UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Aleksej Miloševič

ANALIZA IN PRIMERJAVA PLATFORM ZA PODATKOVNO RUDARJENJE RAPIDMINER IN WEKA

Diplomsko delo

Maribor, september 2016

ANALIZA IN PRIMERJAVA PLATFORM ZA PODATKOVNO RUDARJENJE RAPIDMINER IN WEKA

Diplomsko delo

Študent:	Aleksej Miloševič
Študijski program:	Univerzitetni študijski program
	Informatika in tehnologije komuniciranja
Smer:	Upravljanje informatike
Mentor:	red. prof. dr. Vili Podgorelec
Somentor:	asist. Sašo Karakatič
Lektorica:	dr. Aleksandra Gačić, univ. dipl. prof. zgo. in slov.

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

Številka: E1068865
Datum in kraj: 20. 05. 2016, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 44/2015)
izdajam

SKLEP O DIPLOMSKEM DELU

1. **Alekseju Miloševiču,** študentu univerzitetnega študijskega programa Informatika in tehnologije komuniciranja, smer Upravljanje informatike, se dovoljuje izdelati diplomsko delo.

2. **MENTOR:** **red. prof. dr. Vili Podgorelec**
   SOMENTOR: **asist. Sašo Karakatič**

3. **Naslov diplomskega dela:**
   ANALIZA IN PRIMERJAVA PLATFORM ZA PODATKOVNO RUDARJENJE RAPIDMINER IN WEKA

4. **Naslov diplomskega dela v angleškem jeziku:**
   ANALYSIS AND COMPARISON OF DATA MINING PLATFORMS RAPIDMINER AND WEKA

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih (dva trdo vezana izvoda in en v spiralo vezan izvod) ter en izvod elektronske verzije do 30. 09. 2016 v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

UNIVERZA V MARIBORU

Dekan:
red. prof. dr. Borut Žalik

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv.

ZAHVALA

Zahvaljujem se mentorju dr. Viliju Podgorelcu in somentorju asist. Sašu Karaktiču za strokovno pomoč in vodenje pri izdelavi diplomskega dela.

Zahvaljujem se družini, ki mi je omogočila študij in nudila vzpodbudo ter podporo na študijski poti.

Posebna zahvala gre partnerki za navdih in vzpodbudo pri preseganju samega sebe.

Analiza in primerjava platform za podatkovno rudarjenje RapidMiner in Weka

Ključne besede: strojno učenje, klasifikacija, primerjava , RapidMiner, Weka

UDK: 004.65(043.2)

Povzetek

V pričujočem diplomskem delu sta analizirani in primerjani splošnonamenski platformi za podatkovno rudarjenje RapidMiner in Weka. V uvodnem delu diplomskega dela so razložene osnove strojnega učenja in podatkovnega rudarjenja ter podrobneje definirane metode dela, ki so uporabljene v praktičnem delu. Primerjava je razdeljena na teoretični in eksperimentalni del. V teoretičnem delu so na podlagi definirane metodologije identificirane pomembne lastnosti orodij in primerjane med seboj, v eksperimentalnem delu pa sta primerjani točnost in F-Mera implementacij algoritmov k-najbližjih sosedov, Naključni gozdovi in Naivni Bayes. S pomočjo statističnih testov je bilo ugotovljeno, da se nobena izvedenka algoritma od drugega statistično pomembno ne razlikuje.

Analysis and comparison of data mining platforms RapidMiner and Weka

Key words: machine learning, classification, comparison, RapidMiner, Weka

UDK: 004.65(043.2)

Abstract

The following thesis analyses and compares two general-purpose platforms for data mining, RapidMiner and Weka. The introductory part of this diploma thesis describes the basics of machine learning and data mining as well as the specifically defined work methods, which are used in the experimental part. The comparison is divided into the theoretical and the empirical part. In the theoretical part the important characteristics of the tools are identified and compared on the basis of the defined methodology, whereas in the empirical part the accuracy and the F-measure of implementations of the algorithms K Nearest Neighbor, Random Forest and Naive Bayes are compared. Using appropriate statistical tests, it was found that no version of the algorithm significantly differs from another.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

UPORABLJENE KRATICE

WEKA – Waikato Environment for Knowledge Analysis

ETL – Extract, Transform, Load

CSV – Comma-separated Values

SPSS – Statistical Package for the Social Sciences

ARFF – Attribute-Relation File Format

XRF – Extensible Attribute-Relation File Format

XML – Extensible Markup Language

JSON – JavaScript Object Notation

CLI – Command Line Interface

API – Application Programming Interface

CRISP-DM – Cross Industry Standard Process for Data Mining

SEMMA – Sample, Explore, Modify, Model and Assess

DMAIC – Define, Measure, Analyze, Improve and Control

JRE – Java Runtime Environment

SQL – Structured Query Language

AGPL – Affero General Public Licence

1 UVOD

V sodobni informacijski družbi je stalna praksa zbiranje velikih količin podatkov, ki se shranjujejo v podatkovne baze in kasneje obdelujejo z algoritmi strojnega učenja za namene pridobitve novih spoznanj. Metode in tehnike, ki se ukvarjajo z optimizacijo tega procesa, imenujemo podatkovno rudarjenje. Aplikacija tehnik podatkovnega rudarjenja je univerzalna; redno se uporabljajo v znanstvene, medicinske, vojaške in poslovne namene.

Za čim boljše rezultate je zato pomembno, da imajo podatkovni znanstveniki in analitiki na voljo orodja, ki jim omogočajo poglobljeno razumevanje obravnavanega problema in veliko možnosti pristopa k le-temu. V zadnjih letih se je področje še posebej razširilo saj na trg vsako leto prihaja veliko splošnih in specialističnih orodij, razvijalci obstoječih orodij pa so zaradi konkurenčnosti primorani implementirati najnovejše funkcionalnosti, če želijo ohraniti konkurenčnost.

Pri diplomskem delu smo se omejili na primerjavo odprtokodnih platform RapidMiner in Weka. Hoteli sem ugotoviti, kakšna je razlika med RapidMinerjem, trenutno ene izmed vodilnih platform uporabljenih v industriji [1], in Weko, ki je ena izmed najstarejših platform z zelo aktivno skupnostjo. Za namene primerjave smo osrednji del diplomske naloge razdelili na tri dele. V prvem delu smo opisali splošne koncepte strojnega učenja in podatkovnega rudarjenja ter podrobneje razložil nekatere pomembnejše algoritme in statistične tehnike, ki bodo bralcu pomagale pri boljšem razumevanju predmeta primerjave. V drugem delu smo analizirali funkcionalnosti, posebnosti orodij, uporabniško izkušnjo in možnosti implementacije funkcionalnosti platform v lastne rešitve ter navedene lastnosti med seboj podrobneje primerjali. V eksperimentalnem delu smo izvedli eksperiment, v katerem smo primerjali učinkovitost algoritmov na vzorčnih podatkovnih množicah in s pomočjo statističnih testov ugotavljali, ali so razlike statistično pomembne.

2 STROJNO UČENJE

Strojno učenje obsega študij in izdelavo algoritmov, za katere pravimo, da se »učijo« na podlagi izkušenj. Bolj specifično je cilj strojnega učenja ugotavljanje in posplošitev pravil, ki jih pridobimo s takšno ali drugačno analizo podatkovnih množic [2]. Je temelj sodobnega podatkovnega rudarjenja, ki ga uporabljamo za iskanje in opisovanje strukturiranih vzorcev v podatkih. Novo pridobljeno razumevanje podatkov uporabimo za bolje informirane odločitve, ki botrujejo neki pridobitvi, večinoma v ekonomskem ali znanstvenem smislu [3].

2.1 Tipi strojnega učenja

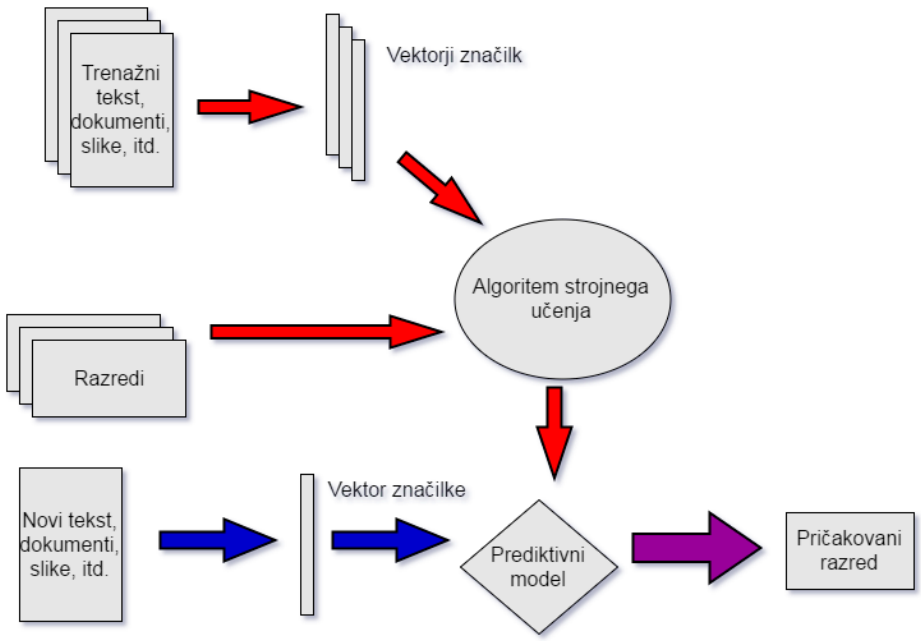
Algoritme v strojnem učenju delimo glede na tipe problemov, ki jih rešujejo. Raznolikost in velika izbira algoritmov omogočata reševanje praktično vsakršnega problema, ki bi se lahko pojavil.

