atributov v načinu brez nepomembnih besed. To predstavlja 12,99 % (5766 proti 749) začetnih atributov v prvem primeru oz. 12,82 % (5795 proti 743) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 56,75 %, je dosežen pri algoritmu SMO v CV10 načinu evalvacije z izločenimi nepomembnimi besedami. Sledi mu osnovna SMO različica v CV10 načinu evalvacije – 56,35 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 44,37 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 1,64 % v korist prvemu.

Tabela 15: Primerjava algoritmov neprečiščenih lematiziranih podatkov brez razreda nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	14.02	15.72	13.81	27.53	29.13	39.91	40.24	58.12	57.96	44.30	44.14
Stowords	/	14.87	13.91	27.29	29.23	39.50	41.84	58.39	57.86	44.85	44.84
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	16.77	16.02	23.41	22.52	44.20	41.24	56.35	54.96	51.72	51.15
Stopwords	/	15.28	15.12	23.14	22.32	43.72	40.84	56.75	56.26	52.16	53.45

4.1.3 Prečiščeni nelematizirani podatki

Model z vključenim razredom nič

Najboljši rezultat dosega algoritem SMO v CV10 načinu evalvacije z rezultatom 51,77 %. Sledi mu različica z izločenimi nepomembnimi besedami v CV10 načinu evalvacije z 51,05 %. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, OneR in Naive Bayes.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 931 atributov v osnovnem načinu in 924 atributov v načinu brez nepomembnih besed. To predstavlja 11,97 % (7781 proti 931) začetnim atributov v prvem primeru oz. 11,80 % (7825 proti 924) v drugem primeru. Izgradnja modela je bistveno

hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 50,65 %, je dosežen pri algoritmu SMO v CV10 načinu evalvacije z izločenimi nepomembnimi besedami. Sledi mu osnovna SMO različica v CV10 načinu evalvacije – 50,59 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 15,41 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 1,12 % v korist prvemu.

Tabela 16: Primerjava algoritmov prečiščenih nelematiziranih podatkov z vključenim razredom nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	36.36	37.75	37.12	14.32	12.69	39.17	35.77	51.77	48.53	42.7	40.9
Stowords	/	37.55	36.8	14.58	13.01	39.6	35.13	51.046	48.59	42.44	40.45
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	37.47	36.60	11.79	10.83	41.72	39.74	50.59	49.42	46.334	43.97
Stopwords	/	37.51	37.31	11.94	10.58	41.61	39.61	50.6	48.97	46.91	45.71

Model brez razreda nič

Najboljši rezultat dosega algoritem SMO v CV10 načinu evalvacije z rezultatom 58,32 %. Sledi mu različica z izločenimi nepomembnimi besedami v CV10 načinu evalvacije z 58,08 %. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, Naive Bayes in OneR.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 919 atributov v osnovnem načinu in 917 atributov v načinu brez nepomembnih besed. To predstavlja 11,84 % (7760 proti 919) začetnih atributov v prvem primeru oz. 11,75 % (7804 proti 917) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 55,48 %, je dosežen pri algoritmu SMO v CV10 načinu. Sledi mu SMO-različica v CV10 načinu evalvacije z izločenimi nepomembnimi besedami – 50,59 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 44,21 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 2,84 % v korist prvemu.

Tabela 17: Primerjava algoritmov prečiščenih nelematiziranih podatkov brez razreda nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random Forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	14.11	14.52	13.19	19.73	19.44	41.44	36.76	58.32	53.68	43.77	37.87
Stowords	/	14.93	14.2	19.66	19.34	39.93	39.68	58.08	53.98	43.12	36.96
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	14.59	14.70	17.95	17.52	44.11	41.09	55.48	51.86	51.37	47.23
Stopwords	/	14.45	17.02	17.95	17.42	44.45	42.09	54.83	51.26	50.72	47.63

4.1.4 Prečiščeni lematizirani podatki

Model z vključenim razredom nič

Najboljši rezultat dosega algoritem SMO v CV10 načinu evalvacije z rezultatom 50,78 % z izločenimi nepomembnimi besedami. Sledi mu osnovna različica v CV10 načinu evalvacije z 50,74 %. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, OneR in Naive Bayes.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 755 atributov v osnovnem načinu in 748 atributov v načinu brez nepomembnih besed. To predstavlja 13,30 % (5675 proti 755) začetnih atributov v prvem primeru oz. 13,07 % (5720 proti 748) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 51,92 %, je dosežen pri algoritmu SMO v CV10 načinu z izločenimi nepomembnimi besedami. Sledi mu SMO osnovna različica v CV10 načinu evalvacije – 51,77 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO algoritma znaša 15, 59 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 1,14 % v korist drugemu.

Tabela 18: Primerjava algoritmov prečiščenih lematiziranih podatkov z vključenim razredom nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	36.33	38.01	36.67	21.58	22.18	38.51	34.74	50.74	48.46	43.13	40.77
Stopwords	/	37.75	36.60	21.64	22.37	38.64	34.81	50.78	48.4	43.29	40.83
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	37.79	37.76	15.3	15.38	41.72	38.91	51.77	49.94	46.88	43.85
Stopwords	/	37.55	37.31	15.37	15.26	41.63	38.97	51.92	50.13	47.38	45

Model brez razreda nič

Najboljši rezultat dosega algoritem SMO v CV10 načinu evalvacije z rezultatom 58,75 % z izločenimi nepomembnimi besedami. Sledi mu osnovna različica v CV10-načinu evalvacije z 58,61 %. Ostali algoritmi dosegajo bistveno slabše rezultate. Sledijo si v naslednjem vrstnem redu: Random Forest, J48, Naive Bayes in OneR.