2.1.1 Nadzorovano učenje

Algoritmi nadzorovanega učenja gradijo odločitvene modele s pomočjo znanih klasifikacij primerkov. Modele uporabimo za reševanje problemov klasifikacije in regresije, odvisno od cilja. Razlika med regresijo in klasifikacijo je v rezultatu funkcije. Klasifikacija primerku dodeli diskreten razred, medtem ko regresivna funkcija izračuna zvezno vrednost na podlagi atributov primerka [4]. Ker v eksperimentalnem delu uporabimo tri algoritme klasifikacije, jim velja nameniti nekoliko več prostora.

Klasifikacija je proces, pri katerem s pomočjo klasifikatorja primerke umestimo v enega izmed prej določenih razredov. Primerke oziroma vektorje značilke sestavljajo posamezni atributi, ki so neodvisne diskretne ali zvezne spremenljivke. Razred je rezultat funkcije, katere vrednost je odvisna od atributov posameznega primerka in v primerku nastopa kot odvisna diskretna spremenljivka. Funkcija je bodisi že podana bodisi pridobljena s pomočjo algoritma [5]. Algoritem klasifikacije poteka v treh delih. V prvem delu zgradimo prediktivni model, na podlagi katerega se kasneje odločimo, kateremu razredu pripada posamezni primerek. V ta namen uporabimo učno množico podatkov, pri katerih je razred

že znan. V drugem delu na tak način pridobljeni model preizkusimo na testni množici, s pomočjo katere preverimo zanesljivost modela. Če je rezultat zadovoljiv, gremo na tretji del, kjer uporabimo model na novih podatkih, katerih razrede skušamo napovedati [4]. Na sliki 2.1 so prikazani vsi trije deli klasifikacije. Rdeče puščice povezujejo podatke in klasifikacijski algoritem, ki so potrebni za izgradnjo modela. Elemente, ki so povezani z modrimi puščicami, kot vhod vstavimo v ustvarjeni prediktivni model, ta pa vstavljenemu primerku pripiše pričakovani razred.



Slika 2.1: Prikaz poteka nadzorovanega učenja

Glede na način učenja delimo lene in takojšnje algoritme. Leni algoritmi shranijo podatke o učni množici in čakajo na poizvedbo po klasifikaciji primerka, preden zgradijo model odločanja. Tovrstni algoritmi so zelo hitri pri grajenju modela, saj je samega učenja v tem času zelo malo, zato pa toliko dlje traja klasifikacija posameznih primerkov. Takojšnji algoritmi ustvarijo model med samim učenjem. Kasnejša uporaba sistema ne vpliva na zgrajeni model, tako da je pričakovani izhod y_i enak za vsak primerek x_i [6].

Kot primer vzemimo filter odvečne pošte poljubnega ponudnika elektronske pošte. Filter klasificira vsako elektronsko sporočilo v enega izmed dveh razredov, »odvečno« ali »ni odvečno«, kot vhod pa prejme vektor z logičnimi izrazi $X = (x_1 \dots, x_j, \dots, x_d)$, kjer je $x_j = 1$, če se j-ta beseda v slovarju pojavi v sporočilu, sicer je vrednost $x_j = 0$. Uporabnik v namene učenja in oblikovanja modela vnese učno množico primerov (x_i, y_i), kjer je $X_i =$

$(x_{i,1}, \ldots, x_{i,d})$ znan vnos in y_i izhod, ki nam poda informacijo o klasifikatorju. Model je primeren, kadar nam klasifikator v izhodu poda primeren razred y_t za značilko x_t [2].

2.1.2 Učenje z ojačitvijo

Pri učenju z ojačitvijo stroj vpliva na okolico z dejanji $a_1 \ldots, a_n$, okolica pa se glede na dejanja spreminja. Glede na spremembe v okolici stroj prejme kazen ali nagrado, izraženo s skalarjem $r_1, \ldots, r_m$. Cilj algoritma je, da maksimira prejemanje nagrad in minimizira prejemanja kazni. S takimi algoritmi rešujemo probleme računalniškega igranja kart, prav tako pa se obširno uporablja v robotiki [7]. Algoritmi te vrste praviloma temeljijo na Markovem procesu odločanja [8].

2.1.3 Nenadzorovano učenje

Pri nenadzorovanem učenju algoritem kot vhod prejme primerke $x_1, x_2 \ldots, x_n$ in skuša med seboj podobne postaviti v skupino ali gručo. Problem nenadzorovanega učenja lahko poenostavimo na učenje verjetnostnega modela podatkov. Algoritmi nenadzorovanega učenja ustvarijo model, ki predstavlja verjetnostno porazdelitev nekega vnosa x_t glede na prejšnje vnose $x_1, x_2 \ldots, x_{t-1}$. Tako model za vnos x_t primerek postavi glede na rešitev enačbe pogojne verjetnosti $P(x_t | x_1, x_2 \ldots, x_{t-1})$. Pomembna razlika med nenadzorovanim učenjem in drugimi tipi učenja je, da učiteljev nadzor nad izvajanjem ni potreben. Algoritem za uspešno učenje in izvajanje ne potrebuje niti klasifikacije učitelja niti nagrad in kazni, ki bi popravljale obnašanje. Celoten proces lahko razumemo kot odkrivanje strukturiranih vzorcev v podatkih [7].

2.2 Metode strojnega učenja

Za namene izvajanja eksperimenta diplomskega dela bomo uporabili tri algoritme klasifikacije. Ti algoritmi so k-najbližjih sosedov, Naivni Bayes in Naključni gozd. Za validacijo smo uporabili metodo k-kratno navzkrižno metodo. Za jasnejšo predstavo metode Random Forest bomo na strnjen način predstavili koncept odločitvenih dreves.

2.2.1 Klasifikacijski algoritmi

K-najbližjih sosedov

Algoritem k-najbližjih sosedov (angl. k-nearest neighbours) deluje na načelu, da si bodo podobni primerki, predstavljeni kot točke v nekem v n-dimenzionalnem prostoru, bližje kot primerki, med katerimi so večje razlike. Klasifikacija novega primerka poteka tako, da pregledamo razrede k najbližjih sosedov in novemu primerku dodelimo razred, ki je pri sosedih najpogostejši. V splošnem lahko primerke razumemo kot točke v n-ti dimenziji, kjer je n število atributov primerka. Absolutni položaj primerka ni toliko pomemben, kot je pomemben relativni položaj glede na druge primerke. Cilj klasifikacije je minimizirati razdaljo med podobno klasificiranimi primerki in maksimirati razdaljo med različno klasificiranimi primerki [5].

Algoritem lahko strnemo v zaporedje petih korakov [9]:

1. Izberemo pozitivno število k, ki predstavlja število pregledanih najbližjih sosedov.
2. Izberemo primerek, ki ga želimo klasificirati.
3. Izberemo k primerkov, ki so danemu primerku po atributih najbolj podobni.
4. Najdemo najpogostejši razred izbranih primerkov.
5. Najpogostejši razred je klasifikacija izbranega primerka.

Za izračun razdalje med primerki obstaja več metod. Nekatere najpomembnejše so predstavljene v tabeli 2.1.

Tabela 2.1 Najpogostejše enačbe za računanje razdalje med primerki

Evklidski: $D(x,y) = (\sum_{i=1}^{m} \mid x_i - y_i \mid^2)^{1/2}$
Minkovsky: $D(x,y) = (\sum_{i=1}^{m} \mid x_i - y_i \mid^r)^{1/r}$
Manhattan: $D(x,y) = \sum_{i=1}^{m} \mid x_i - y_i \mid$
Camberra: $D(x,y) = \sum_{i=1}^{m} \frac{\mid x_i - y_i \mid}{\mid x_i + y_i \mid}$

Naivni Bayes

Naivni Bayes je klasifikacijski algoritem, ki temelji na Bayesovem teoremu s predpostavko o pogojni neodvisnosti med posameznimi atributi primerka. Ker predpostavka ni vedno pravilna, pravimo, da je algoritem »naiven«. Bayesijanski klasifikatorji dodeljujejo najverjetnejši razred danemu primerku, kot ga opišejo atributi. Pri predpostavki, da so atributi medsebojno neodvisni, velja [10]:

$$P(X|C) = \prod_{i=1}^{n} P(x_i|C) \tag{2.1}$$

kjer so:

$X = (x_1, x_2, x_3 \ldots, x_n)$ atributi primerka

$P(x_i|C) = \frac{P(C|x_i)P(C)}{P(x_i)}$

Klasifikacija je končana, kadar za vsak razred glede na atribute izračunamo pogojne verjetnosti in primerku dodelimo razred, za katerega velja najvišja pogojna verjetnost.

Odločitvena drevesa

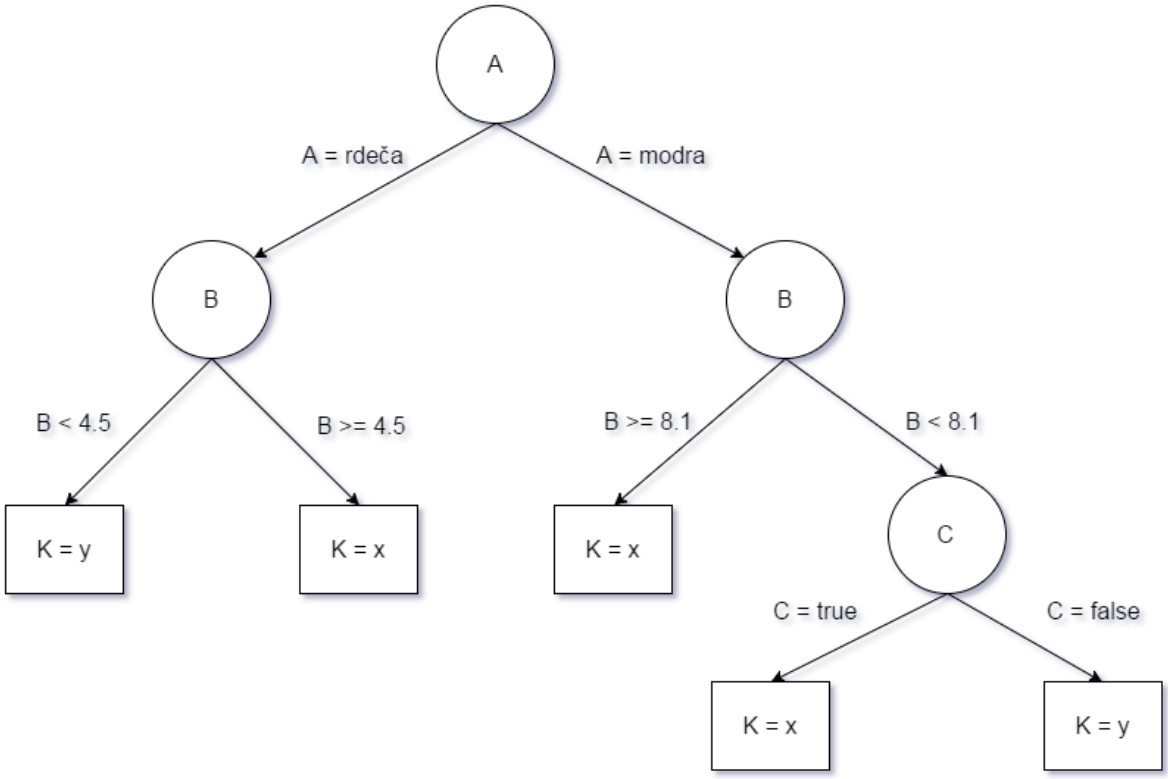
Metoda odločitvenih dreves s pomočjo učnih množic generira model in ga predstavi kot odločitveno drevo. Posamezne primerke klasificira tako, da jih sortira od korenskega vozlišča do nekega lista drevesa, ki primerku poda razred. Vsako vozlišče v drevesu je abstrakcija za test nekega atributa primerka. Vsaka veja iz vozlišča sodi k enemu izmed

možnih vrednosti atributa. Postopek algoritem ponavlja tako dolgo, da ne pride do enega izmed listov drevesa. Od lista drevesa je odvisno, kakšno vrednost razreda pridobi primerek [5].

Odločitvena drevesa uporabimo, kadar problemi izpolnjujejo naslednje značilnosti [11]:

- Primerki so predstavljeni s fiksnim številom atributov, ki imajo majhno število mogočih disjunktnih vrednosti.
- Razredi zavzemajo diskretne vrednosti.
- Zahtevani so disjunktni opisi atributov.
- Podatki učne množice vsebujejo manjkajoče vrednosti atributov.

Na sliki 2.2 je podan primer odločitvenega drevesa. Krogci predstavljajo posamezna vozlišča, kvadratki pa liste. Posamezne črke v vozliščih predstavljajo poizvedbe po vrednostih atributov primerka. Algoritem pri obravnavi primerka vedno presodi korensko vozlišče, ki v primeru na sliki presodi, ali je primerek rdeče ali modre barve. Glede na presojo napreduje po levem ali desnem otroku vozlišča. Algoritem se vedno zaključi z listom. V našem primeru lahko do lista pride na prvi ali drugi ravni, odvisno od primerka. Z zaključkom razredu K dodeli vrednost x ali y.



Slika 2.2 Odločitveno drevo

Naključni gozd

Naključni gozd (angl. Random Forest) je tehnika klasifikacije, ki izboljša zanesljivost klasifikacije algoritmov posameznih odločitvenih dreves. Odločitvena drevesa imajo namreč predispozicijo prekomernega prileganja na učno množico, če so zelo globoka. Metoda naključnega gozda zgradi več sto ali tisoč različnih dreves, ki občutno zmanjšajo mero napak pri kvalifikaciji enega samega drevesa [12] .

Ansambel dreves tvorimo na naslednji način [13]:

1. Tvorimo množico podatkov iz učne množice z metodo ponovnega vzorčenja. Število primerkov v ustvarjeni množici podatkov je enaka kot v učni množici.
2. Za vsako drevo izberemo naključno število atributov. Izbrane atribute s pomočjo algoritmov za izgradnjo dreves uporabimo za generiranje vozlišč in listov novega drevesa v ansamblu.
3. Brez rezanja vsako drevo zgradimo do konca in ga vstavimo v končni klasifikacijski ansambel dreves.

Primerke, ki se v na novo ustvarjeni množici ne nahajajo, skupaj s testno množico uporabimo za testiranje tako individualnih dreves kot celotnega gozda. Pri klasifikaciji primerka vsako izmed tako generiranih dreves iz ansambla poda svoj glas. Primerku dodelimo tisti razred, za katerega je glasovalo največ dreves [13].

2.2.2 Gručenje

Najbolj znan primer algoritma gručenja je najbrž algoritem k-means. Dovoljuje le numerične atribute, ker predpostavlja, da je za vsak atribut primerkov mogoče izračunati aritmetično sredino. Osnovna različica algoritma je sestavljena iz treh delov: inicializacije, dodelitve primerkov h gruči in posodobitve lokacije centroidov [14].

Inicializacija

Pri procesu inicializacije moramo definirati, na koliko gruč K želimo razdeliti podatke, opredeliti primerke oz. točke $x_1, x_2 \ \ldots, x_n$ in na naključne lokacije v d – dimenzionalnem prostoru –postaviti K centroidov c, kjer je d število atributov primerkov. Centroid je definiran kot vektor velikosti d z naključno generiranimi vrednostmi in predstavlja središče ene izmed gruč [15].

Dodelitev primerkov h gruči

Vsak primerek dodelimo eni izmed gruč. To storimo s primerjanjem razdalje vsakega primerka z vsakim centroidom gruče. Poljubna točka spada v gručo, če velja:

$$argmin\, D(x_i, c_j) \tag{2.2}$$

$D(x_i, c_j)$ je razdalja med primerkom x_i in centroidom gruče j. Razdaljo izračunamo z najbolj primerno metodo glede na svoj problem. Torej primerek x_i, ki ima najkrajšo razdaljo do centroida c_j, pripišemo h gruči j [15].

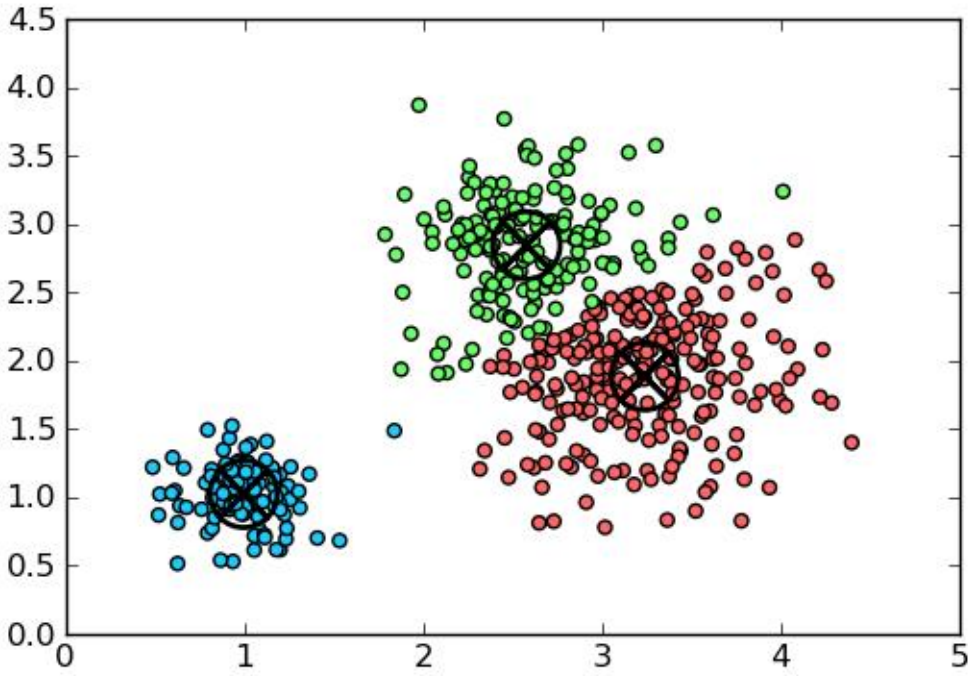
Posodobitev lokacije centroid

Ko vsak primerek dodelimo k neki gruči, lokacijo centroidov posodobimo na aritmetično sredino atributov primerkov gruče, ki gruči centroida pripadajo [15]. Torej za vsako gručo $j = 1 \ ... \ g$ izračunamo nove koordinate centroida c po enačbi:

$$c_j(a) = \frac{1}{c_j} \sum_{x_i \to c_j} x_i(a) \tag{2.3}$$

Za $a = 1 \ ... \ d$, kjer je a neki specifični atribut.