Pri izločanju besed, ki nimajo pozitivne informacijske vrednosti, imamo za izgradnjo modela na voljo 726 atributov v osnovnem načinu in 724 atributov v načinu brez nepomembnih besed. To predstavlja 12,81 % (5669 proti 726) začetnih atributov v prvem primeru oz. 12,68 % (5712 proti 724) v drugem primeru. Izgradnja modela je bistveno hitrejša, vendar je posledica slabša točnost modela. Opazno se popravi točnost modelov, ki temeljijo na drevesni strukturi – J48 in Random Forest. Najboljši rezultat, ki znaša 57,48 %, je dosežen pri algoritmu SMO v CV10 načinu z izločenimi nepomembnimi besedami. Sledi mu osnovna SMO različica v CV10 načinu evalvacije – 57,45 %.

Razlika med izhodiščem (ZeroR) in najboljšim rezultatom SMO-algoritma znaša 44,61 %. Razlika med najboljšim modelom osnovne različice in tistim, ki ima vključene zgolj atribute s pozitivno informacijsko vrednostjo, znaša 1,27 % v korist prvemu.

Tabela 19: Primerjava algoritmov prečiščenih lematiziranih podatkov brez razreda nič v %

	ZeroR	**OneR**		**NaiveBayes**		**J48**		**SMO**		**Random forest**	
		CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66	CV10	Per66
Brez Stopwords	14.14	14.38	15.11	28.14	26.59	38.86	37.46	58.61	57.80	44.44	42.5
Stowords	/	14.24	19.23	28.41	27.49	40.02	39.07	58.75	57.80	44.06	41.79
InfoGainAttributeEval + izbrani algoritem											
Brez Stopwords	/	14.28	12.29	23.9	23.767	43.31	38.07	57.45	55.39	51.87	50.25
Stopwords	/	14.89	13.9	23.83	23.57	43.31	39.27	57.48	55.39	52.89	50.05

4.2 PRIMERJAVA REZULTATOV

V prejšnjem poglavju smo naredili splošen pregled rezultatov po posameznih različicah turističnih korpusov in scenarijih, vključenih v njihovo izdelavo. Iz pregleda modelov je jasno razvidno, da je izmed testiranih algoritmov SMO najboljši algoritem za klasifikacijo besedil. V tem poglavju smo se osredotočili na iskanje odgovorov o dejavnikih, ki vplivajo na točnost našega modela. Zaradi obsežnega števila algoritmov smo analizirali zgolj rezultate SMO-algoritma, ki v kombinaciji s CV10 evalvacijo daje najboljše rezultate. V tabeli 20 so za lažji pregled zbrane evalvacije SMO-algoritma v vseh podatkovnih različicah in scenarijih. V nadaljevanju sledi podrobna analiza dejavnikov, ki po naši oceni vpliva na točnost končnega modela.

Tabela 20: Rezultati SMO algoritma s CV10 evalvacijo v %

	Neprečiščeni nelematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	51,56	51,11	56,26	56,39
InfoGain	50,90	51,11	53,63	53,59
	Neprečiščeni lematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	51,25	51,74	58,12	58,39
InfoGain	52,06	51,87	56,35	56,75
	Prečiščeni nelematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	51,77	51,05	58,32	58,08
InfoGain	50,59	50,65	55,48	54,83
	Prečiščeni lematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	50,74	50,78	58,61	58,75
InfoGain	51,77	51,92	57,45	57,48

4.2.1 Analiza dejavnikov točnosti modela

V kolikšni meri čiščenje podatkov vpliva na točnost modela?

Iz tabele 21 lahko vidimo, da ima čiščenje podatkov pozitiven prispevek k točnosti modela v primeru, ko v modeliranje nimamo vključenega razreda 0. Večji prispevek je zaznati na

podatkih, ki predhodno niso bili lematizirani. Na nelematiziranih in lematiziranih podatkih, kjer je vključen razred 0, je zaznati poslabšanje točnosti v primeru čiščenja podatkov.

Tabela 21: Vpliv čiščenja podatkov v %

	Neprečiščeni proti prečiščeni nelematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	+0,21	-0,06	+2,06	+1,69
InfoGain	-0,31	-0,46	+1,85	+1,24
	Neprečiščeni proti prečiščeni lematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	-0,51	-0,96	+0,49	+0,36
InfoGain	-0,29	+0,05	+1,1	+0,73

V kolikšni meri izločanje nerazvrščenih besedil (razred 0) vpliva na točnost modela?

Iz tabele 22 lahko razberemo, da ima izločanja nerazvrščenih podatkov precejšen prispevek k točnosti modela. Pri treh od štirih scenarijev, ko imamo v modeliranje vključene vse besede (brez InfoGain), je prispevek največji v kombinaciji z izločenimi nepomembnimi besedami, kjer v najboljšem primeru prečiščenih lematiziranih podatkov dosega 7,98 % izboljšanja točnosti modela.

Pri scenariju z izločenimi besedami brez informacijske vrednosti je prispevek manjši v primerjavi z osnovno različico. Tu je situacija ravno obratna, saj izločanje nerazvrščenih besedil v treh od štirih primerov daje boljše rezultate v primeru, ko nimamo izločenih nepomembnih besed. Največji prispevek v primeru prečiščenih lematiziranih podatkov znaša 5,86 %.

Tabela 22: Vpliv izločanja nerazvrščenih podatkov v %

	Neprečiščeni nelematizirani podatki	
	Brez Stopwords	Stopwords
Brez InfoGain	+4,7	+5,28
InfoGain	+2,73	+2,48
	Neprečiščeni lematizirani podatki	
	Brez Stopwords	Stopwords
Brez InfoGain	+6,87	+6,65
InfoGain	+4,29	+4,88
	Prečiščeni nelematizirani podatki	
	Brez Stopwords	Stopwords
Brez InfoGain	+6,55	+7,03
InfoGain	+4,89	+4,18
	Prečiščeni lematizirani podatki	
	Brez Stopwords	Stopwords
Brez InfoGain	+7,87	+7,97
InfoGain	+5,86	+5,56

V kolikšni meri lematizacija vpliva na točnost modela?

Lematizacija prispeva k povečanju točnosti modela v 13 od 16 primerov. Največji prispevek je mogoče razbrati pri kombinaciji neprečiščenih podatkov z uporabo nepomembnih besed tako na osnovni različici kot na različici brez informacijsko pomembnih besed. Prispevek je 2 % v prvem primeru in 3,16 % v drugem primeru. Negativen prispevek ima v treh primerih osnovne različice (brez InfoGain) z vključenim razredom 0.

Tabela 23: Vpliv lematizacije podatkov v %

	Neprečiščeni nelematizirani proti lematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	-0,31	+0,63	+1,86	+2
InfoGain	+1,16	+0,76	+2,72	+3,16
	Prečiščeni nelematizirani proti lematizirani podatki			
	Z vključenim razredom 0		Brez razreda 0	
	Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
Brez InfoGain	-1,03	-0,27	+0,29	+0,67
InfoGain	+1,18	+1,27	+1,97	+2,65

V kolikšni meri izločanje nepomembnih besed vpliva na točnost modela?

Iz tabele 24 je mogoče razbrati, da ima izločanje nepomembnih besed zanemarljiv vpliv na točnost modela. Največji prispevek ima v primeru lematiziranih podatkov. Pri nelematiziranih podatkih v veliki večini povzroča slabšanje točnosti modela. K najboljšemu modelu lematizacija prispeva zgolj 0,14 % točnosti.

Tabela 24: Vpliv izločanja nepomembnih besed v %

	Neprečiščeni nelematizirani podatki	
	Z vključenim razredom 0	Brez razreda 0
Brez InfoGain	-0,45	+0,13
InfoGain	+0,21	-0,07
	Neprečiščeni lematizirani podatki	
	Z vključenim razredom 0	Brez razreda 0
Brez InfoGain	+0,49	+0,27
InfoGain	-0,19	+0,4
	Prečiščeni nelematizirani podatki	
	Z vključenim razredom 0	Brez razreda 0
Brez InfoGain	-0,72	-0,24
InfoGain	+0,06	-0,65
	Prečiščeni lematizirani podatki	
	Z vključenim razredom 0	Brez razreda 0
Brez InfoGain	+0,04	+0,14
InfoGain	+0,15	+0,03

V kolikšni meri izločanje informacijsko nepomembnih besed vpliva na točnost modela?

Iz tabele 25 je razvidno, da vpliv izločanja informacijsko nepomembnih besed nima pozitivnega prispevka k točnosti modelov. Izjema so neprečiščeni in prečiščeni lematizirani podatki z vključenim razredom 0. V vseh ostalih primerih se negativni prispevek giblje med -2,84 % in 0 %. Naš najboljši model poslabša za 1,27 %. Zaključimo lahko, da izločanje informacijsko nepomembnih besed sicer bistveno vpliva na krajši čas izgradnje modela zaradi zmanjšanja števila atributov, vendar se točnost modela poslabša.

Tabela 25: Vpliv izločanja informacijsko nepomembnih besed v %

Neprečiščeni nelematizirani podatki			
Z vključenim razredom 0		Brez razreda 0	
Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
-0,66	0	-2,63	-2,8
Neprečiščeni lematizirani podatki			
Z vključenim razredom 0		Brez razreda 0	
Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
+0,81	+0,13	-1,77	-1,64
Prečiščeni nelematizirani podatki			
Z vključenim razredom 0		Brez razreda 0	
Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
-1,18	-0,4	-2,84	-3,25
Prečiščeni lematizirani podatki			
Z vključenim razredom 0		Brez razreda 0	
Brez Stopwords	Stopwords	Brez Stopwords	Stopwords
+1,03	+1,14	-1,16	-1,27

4.3 PODROBNA ANALIZA NAJBOLJŠEGA MODELA SMO

Najboljši rezultat smo dosegli z modelom SMO v različici s prečiščenimi lematiziranimi podatki brez vključenega razreda 0 in z izločenimi nepomembnimi besedami. Od 2921 besedil jih je model pravilno klasificiral 1716, kar znaša 58,75 %. Kappa statistika znaša 0,55. Povprečna absolutna napaka pa 0,07.