Algoritem ponavljamo toliko časa, da se nobena izmed lokacij centroidov v zadnjem delu algoritma več ne posodobi na novo točko. Povedano drugače, algoritem se ustavi, kadar sistem doseže točko konvergence. Končni rezultat nekega primera izvedbe algoritma, kjer je število gruč $K = 3$, vidimo na sliki 2.3.



Slika 2.3 Vizualizacija rezultata algoritma k-means

(Vir: http://blog.mpacula.com/wp-content/uploads/2011/04/kmeans1.png)

2.2.3 K-kratna navzkrižna validacija

V praktičnem delu bomo za namene učenje množice uporabili k-kratno navzkrižno validacijo (angl. k-cross validation*)*. Pri tej metodi podatke razdelimo na *k* rezov (angl. fold) približno enake velikosti in postopek učenja izvedemo *k*-krat. Pri vsaki iteraciji je eden izmed rezov testni, *k*-1 rezov pa učnih. Ko algoritem zaključi z vsemi *k* iteracijami učenja, izračunamo povprečne napake vseh *k* poskusov. Glavna prednost metode je, da postane veliko manj pomembno, na kakšen način razdelimo podatke v učne in testne reze, vendar pa je prav zaradi tega algoritem približno *k*-krat počasnejši kot pri standardni metodi »holdout«. Postopek služi za natančnejše napovedovanje točnosti modela v primerih, ko je razpoložljivost podatkov slabša. Pravo mero napake izračunamo kot povprečje napak vseh rezov:

$$E = \frac{1}{k}\sum_{i=1}^{k} E_i \tag{2.4}$$

kjer je:
E_i – mera napake posameznega reza,
k – število rezov,
E –prava mera napake.

2.2.4 Mere kakovosti klasifikacije

Kakovost klasifikacije bomo v praktičnem delu ocenjevali s konceptoma klasifikacijske točnosti (angl. classification accuracy) in F-mere (angl. F-measure ali F-score).

Klasifikacijska točnost

Klasifikacijska točnost je odvisna od števila pravilno klasificiranih primerkov v vzorcu. V praksi to pomeni verjetnost pravilne klasifikacije naključnega primerka v vzorcu. Točnost izračunamo po formuli:

$$T = \frac{p}{n} \tag{2.5}$$

kjer je:

T – delež pravilno klasificiranih primerkov (za izraz v odstotkih pomnožimo s 100),

p – število pravilno klasificiranih primerkov,

n – skupno število primerkov.

F-mera

F-mera F_1 je mera točnosti klasifikacije glede na priklic in preciznost vzorca. Vprašamo se, kolikšen delež izbranih primerkov je relevanten glede na poizvedbo.

Preciznost definiramo kot:

$$preciznost = \frac{tp}{tp + fp} \tag{2.6}$$

kjer je:

tp – število pravilno klasificiranih pozitivnih primerkov in

fp – število primerkov napačno klasificiranih kot pozitivni.

Pri priklicu se vprašamo o deležu izbranih primerkov, ki so relevantni. Definiramo ga kot:

$$priklic = \frac{tp}{tp + fn} \tag{2.7}$$

kjer je:

tp – število pravilno klasificiranih primerkov in

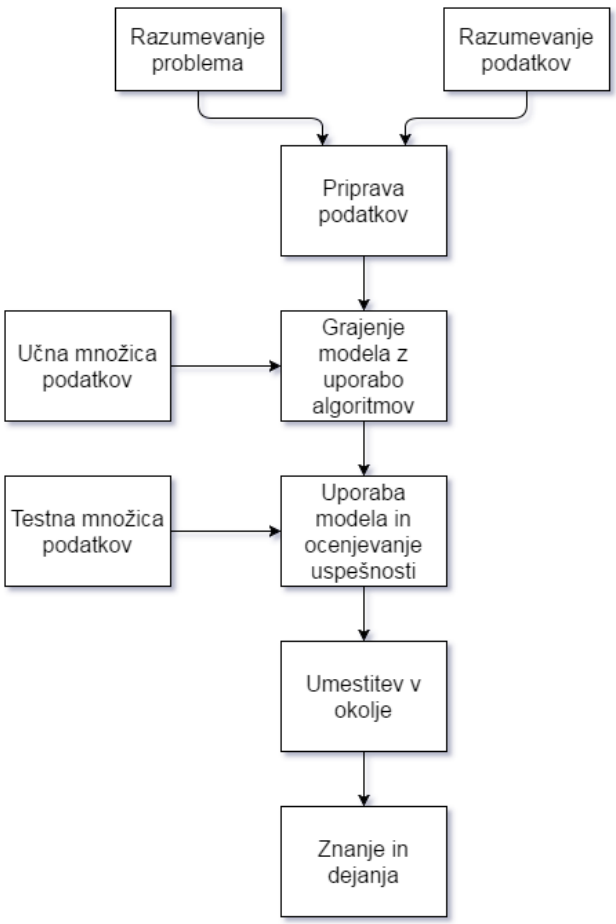
fn – število primerkov napačno klasificiranih kot negativni.

F-mera F_1 je harmonična sredina preciznosti in priklica.

$$F_1 = 2 * \frac{preciznost * priklic}{preciznost + priklic} \tag{2.8}$$

2.3 Metodologije uporabe strojnega učenja pri podatkovnem rudarjenju

Rezultati algoritmov so predvidevanja o novih primerkih, na primer nakupovalne navade glede na vektorja značilk specifičnega primerka [3]. Metodologijo odkrivanja povezav med vzorci formalizira več okvirjev, med katerimi je najpopularnejši Cross Industry Standard Process for Data Mining (CRISP-DM), ki ga je razvil konzorcij podjetij, ki se ukvarjajo s podatkovnim rudarjenjem in strojnim učenjem. Omembe vredni so tudi Sample, Explore, Modify, Model and Assess (SEMMA); Define, Measure, Analyze, Improve and Control (DMAIC); in Selection, Preprocessing, Transformation, Data Mining, Interpretation and Evaluation. Vsi omenjeni okvirji imajo veliko skupnih značilnosti. Kotu pravi, da jih lahko povzamemo v generičnem procesu, prikazanem na sliki 2.4, ki je enak ne glede na uporabljena orodja [3].



Slika 2.4 Proces podatkovnega rudarjenja

Razumevanje problema

Razumevanje je predpogoj za uspešno izvajanje procesa učenja. Problem moramo dobro definirati in ugotoviti, na kakšen način bomo prišli do želenega cilja, predvsem z izbiro primernega algoritma [3].

Razumevanje podatkov

Razumevanje načina zbiranja, shranjevanja, transformiranja, poročanja in uporabe podatkov je poglavitno za razumevanje informacij, ki jih pridobimo z algoritmi. Oceniti moramo kakovost, kvantiteto in razpoložljivosti podatkov ter, kako so ta dejstva povezana s pomanjkanjem ali napačnimi podatki v sistemu [3].

Priprava podatkov

Podatki morajo biti v formatu, primernem za obdelovanje algoritma, kar pomeni, da so strukturirani v tabelaričnem formatu z vnosi po vrsticah in atributi po stolpcih. Če so podatki v drugačnem formatu, jih moramo transformirati, na primer s transponiranjem. Uporabimo metode Exploratory Data Analysis (EDA), s pomočjo katerih odkrijemo strukturo, nekonsistentnosti, maksimume in minimume ter zakonitosti povezav med posameznimi podatkovnimi točkami. Zagotoviti je treba ustrezno kakovost podatkov. V te namene uporabimo tehnike predprocesiranja, kot so odstranitev duplikatov, odstranitev osamelcev, substitucija manjkajočih podatkov in standardizacija atributov [3].

Grajenje modela z uporabo algoritmov

Cilj in razpoložljivost podatkov kažeta, kateri algoritem uporabimo. Uporabimo prej pripravljene podatke v namene grajenja modela. Poznati moramo vse atribute predpripravljenih podatkov, vključno z razredom [3].

Uporaba modela in ocenjevanje uspešnosti

Pridobljeni model preizkusimo proti drugi množici podatkov. Če znanje in podatki, ki jih uporabimo v učni množici, ne zagotavljajo primerne ocene klasifikatorja, se bo algoritem

težko prilagajal glede na nove primerke [3]. Ta pojav imenujemo prekomerno prileganje. V praksi je to pojav, ko je na primer v učni množici algoritem točen 100 % časa, v testni pa 50 %, namesto da bi bil v obeh primerih točen pri 75 % primerkov [16].

Umestitev v okolje

Uvedemo tehnično integracijo orodij in modela v poslovne procese. Pri tem moramo narediti kompromis med časom grajenja modela in časom izdelovanja specifičnih napovedi in analize. Nekateri algoritmi so v fazi gradnje modela namreč počasnejši, vendar lahko po vzpostavitvi modela do posameznih napovedi dostopamo zelo hitro. Razmisliti je treba o fleksibilnosti ponovnega modeliranja, predvsem glede na hitrost sprememb poslovnih pravil in ciljev [3].

Znanje in dejanja

Organizacija preko podatkovnega rudarjenja pridobiva nova znanja, na podlagi katerih sprejema bolje informirane odločitve [3].

3 TEORETIČNI OPIS IN PRIMERJAVA PLATFORM RAPIDMINER IN WEKA

V tem poglavju smo se lotili prvega dela problema diplomskega dela. Sistematično smo opisali glavne značilnosti, prednosti in slabosti obeh orodij in nato orodja med seboj primerjali. Za obe orodji smo obravnavali preprosto implementacijo algoritma klasifikacije v uporabniškem vmesniku in z uporabo knjižnic programskega jezika Java. Še prej smo definirali primerno metodologijo, ki nas je vodila pri identifikaciji in ocenjevanju najpomembnejših aspektov tovrstnih orodij.

3.1 Metodologija

Metodologijo, ki smo jo uporabili za primerjavo orodij, smo povzeli po metodologiji, ki so jo razvili Chen et al. v članku »A Survey of Open Source Data Mining Systems« [17], in sicer z namenom ugotavljanja razlik med kakovostjo različnih vidikov odprtokodnih orodij. Sistem smo obravnavali iz treh vidikov, ki jih bomo definirali v naslednjih podpoglavjih.

3.1.1 Splošne značilnosti

V obzir vzamemo značilnosti, kot so uporabljena licenca, programski jezik, v katerem je orodje napisano, minimalne specifikacije poganjanja orodja in združljivost z operacijskimi sistemi [17]. Posebno pozornost velja posvetiti morebitnim razlikam med zastonjskimi in plačljivimi različicami orodij ter kako močno omejijo uporabnika v primeru, da za orodje ni pripravljen plačati.

.

3.1.2 Vidik podatkovnih virov in funkcionalnosti

V praksi podatki prihajajo tako iz podatkovnih baz in skladišč kot tudi iz nepovezanih datotek. Pomembno je, da orodje podpira čim več podatkovnih formatov. Funkcionalnost platforme definiramo kot sposobnost reševanja raznolikih problemov na ustrezen in učinkovit način. Pričakuje se, da bo sistem nudil osnovna orodja za podatkovno rudarjenje,

kot so odločitvena drevesa, Bayesianske metode in gručenje k-means kot tudi sodobnejše metode. Posebej izpostavimo naslednje značilnosti [17]:

Predpriprava podatkov

Sama predpriprava v procesu podatkovnega rudarjenja lahko traja dlje časa in je pogosto ključna pri doseganju rešitve problema. Dober sistem mora nuditi čim več obdelovalnih funkcionalnosti [17].

Integracija raznolikih tehnik

Za reševanje poslovnih in akademskih problemov uporabljamo veliko različnih tehnik, ki so odvisne od dejavnikov, kot sta tip problema in razpoložljivost podatkov. Zaradi tega je za uporabnika pomembno, da sistem v svoje delovanje integrira čim več orodij, kot na primer algoritme za ustvarjanje prediktivnih modelov, orodja za statistično obdelovanje rezultatov in več [17].

Skalabilnost

Pomembno je, da je sistem sposoben prilagajati količino sredstev glede na to, kako zahtevne so operacije, ki jih uporabnik zahteva.

3.1.3 Uporabniški vidik

Orodje mora biti enostavno priučljivo tako za začetnega kot naprednega uporabnika, tako da bo sposoben reševanja kompleksnih problemov v realnem svetu. Omenimo nekaj dejavnikov, ki krojijo kakovost izkušnje uporabnika [17]:

Vizualizacija podatkov in modelov

Razumevanje podatkov in modelov je ključ do rešitve problema, zato tako začetniki kot eksperti potrebujejo dobra orodja za vizualizacijo [17].

Razširljivost

Pomembno je, da odprtokodni sistemi za podatkovno rudarjenje nudijo elemente arhitekture za razvoj in implementacijo novih metod. Dobra razširljivost pomeni enostavno integracijo uporabnikovih lastnih rešitev [17].

Skupnost

Odprto-kodna narava obravnavanih sistemov omogoča interesentom, da pomagajo pri vzdrževanju in posodabljanju. K temu sodita tudi pomoč na spletnih forumih in razvoju razširitev, če orodje to omogoča [17].

3.2 RapidMiner

RapidMiner je platforma in popolno integrirano okolje za strojno učenje, podatkovno rudarjenje, tekstovno rudarjenje, prediktivno analitiko in poslovno inteligenco. Platformo razvija istoimensko podjetje RapidMiner. Zaščitena je pod komercialno licenco Business Source, ki poskuša skleniti kompromis med prednostmi odprtokodnega načina razvoja in zadostnim zaslužkom, da lahko podjetje zagotovi nadaljnje razvijanje platforme na profesionalni ravni. V primeru RapidMinerja je koda osnovnih funkcionalnosti podatkovnega rudarjenja dostopna vsem, za dodatne možnosti, ki so nadgradnja osnove, pa je treba plačati. Predlog Wideniusa in Nymana je, da po določenem času koda, zaščitena pod licenco Business Source, preide na odprtokodno licenco [18]. V RapidMinerjevem primeru je to AGPL v3 (angl. Affero General Public Licence v3). Prehod z ene licence na drugo se zgodi, kadar določena različica platforme zastara kot posledica izida nove izdaje. V delu obravnavamo RapidMiner 7.1, ki je v času pisanja najnovejša različica.

RapidMiner Studio

RapidMiner Studio služi kot orodje za uvažanje in predprocesiranje podatkov, izvajanje metod podatkovnega rudarjenja in opravljanje statistične analize rezultatov. Podpira tako lokalno izvajanje dela kot delo z uporabo tehnologije RapidMiner Cloud. Sistemske zahteve so prikazane v tabeli 3.1 [19]:

Tabela 3.1 Sistemske zahteve

Minimalne zahteve	**Priporočene zahteve**
Dvojedrni procesor z 2 GHz ali hitreje	Štirijedrni procesor s 3 GHz ali hitreje
4 GB RAM	12 GB RAM
Več kot 1 GB prostega prostora na disku	Več kot 100 GB prostega prostora na disku
32-bit OS	64-bit OS
Java 7	Java 8 (zahtevano v primeru Linux OS)

Zahteva po pomnilniku in prostem prostoru na disku je visoka zato, ker najbolj vplivata na velikost podatkovnih množic, ki jih lahko v orodje naložimo in analiziramo. Vodič RapidMiner Studio predlaga [19], da za najboljšo izkušnjo uporabljamo kar se da močan sistem. Orodje je napisano v programskem jeziku Java in je posledično kompatibilno z vsemi operacijskimi sistemi, za katere je na voljo primeren JRE (angl. Java Runtime Environment). Najbolj relevantni so Windows 7, Windows 8, Windows 8.1, Windows 10, Linux in MacOS X 10.8-10.10.

RapidMiner Studio je na voljo v zastonjski različici, imenovani RapidMiner Studio Community in plačljivi RapidMiner Studio Professional, ki je namenjen podatkovnemu rudarjenju za profesionalne namene. Plačljiva različica ponuja dodatne možnosti ponovne uporabe procesov in poveča anonimnost podatkov, prav tako pa zagotavlja štiriindvajseturno tehnično podporo. Uporabniki zastonjske različice se morajo zadovoljiti s spletnimi viri in forumi RapidMiner [20].

RapidMiner Server

Uporabniki se za namene kolaboracije in večje procesne moči lahko preko RapidMiner Studia povežejo na strežnik, ki omogoča oddaljeno in načrtovano izvajanje analiz procesov, repozitorije za delo v skupinah, upravljanje z uporabniki in spletni dostop do poročil in rezultatov skupnega dela. Strežnik je optimiziran za obdelovanje velike količine podatkov. Izvajanje operacije je mogoče implementirati s strežnika neposredno v poslovne procese podjetja preko spletnih storitev ali za to namenjenega API. Glavni namen uporabe je v poslovne in akademske namene [21]. Na voljo je samo v plačljivi različici, vendar lahko

interesenti na spletni strani podjetja prenesejo demonstracijsko različico. Cena strežnika ni fiksna, temveč podjetje postavi ponudbo glede na potrebe interesenta [20].

RapidMiner Radoop

Radoop je vmesnik med posameznimi orodji RapidMiner in strežniško gručo, ki poganja operacije na platformi Apache Hadoop. Namestimo ga kot razširitev v RapidMiner Studio ali RapidMiner Server ter na tak način omogočimo ETL (angl. Extract, Transform, Load), podatkovno analizo in izvajanje procesov strojnega učenja v okolju Hadoop [22]. RapidMiner Radoop je na voljo v različicah RapidMiner Studio Professional in RapidMiner Server v obliki razširitve [20].

RapidMiner Cloud

RapidMiner Cloud uporabniku omogoča izvajanje operacij in dostop do analitičnih modelov, ne glede na katerem računalniku delo izvaja. Največja prednost izvajanja procesov na oblaku je, da sprostimo sredstva na domačem računalniku, prav tako pa lahko na oblaku izvajamo več procesov paralelno. RapidMiner Cloud je dostopen na obeh različicah RapidMiner Studio, vendar je Community različica hudo omejena pri velikosti repozitorija, delovnem pomnilniku in procesni moči. Če uporabnik ne potrebuje zmogljivosti inačice Professional, lahko kupi naročnino na storitev Cloud Professional, ki na cenovno dostopnejši način poveča zmogljivost sistema Cloud [23]. Čas procesiranja je izražen v kreditnih točkah. Med izvajanjem operacij na oblaku sistem od računa uporabnika odteguje kreditne točke, ki jih je možno dokupiti, kot to vidimo v tabeli 3.2. Kreditne točke se vsak mesec obnavljajo [23].