V podrobnem pregledu statistike po razredih na sliki 20 lahko vidimo, da se delež resnično pozitivnih le primeru razreda 13, 20 in 22 nahaja nad vrednostjo 0,800, ki jo smatramo za zelo dobro. Evalvacijo uspešnosti posameznega razreda je moč razbrati iz stolpca ROC Area. V utežnem povprečju znaša ROC area 0,900, kar predstavlja odličen rezultat. Razredi, ki ne dosegajo vrednost nad 0.800, so 1, 5, 8 in 15 [44]. Razlog za slab rezultat gre iskati predvsem v majhnem številu učnih primerov za naštete razrede.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1716               58.747  %
Incorrectly Classified Instances      1205               41.253  %
Kappa statistic                          0.5511
Mean absolute error                      0.0714
Root mean squared error                  0.187
Relative absolute error                 99.8717 %
Root relative squared error             98.9148 %
Total Number of Instances             2921

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.001     0.000     0.000     0.000     -0.003    0.756     0.025     1
                 0.697    0.143     0.446     0.697     0.544      0.466    0.844     0.402     2
                 0.690    0.014     0.661     0.690     0.675      0.662    0.925     0.573     3
                 0.553    0.032     0.409     0.553     0.470      0.451    0.908     0.311     4
                 0.000    0.001     0.000     0.000     0.000     -0.002    0.733     0.036     5
                 0.443    0.017     0.515     0.443     0.477      0.458    0.925     0.381     6
                 0.305    0.006     0.630     0.305     0.411      0.426    0.816     0.310     7
                 0.000    0.000     0.000     0.000     0.000     -0.001    0.431     0.002     8
                 0.407    0.013     0.486     0.407     0.443      0.429    0.878     0.286     9
                 0.356    0.013     0.591     0.356     0.444      0.438    0.892     0.339     10
                 0.429    0.021     0.516     0.429     0.468      0.445    0.843     0.315     11
                 0.143    0.013     0.217     0.143     0.172      0.160    0.816     0.120     12
                 0.810    0.019     0.786     0.810     0.797      0.780    0.974     0.744     13
                 0.400    0.000     1.000     0.400     0.571      0.631    0.857     0.641     14
                 0.087    0.002     0.286     0.087     0.133      0.154    0.662     0.087     15
                 0.357    0.002     0.500     0.357     0.417      0.420    0.834     0.268     16
                 0.476    0.000     0.909     0.476     0.625      0.656    0.906     0.455     17
                 0.390    0.007     0.612     0.390     0.476      0.478    0.899     0.340     18
                 0.531    0.024     0.469     0.531     0.498      0.477    0.936     0.365     19
                 0.896    0.010     0.870     0.896     0.883      0.874    0.993     0.859     20
                 0.480    0.022     0.488     0.480     0.484      0.461    0.883     0.340     21
                 0.827    0.005     0.860     0.827     0.843      0.838    0.986     0.821     22
                 0.776    0.051     0.704     0.776     0.738      0.697    0.929     0.638     23
                 0.412    0.017     0.529     0.412     0.464      0.445    0.889     0.351     24
                 0.273    0.000     0.750     0.273     0.400      0.451    0.865     0.244     25
                 0.500    0.017     0.557     0.500     0.527      0.509    0.907     0.404     26
Weighted Avg.    0.587    0.038     0.588     0.587     0.576      0.551    0.900     0.473
```

Slika 20: Povzetek evalvacije najboljšega modela SMO

Na sliki 21 je predstavljana matrika resnično pozitivnih in lažno pozitivnih besedil, s katere lahko identificiramo pravilno in nepravilno razvrščena besedila iz učne množice. Matrika nam nudi dober prikaz distribucije nepravilno razvrščenih besedil, s čimer lahko analiziramo, zakaj je bil nek delež besedil klasificiran v napačen razred. Za primer si poglejmo razred 26, kjer je razvidno, da je SMO-algoritem velik del napačno razvrščenih klasificiral v razred 2 (b) in 21(u). Smiselno bi bilo ta besedila preveriti z namenom odkrivanja vzroka, ki vpliva na napačno klasifikacijo.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	<-- classified as
0	9	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	1	1	0	1	a = 1
1	288	1	13	0	21	2	0	3	3	13	3	2	0	0	0	0	3	8	0	9	5	12	10	0	16	b = 2
0	12	78	1	0	1	0	0	1	0	1	3	7	0	0	0	0	1	1	0	0	0	3	3	0	1	c = 3
0	22	0	63	0	0	0	0	9	0	2	1	0	0	0	0	0	0	2	3	2	0	7	3	0	0	d = 4
0	5	7	0	0	0	0	0	0	0	2	0	3	0	0	0	0	0	1	2	1	0	2	0	0	0	e = 5
0	28	5	2	0	51	2	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	10	2	0	10	f = 6
0	33	2	8	0	1	29	0	0	2	2	2	1	0	0	0	0	0	2	0	2	4	1	5	0	1	g = 7
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	1	h = 8
0	8	0	11	0	0	0	0	35	2	0	3	5	0	0	0	0	2	6	5	1	0	4	2	0	2	i = 9
0	17	0	4	0	4	1	0	1	52	8	0	6	0	0	0	0	1	0	0	9	0	39	2	0	2	j = 10
1	35	2	4	0	1	2	0	1	8	63	3	5	0	0	0	0	2	2	0	6	2	8	0	0	2	k = 11
0	15	1	8	0	0	5	0	5	0	1	10	1	0	0	0	0	0	13	1	0	0	3	4	0	3	l = 12
0	8	7	1	1	1	0	0	2	0	2	0	187	0	1	5	0	0	5	0	2	0	6	1	1	1	m = 13
0	1	0	1	0	0	0	0	0	0	3	0	0	6	0	0	0	0	1	1	0	0	0	2	0	0	n = 14
0	4	0	1	0	0	0	0	0	0	3	1	0	0	2	0	0	2	1	2	2	0	1	4	0	0	o = 15
0	2	0	0	0	0	0	0	1	0	0	0	4	0	0	5	0	0	0	0	0	0	2	0	0	0	p = 16
0	1	0	2	0	0	0	0	0	1	0	1	0	0	0	0	10	1	0	1	2	0	2	0	0	0	q = 17
0	14	4	3	0	4	0	0	0	2	2	0	2	0	1	0	0	30	0	0	2	0	12	1	0	0	r = 18
0	10	0	7	0	2	0	0	2	0	0	13	3	0	0	0	1	0	60	9	1	0	2	3	0	0	s = 19
0	2	0	2	0	0	0	0	6	0	0	0	0	0	0	0	0	0	10	181	0	0	0	1	0	0	t = 20
0	34	0	2	0	1	0	0	0	4	7	0	3	0	1	0	0	1	1	0	59	1	4	0	0	5	u = 21
0	9	1	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	1	0	1	86	2	0	0	0	v = 22
1	26	5	9	0	4	1	0	2	13	7	0	3	0	0	0	0	3	6	1	4	1	305	2	0	0	w = 23
0	37	4	9	0	2	0	1	4	1	3	3	1	0	0	0	0	0	5	0	1	1	4	54	0	1	x = 24
0	1	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1	0	0	0	1	0	3	1	y = 25
1	24	0	1	0	6	2	0	0	0	1	2	1	0	1	0	0	2	1	0	14	0	2	1	0	59	z = 26

Slika 21: Matrika pravilnih in nepravilnih razvrstitev najboljšega SMO modela

4.4 SKLEP

Primerjava algoritmov, ki smo jih testirali na študiji primera turističnega korpusa Slovenije, je pokazala, da je SMO najboljši klasifikacijski algoritem izmed testiranih. V prečiščeni in lematizirani različici turističnega korpusa je skupaj z odstranjenimi nepomembnimi besedami dosegel 58,75 % uspešnosti klasifikacije besedil iz učne množice. Kappa statistika je ob tem znašala 0,55, povprečna absolutna napaka pa 0,07. Evalvacija uspešnosti razredov z izjemo razredov 1, 5, 8 in 15 dosega rezultate med 0,800-1, kar predstavlja odličen rezultat. Uteženo povprečje znaša 0,900, s čimer lahko potrdimo, da je model uspešen.

Poleg iskanja najprimernejšega klasifikacijskega algoritma smo analizirali 5 dejavnikov, ki naj bi po našem mnenju vplivali na točnost končnega modela. Analizo smo zaradi obsega interpretacije izvedli samo na SMO-algoritmu. Odkrili smo, da ima čiščenje podatkov pozitiven prispevek k točnosti modela v primeru, ko v modeliranje nimamo vključenega razreda 0. Večji prispevek smo zaznali na podatkih, ki predhodno niso bili lematizirani. Na nelematiziranih in lematiziranih podatkih, kjer je vključen razred 0, je bilo zaznati celo poslabšanje točnosti v primeru čiščenja podatkov. Na nelematiziranih podatkih je bilo zaznati okrog 2 % prispevka, na lematiziranih pa okrog 1 %.

Izločanje nerazvrščenih besedil (razreda 0) ima precejšen prispevek k točnosti modela. Najboljši rezultat dosega pri osnovni različici z izločenimi nepomembnimi besedami, kjer dosega v povprečju 6,79-odstotno izboljšanje. Pri scenariju z izločenimi besedami brez informacijske vrednosti (InfoGain) je prispevek manjši v primerjavi z osnovno različico. Tu

algoritem daje boljše rezultate, ko nimamo izločenih nepomembnih besed. V povprečju dosega 4,59-odstotno izboljšanje.

Za lematizacijo velja, da povečuje točnost modela v kar 15 od 18 primerov. Največji prispevek je bil prisoten pri kombinaciji neprečiščenih podatkov z uporabo nepomembnih besed tako v osnovni različici kot v različici brez informacijsko pomembnih besed. K točnosti najboljšega modela lematizacija prispeva 0,67 %.

Izločanje nepomembnih besed ima zanemarljiv vpliv na točnost modela. Največji prispevek ima v primeru lematiziranih podatkov. K najboljšemu modelu prispeva zgolj 0,14 % točnosti. Pri nelematiziranih podatkih v veliki večini povzroča slabšanje točnosti modela.