Tabela 3.2 Ponudbe storitve RapidMiner Cloud

Uporaba storitev RapidMiner Cloud	**RapidMiner Studio Community**	**RapidMiner Studio Professional**	**Naročnina Cloud Professional**
Cena	Zastonj	Vključen v ceno različice Professional	$ 79/mesec
Velikost repozitorija	20 MB	5 GB	5 GB
Velikost delovnega pomnilnika	8 GB	16 GB, 32 GB, 64 GB	16 GB, 32 GB, 64 GB
Kreditnih točk na mesec	100	200	200
Dodatne kreditne točke	Za nakup dodatnih točk je potrebna naročnina Cloud Professional	$ 19 na 100 kreditov	$ 19 na 100 kreditov

3.2.1 Vidik podatkovnih virov in funkcionalnosti

Predpriprava podatkov

Platforma podpira praktično vse najmodernejše in najbolj uporabljene podatkovne vire, vendar je zastonjska različica v tem vidiku hudo omejena. Od nepovezanih formatov podpira samo razpredelnice Excel in CSV, poveže pa se lahko samo na odprto-kodne podatkovne baze, kot sta MySql in Postgre. Izdaja Professional poleg omenjenega podpira še večino komercialnih podatkovnih baz, prej omenjeni Hadoop, NoSql in množico drugih nepovezanih podatkovnih virov, kot na primer Wekin arff in SPSS [20]. RapidMiner podatkovne tipe pri uvozu podatkovnih množic definira na svojstven način, prikazan v tabeli 3.3 [24]:

Tabela 3.3 Podatkovni tipi platforme RapidMiner

Podatkovni tip	**RapidMiner ime**	**Uporaba**
Nominalni	nominal	Kategorične ne numerične vrednosti
Številčne vrednosti	numeric	Za splošne številčne vrednosti
Cela števila	integer	Vsa cela števila
Realna števila	real	Vsa realna števila
Tekst	text	Tekst brez strukture
Dvovrednostni nominalni	binominal	Poseben primer nominalne vrednosti, kjer sta dovoljeni le dve vrednosti
Večvrednostni nominalni	polynominal	Poseben primer nominalne vrednosti, kjer sta dovoljeni več kot 2 vrednosti
Date Time	date_time	Datum in čas
Date	date	Datum
Time	time	Čas

Za morebitne težave z uvoženimi podatki RapidMiner ponuja orodja 2-D in 3-D vizualizacije prikaza odvisnosti. Platforma ponuja močna orodja za transformacijo množic podatkov. Naštejmo nekaj najpomembnejših [25]:

- Agregacija
- Diskretizacija
    - Po velikosti
    - Po številu košev, navedenih s strani uporabnika
    - Po frekvenci
    - Po uporabniški specifikaciji
    - Po entropiji
- Normalizacija
    - Normalizacija vrednosti atributov
    - Vzvratna normalizacija

- Filtriranje
    - Filtriranje primerkov glede na pogoj
    - Filtriranje primerkov glede na interval
- Vzorčenje
    - Naključno vzorčenje
    - Stratificirano vzorčenje
    - Ponovno vzorčenje
    - Kennard-Stone vzorčenje
- Analiza glavnih komponent (angl. Principal Component Analysis)
- Upravljanje z manjkajočimi vrednostmi
    - Zamenjava manjkajočih vrednosti z vrednostmi s specificiranimi vrednostmi
    - Zamenjava manjkajočih vrednosti z vrednostmi, pridobljenimi s prediktivnim modelom
    - Zamenjava manjkajoče vrednosti z »NaN« vrednostjo
    - Zamenjava neskončnih vrednosti s specificiranimi vrednostmi
    - Odstranjevanje neuporabljenih nominalnih vrednosti

Zgornji seznam še zdaleč ni izčrpen, vendar dobro oriše raven nadzora, ki ga uporabnik z uporabo platforme ima nad obdelavo podatkov. Posebnih razlik med zastonjsko in plačljivo različico, kar se tiče predprocesiranja, ni.

Integracija različnih tehnik

RapidMiner podpira praktično vsakršno metodo strojnega učenja in validacije zahvaljujoč dolgoročnemu razvoju razvijalcev in odprto-kodnega značaja ter razširljivosti platforme. Prav zaradi razširljivosti lahko vešči uporabniki razvijejo nove metode ali prilagodijo obstoječe različice za uporabo v RapidMinerju [26]. V tabeli 3.4 je naštetih nekaj najpomembnejših metod strojnega učenja in metod za ocenjevanje modelov, ki jih RapidMiner podpira [20] [27]. Tako pridobljene podatke lahko analiziramo s statističnimi testi, kot sta Studentov t-Test in ANOVA.

Tabela 3.4 Najpomembnejše funkcionalnosti podatkovnega rudarjenja platforme RapidMiner

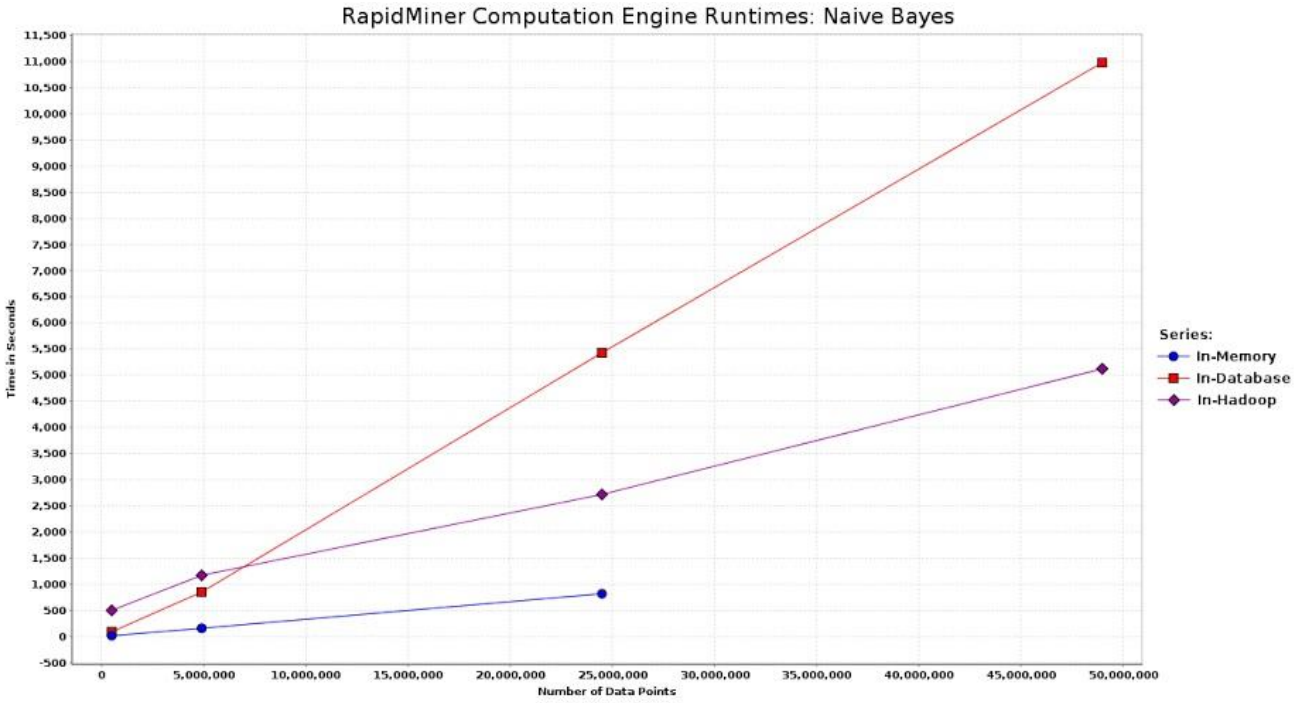
Ugotavljanje povezav med podatki	**Prediktivna analiza**	**Preskriptivna analiza**	**Ocenjevanje modelov**
Korelacija	Odločitvena drevesa	Simulacije	Navzkrižna validacija
Asociacija	Bayesovi klasifikatorji	Optimizacija izidov	ROC
Gručenje	Metoda podpornih vektorjev		Test statistične značilnosti
	Nevronske mreže		Validacija »izpusti-enega«
	Analiza diskriminant		Testiranje nazaj

Skalabilnost

Skalabilnost je karakteristika, kjer so razlike med brezplačno in plačljivo različico največje. Pri brezplačni različici imamo izbiro poganjanja operacij na lokalnem stroju ali na oblaku [24]. V primeru, da izvajamo operacije na lokalnem stroju, smo omejeni na zmogljivost strojne opreme, pri uporabi tehnologije RapidMiner Cloud pa na izredno malo prostora v repozitoriju (20MB), majhno količino pomnilnika in malo kreditnih točk [23].

RapidMiner Studio Professional kot dodatno možnost ponuja množico operaterjev, pri katerih se podatki za namene klasifikacije ne naložijo v spomin stroja, temveč jih algoritem inkrementalno nalaga iz podatkovne baze. To dejstvo omogoča zelo velike podatkovne množice, ker je velikost le-teh omejena zgolj s prostorom na strežniku podatkovne baze. Z razširitvijo Radoopa obstaja možnost uporabe platforme Hadoop kot tehnologije za porazdeljeno shranjevanje in procesiranje podatkov. Hitrost računanja je v tem primeru odvisna od števila strežniških gruč. Tehnologija deluje počasneje na manjših podatkovnih množicah, vendar je skalabilnost takih sistemov dosti boljša. Zaradi velike skupne uporabe je priporočljivo, da gruče Hadoop uporabimo, kadar izvajamo operacije na več kot 50 milijonov podatkovnih točk [28].