Izločanja informacijsko nepomembnih besed nima pozitivnega prispevka k točnosti modelov. Izjema so neprečiščeni in prečiščeni lematizirani podatki z vključenim razredom 0 v vseh ostalih primerih se negativni prispevek giblje med 2,84 % in 0 %. Naš najboljši model poslabša za 1,27 %. Zaključimo lahko, da izločanje informacijsko nepomembnih besed sicer bistveno vpliva na krajši čas izgradnje modela zaradi zmanjšanja števila atributov, vendar se točnost modela poslabša.

Dejavnik, ki po naši analizi prispeva 7,97 % k točnosti najboljšega modela, je izločanje nekvalificiranih besedil. Čiščenje, lematizacija in izločanje nepomembnih besed ima zanemarljiv, a kljub vsemu pomemben prispevek v skupnem deležu 1,17 %.

5 ZAKLJUČEK IN NADALJNJE DELO

V magistrskem delu smo naprej predstavili motivacijo, ki bila ključnega pomena pri izbiri in oblikovanju teme magistrskega dela. Podrobneje smo opisali problematiko, ki jo rešujemo in prešli na pregled področja, ki se je s to problematiko že srečeval. V raziskovalnem ozadju smo predstavili pojme procesa odkrivanja znanja iz podatkov, podatkovno rudarjenje in uporabljene tehnike, metodologijo CRISP-DM, obdelavo naravnih jezikov in tekstovno rudarjenje ter izbrano orodje, ki smo ga uporabili v drugem delu magistrskega dela. Tu smo podrobneje opisali postopek izgradnje napovednega modela, ki je temeljil na primeru turističnega korpusa. Izgradnja modela je potekala v skladu s CRISP-DM-metodologijo, kjer smo naprej predstavili podatke, izvedli preobdelavo in jih z ustreznimi tehnikami pripravili na uvoz v WEKO. Nanje smo nato aplicirali izbrane algoritme, ki smo jih predhodno opisali. Zgrajene modele je bilo treba z ustreznimi testi evalvirati in podati končne zaključke o vplivu dejavnikov lematizacije, čiščenja besedil, izločanja nerazvrščenih besedi, nepomembnih besed in besed brez informacijske vrednosti. Rezultat najboljšega klasifikacijskega algoritma smo podrobneje predstavili v poglavju Primerjava rezultatov, za konec pa podrobneje analizirali najboljšo verzijo SMO modela z namenom odkrivanja prihodnjih izboljšav. V sklepu smo izpostavili rezultate analize modelov.

V nadaljnjem delu bi se bilo smiselno osredotočiti na ključno pomanjkljivost modelov, ki jo je izpostavila ROC-analiza in se nanaša na pomanjkanje učnih primerov v določenih razredih, zaradi česar prihaja do podvzorčenja podatkov in posledično slabše točnosti modela. Rešitev bi bilo treba iskati v tehnikah, ki rešujejo težave z neuravnoteženimi podatki. Dve izmed njih sta stroškovno občutljiv klasifikator (*ang. Cost-Sensitive Classification*) in ponovno vzorčenje podatkov z namenom uravnoteženja distribucije posameznega razreda. V Weki je za to namenjen nadzorovani filter za ponovno vzorčenje (*angl. Resample*). Pri izločanju nepomembnih besed bi bilo treba obstoječi seznam besed nagraditi v smeri ročnega izbora nepomembnih besed za vsak razred posebej. Trenutni seznam temelji na frekvenci pojavnosti besed v vseh razredih in po dogovoru vključuje zgolj prvih 20 najpogostejših besed. Poleg izločanja nepomembnih besed bi bilo v prihodnje smiselno testirati še nekatere druge klasifikacijske algoritme, ki jih sedaj nismo vključili v analizo in s katerimi bi mogoče dosegali boljše rezultate od trenutnih.

Zgrajen model bi bilo možno uporabiti v praksi kot klasifikator za nacionalno spletno mesto, kjer bi bila dostopna vsa besedila povezana s turizmom. Ta besedila bi lahko z razvitim modelom avtomatsko klasificirali ob predpostavki, da bi točnost modela v prihodnje še izboljšali. To bi pripomoglo k temu, da bi izločili subjektivni človeški faktor pri razvrščanju besedil v posamezne kategorije, pohitrili proces ročne klasifikacije in uporabnikom turističnih storitev omogočili lažje iskanje po neurejenih bazah turističnih besedil. Čeprav natančnost modela na prvi pogled ni zavidanja vredna, pa je naš model kar za 44,61 % boljši od izhodiščnega modela (ZeroR). Rezultati sicer niso superiorni, vendar

zadovoljivi, da lahko razmišljamo o implementaciji sistema za strojno razvrščanje besedil tudi v praksi.

6 LITERATURA IN VIRI

[1] C. C. Aggarwal in C. Zhai, "Introduction," v *Mining Text Data*. New York: Springer, 2012, str. 1-3.

[2] R. Balamurugan in S. Pushpa, "A review on various text mining techniques and algorithms," v *2nd International Conference on Recent Innovations in Science, Engineering and Management*. Chennai, 2015, str. 837-848.

[3] A. Bronshtein. Train/Test Split and Cross Validation in Python. (2017). *Medium* [online]. Dosegljivo: https://medium.com/towards-data-science/train-test-split-and-cross-validation-in-python-80b61beca4b6. [Dostopano: 26.7.2017].

[4] S. Colja, "Tehnike podatkovnega rudarjenja," v Algoritmi in tehnike podatkovnega rudarjenja na bazi procesnih parametrov. Maribor: Fakulteta za matematiko in fiziko, 2011, str. 11-12.

[5] U. Fayyad, G. Piatetsky-Shapiro in P. Smyth, Knowledge Discovery and Data Mining: Towars a Unifying Framework. 1996, str. 82-86.

[6] V. C. Gandhi in J. A. Prajapati, "Review on Comparison between Text Classification Algorithms," v *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*. Oktober 2012, str. 75-78.

[7] M. Hall, Class InfoFainAttributeEval. *The University of North Carolina at Chapel Hill, School of Information and Library Science* [online]. Dosegljivo: https://ils.unc.edu/bmh/neoref/nrplantgene/tmp/weka-34/doc/weka/attributeSelection/InfoGainAttributeEval.html. [Dostopano: 20.7.2017].

[8] J. Han, M. Kamber in J. Pei, "Summary," v *Data Mining Concept and Techniques*, vol.2. San Francisco: Morgan Kaufmann Publishers, 2006, str. 33.

[9] J. Han, M. Kamber in J. Pei, " What Motivated Data Mining? Why Is It Important?," v *Data Mining Concept and Techniques*, vol.2. San Francisco: Morgan Kaufmann Publishers, 2006, str. 3-4.

[10] G. Harper in S. D. Pickett, "Methods for mining HTS data," v *Drug Discovery Today*, 11 (15–16). Avgust 2016, str. 694–699.

[11] L. N. Jovan, "Metode," v *Priprava podatkov za občinskega virtualnega asistenta s pomočjo strojnega učenja*. Ljubljana: Fakulteta za računalništvo in infromatiko, 2016, str. 13-19.

[12] L. N. Jovan, "Klasifikacija," v *Prepoznava žaljivih objav z metodami strojnega učenja*. Ljubljana: Fakulteta za računalništvo in informatiko, 2013, str.14-17.

[13] S. Kampakis. Performance measures: Cohen's Kappa statistic. (Maj 2016). *The data scientist* [online]. Dosegljivo: http://www.skampakis.com/performance-measures-cohens-kappa-statistic/. [Dostopano: 26.7.2017].

[14] S. Kunwar. Text Documents Clustering using K-Means Algorithm. (2013) *CodeProject* [online]. Dosegljivo:

https://www.codeproject.com/Articles/439890/Text-Documents-Clustering-using-K-Means-Algorithm. [Dostopano: 25.6.2017].

[15] N. R. Mabroukeh in C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," v *ACM Computing Surveys*. 2010, str.1–41.

[16] J. McGonagle. Naive Bayes Classified. *Brilliant* [online]. Dosegljivo: https://brilliant.org/wiki/naive-bayes-classifier/. [Dostopano: 22.7.2017].

[17] R. Mikut in M. Reischl. (2011). *Data mining tools*. str. 431-443.

[18] R. A. Muenchen. (2017). *The Popularity of Data Science Software*. Dosegljivo: http://r4stats.com/articles/popularity/. [Dostopano: 13.7.2017].

[19] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines", Microsoft Research: April 1998.

[20] C. Priyadharsini in A. S. Thanamani, "An Overview of Knowledge Discovery Database and Data Mining Techniques," v *Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14), 6.-7. marec 2014, Tirupur, Indija.* Tirupur: 2014, str. 1572-1575.

[21] B. M. Ramageri, "Data mining techiques and applications," v *Indian Journal of Computer Science and Engineering*. Vol. 1 No. 4. Maharashtra: Modern Institute of Information Technology and Research, str. 301-305.

[22] P. Ross. OneR: The simplest method. *Edinburgh Napier University* [online]. Dosegljivo: http://www.soc.napier.ac.uk/~peter/vldb/dm/node8.html. [Dostopano: 21.7.2017].

[23] K. B. Sathees in R. Karthika, "A survey on text mining process and tecniques," v *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3*. Trichirappalli, Julij 2014, str. 2279-2284.

[24] S. Sayad. OneR. *University of Toronto, Faculty of Applied Science & Engineering* [online]. Dosegljivo: http://chem-eng.utoronto.ca/~datamining/dmc/oner.htm. [Dostopano: 21.7.2017].

[25] S. Sayad. ZeroR. *University of Toronto, Faculty of Applied Science & Engineering* [online]. Dosegljivo: http://chem-eng.utoronto.ca/~datamining/dmc/zeror.htm. [Dostopano: 20.7.2017].