Klasifikacija na podlagi podatkovnih množic, naloženih v spomin, je zaradi hitrosti spomina zelo hitra, vendar podpira relativno majhno število podatkovnih točk. Glavni vzrok je omejenost velikosti pomnilnika. Izračuni na podatkih v podatkovni bazi so v primerjavi veliko počasnejši, vendar je velikost podatkovne množice omejena samo s količino prostora v podatkovni bazi. Platforma Hadoop je zaradi velike skupne uporabe do 5 milijonov primerkov v podatkovni množici najpočasnejša od vseh možnosti, pri ekstremnih količinah podatkov pa je najučinkovitejši način obdelave podatkov. To lahko sklepamo iz grafa na sliki 3.1. Algoritem se je izvajal na treh gručah strežnikov Hadoop. [28].



Slika 3.1 Primerjava skalabilnosti glede na analitični pogon

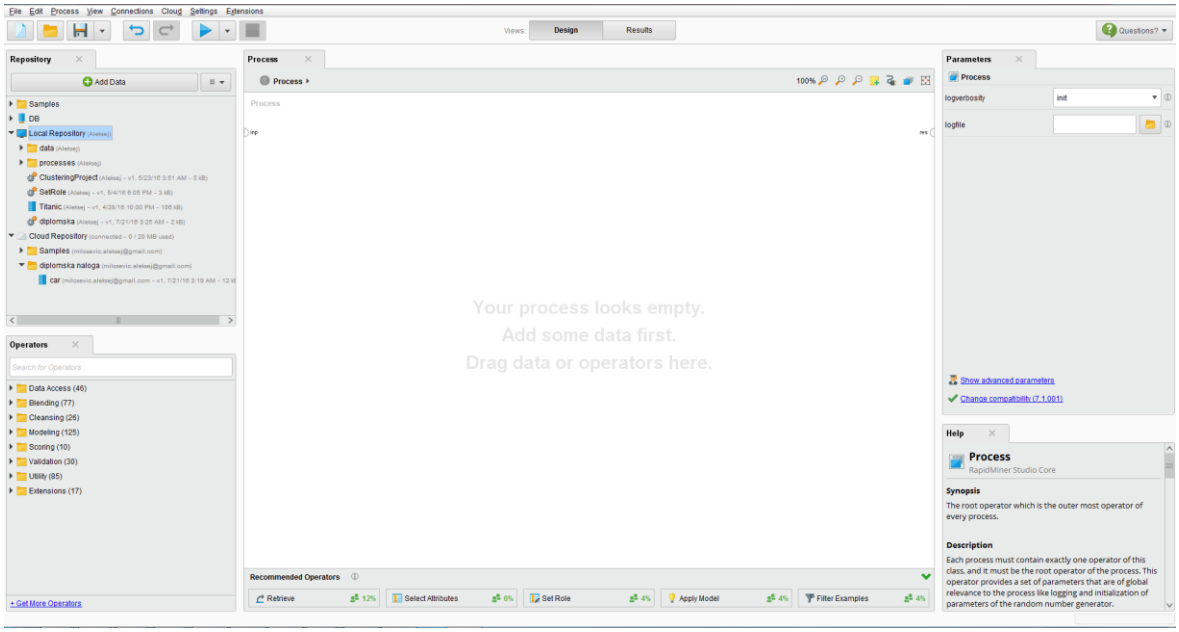
(Vir: https://rapidminer.com/wp-content/uploads/2013/07/RapidMiner.-runtime_engines_nb1.jpg)

3.2.2 Uporabniški vidik

V tem podpoglavju bomo opisali delovanje orodja RapidMiner Studio Community iz uporabniškega vidika. Uporabniku sta na voljo dva načina uporabe orodja: preko uporabniškega vmesnika ali preko uporabe RapidMinerjeve odprtokodne javanske knjižnice. Naprednim uporabnikom je na voljo tudi ukazna vrstica.

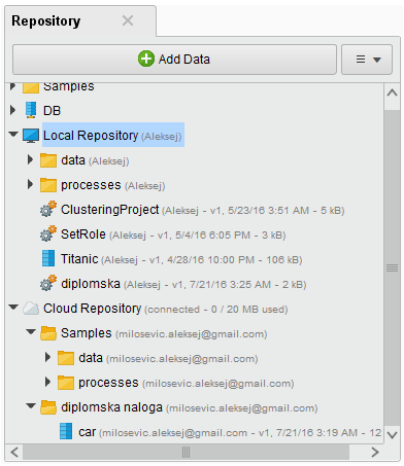
3.2.2.1 Grafični uporabniški vmesnik RapidMiner Studio

Grafični vmesnik je razdeljen na zbirko funkcionalnosti, imenovanih pogledi. Pogledi so skupina panelov, ki skupaj funkcionirajo kot zaključena celota. Lahko jih poljubno dodajamo, spreminjamo ali brišemo. Privzeta pogleda sta dva, in sicer pogled »Design«, prikazan na sliki 3.2, in pogled »Results«.



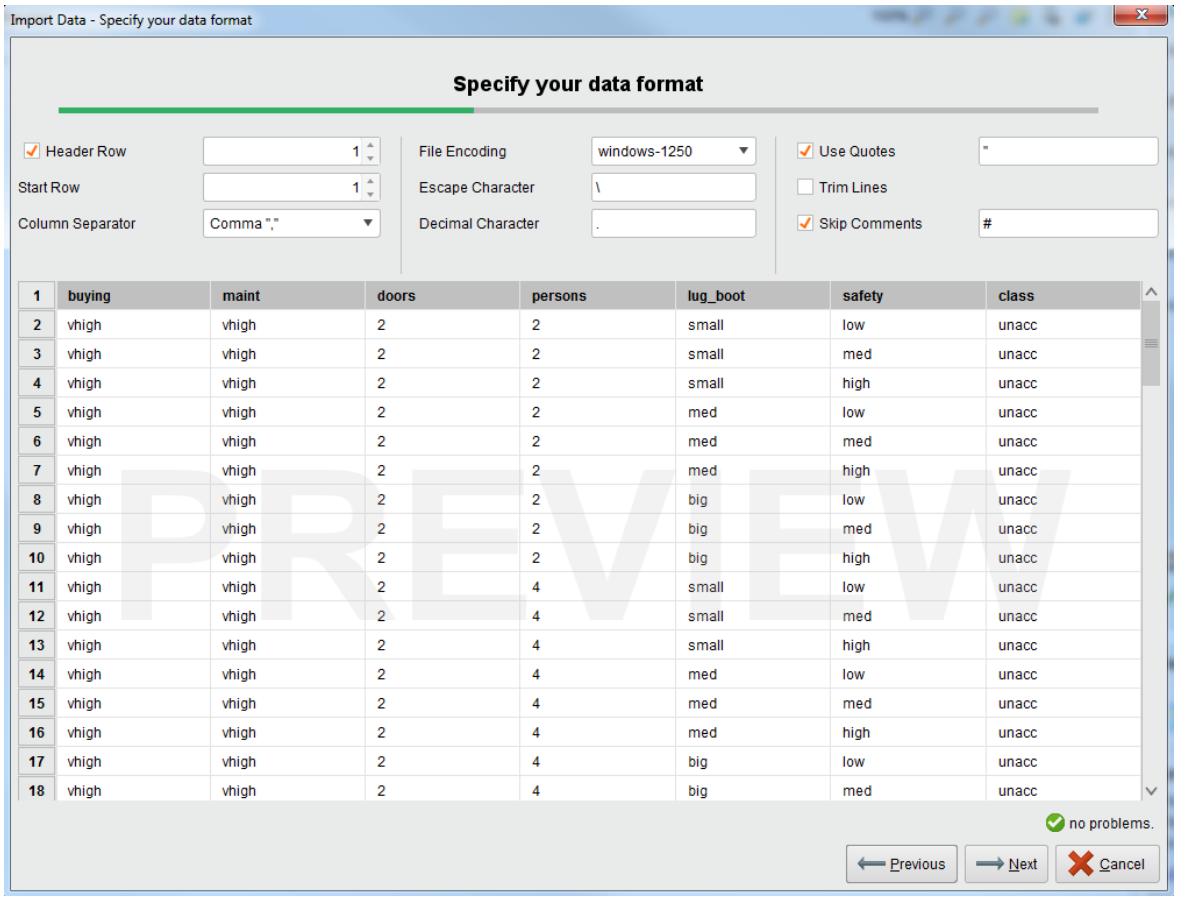
Slika 3.2 Prazen pogled »Design«

Pogled Design vsebuje naslednje panele: »Repository«, »Operators«, »Process«, »Parameters«, »Help« in »Wisdom of the Crowds«. Panel »Repository« prikazuje lokalni repozitorij in repozitorij RapidMiner Cloud. Vanj shranjujemo tako podatkovne množice kot celotne procese [23]. Prikaz primera videza panela »Repository« je na sliki 3.3.



Slika 3.3 Panel Repository

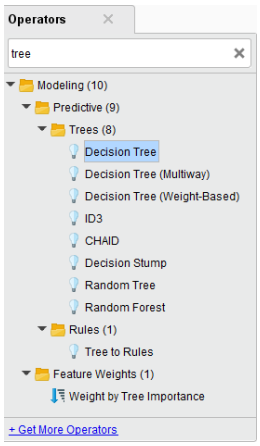
S klikom na gumb »Add Data« preko čarovnika uvozimo poljubno podatkovno množico. Ko izberemo datoteko, se nam odpre čarovnik, ki analizira podatke in nas opozori na nepravilnosti. Povpraša nas tudi po tipu kodiranega nabora znakov, separatorja podatkov in podobno. Na sliki 3.4 vidimo primer uvoza datoteke car.csv. Če je uvoz uspešen, se uvožena datoteka pojavi v repozitoriju. Če smo jo naložili na oblak, se pojavi v razdelku »Cloud Repository«, če pa na lokalnem trdem disku, pa »Local Repository«.