[26] C. Shearer, "The CRISP-DM model: the new blueprint for data mining," v J Data Warehousing," *v Journal of Data Warehousing*, št.5, str.13-22, 2000.

[27] B. Škoda, "Obdelava naravnega jezika," v Rudarjenje razpoloženja na komentarjih rtvslo.si. Ljubljana: Fakulteta za računalništvo in informatiko, 2013, str. 17-20.

[28] N. Tax et al.. (2016). "Mining Local Process Models," v *Journal of Innovation in Digital Ecosystems*. Elsevier, vol.3, št.2, str.183–196.

[29] M. Trivede et. al., "Comparison of Text Classification Algorithms," v *International Journal of Engineering Research & Technology (IJERT)*. Februar 2015, str. 334-336.

[30] J. Vičič et al., Namen in metode urejanja večjezičnega korpusa turističnih besedil (TURK). Koper: Univerza na Primorskem, 2009, str. 65-74.

[31] M. Vidoni, "Predobdelava podatkov," v *Napovedovanje oseba komentiranja spletnih novic z modeli strojnega učenja*. Ljubljana: Fakulteta za računalništvo in informatiko, 2016, str. 5-11.

[32] I. H. Witten , "Text mining," University of Waikato: Hamilton, New Zealand, 2003.

[33] I. H. Witten, E. Frank in M. A. Hall, *Data mining: Practical Machine Learning Tools and Techniques*, vol.3. Burlington: Morgan Kaufmann Publishers, 2011.

[34] AI - Natural Language Processing. *Turorial Point* [online]. Dosegljivo: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm. [Dostopano: 9.8.2017].

[35] Besedilni korpus. (2016). *Wikipedija* [online]. Dosegljivo: https://sl.wikipedia.org/wiki/Besedilni_korpus. [Dostopano: 15.7.2017].

[36] Data mining: What it is and why it matters. *SAS* [online]. Dosegljivo: https://www.sas.com/en_us/insights/analytics/data-mining.html. [Dostopano: 22.7.2017].

[37] Data Requirement for Process Mining. *Fluxicon* [online]. Dosegljivo: https://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/. [Dostopano: 25.6.2017].

[38] Decision Trees - C4.5. *Octavian's blog* [online]. Dosegljivo: https://octaviansima.wordpress.com/2011/03/25/decision-trees-c4-5/. [Dostopano: 23.7.2017].

[39] Informacijska doba. (16.6.2017). *Wikipedija* [online]. Dosegljivo: https://sl.wikipedia.org/wiki/Informacijska_doba. [Dostopano: 15.6.2017].

[40] *Lematizacija*, Wikipedija [online]. Dosegljivo: https://sl.wikipedia.org/wiki/Lematizacija. [Dostopano: 7.7.2017].

[41] Natural language processing. (2017). *Wikipedija* [online]. Dosegljivo: https://en.wikipedia.org/wiki/Natural_language_processing#Major_evaluations_and_tasks. [Dostopano: 9.8.2017].

[42] Support vector machine. *Amitranga* [online]. Dosegljivo: https://amitranga.files.wordpress.com/2014/03/image44.png. [Dostopano: 22.7.2017].

[43] Text Mining. (Avgust, 2017). *Wikipedija* [online].

Dosegljivo: https://en.wikipedia.org/wiki/Text_mining. [Dostopano: 5.7.2017].

[44] The Area Under An ROC Curve [online]. *University of Nebraska - Medical Center*. Dosegljivo: http://gim.unmc.edu/dxtests/roc3.htm. [Dostopano: 3.8.2017].

[45] The KDD process for extracting useful knowledge from volumes of data. *Centro de investigacion en inteligenca de negocios* [online]. Dosegljivo: http://www.ceine.cl/the-kdd-process-for-extracting-useful-knowledge-from-volumes-of-data/. [Dostopano: 22.7.2017].

[46] *TI SI - Turizem Identiteta Slovenije* [online]. Dosegljivo: http://tisi.upr.si/. [Dostopano: 20.8.2017].

[47] Večjezični korpus turistični besedi. *TURK - Turistični korpus* [online]. Dosegljivo: http://turk.upr.si/turk2opis.html. [Dostopano: 18.7.2017].

[48] Vlada republike Slovenije. Strategija razvoja slovenskega turzima 2012-2016. (Junij 2012). *Ministrstvo za gospodarski razvoj in kulturo* [online]. Dosegljivo: http://www.mgrt.gov.si/fileadmin/mgrt.gov.si/pageuploads/turizem/Turizem-strategije_politike/Strategija_turizem_sprejeto_7.6.2012.pdf. [Dostopano: 17. 6. 2017].

[49] Weka - Evaluation: Assesing the performance. (2016). *Department of Linguistics and Philology at Uppsala University* [online]. Dosegljivo: http://stp.lingfil.uu.se/~santinim/ml/2016/Lect_04/Lab03_Evaluation.pdf. [Dostopano: 25.7.2017].

[50] Weka (machine learning). (2017). *Wikipedija* [online]. Dosegljivo: https://en.wikipedia.org/wiki/Weka_(machine_learning). [Dostopano: 14.7.2017].

[51] Weka (Strojno učenje). (2013). *Wikipedija*. Dosegljivo: https://sl.wikipedia.org/wiki/Weka_(strojno_u%C4%8Denje). [Dostopano: 14.7.2017].