Slika 3.4 Uvoz datoteke car.csv

Operatorji

Pri gradnji procesov delamo skoraj izključno s komponentami, imenovanimi operatorji. Celotno množico operatorjev najdemo v panelu »Operators«, ki je prikazan na sliki 3.5. Za lažje iskanje imamo na voljo iskalno okence. Preko povezave »Get More Operators« dostopamo do menija, preko katerega lahko namestimo uporabniške razširitve, ki implementirajo operatorje raznolikih funkcionalnosti.

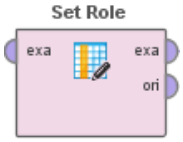


Slika 3.5 Panel »Operators«

Operator je sestavljen iz naslednjih komponent [24]:

- vhodi,
- izhodi,
- operacija, ki jo operator izvede na vhodnih podatkih,
- parametri, ki nadzirajo pogoje in specifike izvajanja operacije.

Vhodi in izhodi operatorjev so generirani ali uporabljeni preko vrat (angl. port), ki pričakuje določen tip vhoda [24]. Primer reprezentacije operatorja v uporabniškem vmesniku RapidMiner Studio vidimo na sliki 3.6.



Slika 3.6 Operator "Set Role"

Operatorji lahko prejmejo vhode različnih virov; neposredno iz podatkovne baze, podatkovne datoteke ali izhodnih vrat drugih operatorjev. Barva operatorjev nakazuje, kakšen tip vhoda sprejme [24]. V primeru na sliki 3.6 operator »Set Role« kot vhod prejme podatkovno množico, kar nakazujeta modrikasta barva vrat in pripis »exa« ali primer (angl. example), množici pripiše razred (v RapidMinerju označen kot »label«) in tako modificirano množico pošlje naslednjemu operatorju preko istoimenskih izhodnih vrat. Omenjeni operator ima možnost kot izhod vrniti še originalno, torej nespremenjeno, množico preko vrat »ori«.

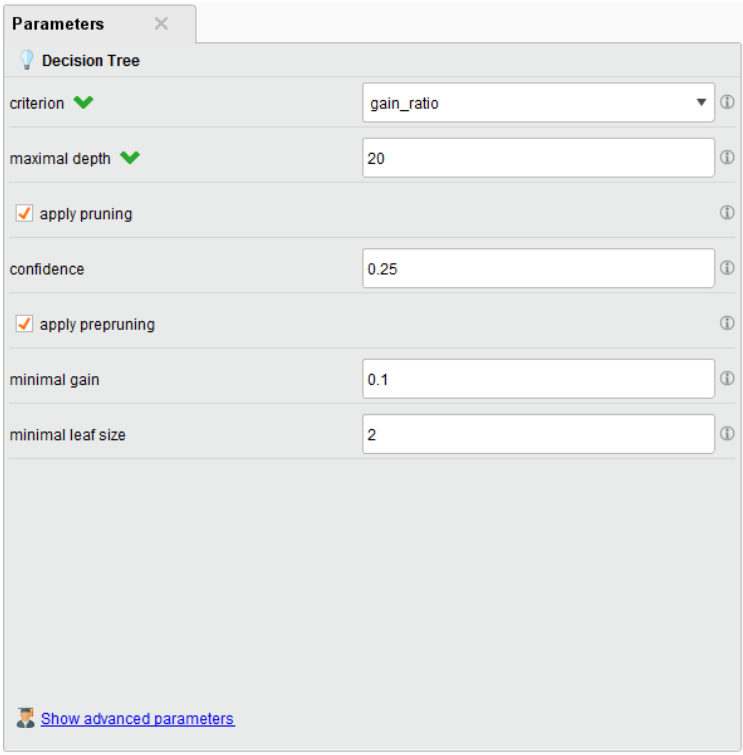
RapidMiner Studio ponuja močna orodja za vizualizacijo stanja in težav operatorjev glede na celoten proces. Orodje jasno označi vhodna in izhodna vrata, ki potrebujejo povezavo ali jim primanjkuje pomembna informacija o konfiguraciji [24]. Operator kot tak vizualizira status preko elementov, ilustriranih na sliki 3.7.

Slika 3.7 Možni statusi operatorjev

Od leve proti desne indikatorji pomenijo naslednje [24]:

- **Statusna luč**: Lahko je rdeče ali rumene barve. Rdeča barva pomeni, da je potrebna nastavitev parametrov ali pa da vrata, nujna za delovanje operatorja, nimajo povezave. Rumena barva pomeni, da je konfiguracija operatorja sicer pravilna, ampak da le-ta še ni popolnoma implementiran.
- **Opozorilni trikotnik**: Indikator za posebna statusna sporočila operatorja.
- **Prekinitvena točka**: Orodje uporabniku omogoča, da v namene razhroščevanja procesa posamezne operatorje nastavi kot prekinitvene točke. Odloči se lahko, ali se bo izvajanje procesa končalo pred ali po izvajanju obravnavanega operatorja.
- **Komentar**: Indikator, da je uporabnik operatorju dodal komentar.
- **Podproces**: Indikator, da operator vsebuje podproces. Z dvojnim klikom na kvadratek operatorja dostopamo do podrobnosti podprocesa.

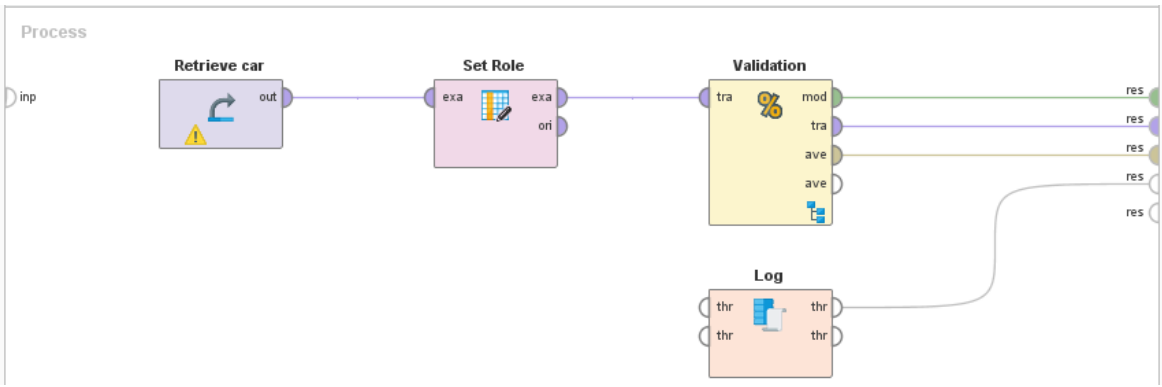
Preko panela »Parameters« dostopamo do nastavitev parametrov izbranega operatorja. Nabor parametrov je odvisen od tipa operatorja [24]. Na sliki 3.8 vidimo osnovne parametre operatorja »Decision Tree«. S klikom na gumb »Show advanced parameters« se nam pokažejo dodatne možnosti, ki so namenjene naprednim uporabnikom. Zelene kljukice poleg nekaterih možnosti povedo, da večina uporabnikov orodja RapidMiner Studio uporablja izbrano nastavitev.



Slika 3.8 Panel Parameters operatorja Decision Tree

Gradnja in validacija modela

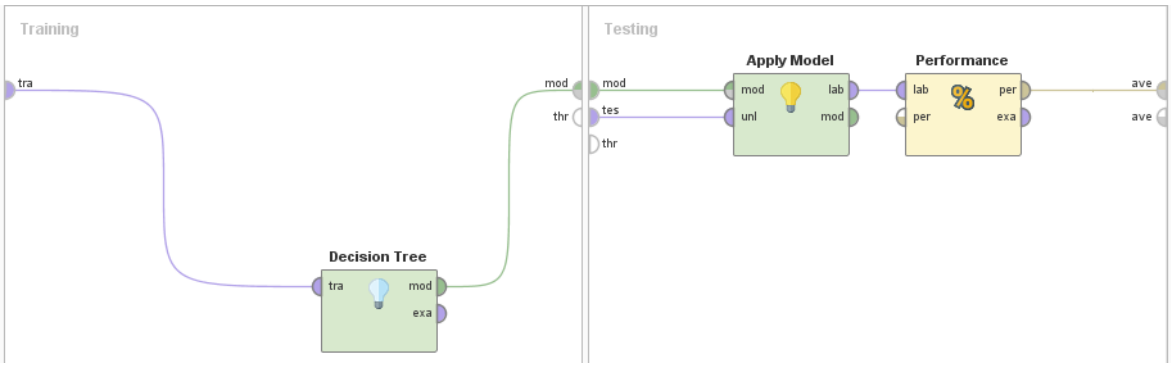
Procesi, kot jih definira RapidMiner v uporabniškem priročniku [19], so med seboj povezani operatorji, ki skupaj tvorijo neki rezultat. Vsak proces se v ozadju pretvori v dokument xml, kar omogoča enostavno prenosljivost in vgrajevanje procesa v lastne programske rešitve preko Rapidminer API [18]. Kot sem omenil v uvodu podpoglavja, si bomo ogledali primer implementacije procesa klasifikacije z odločitvenim drevesom, kot je prikazan na sliki 3.9.



Slika 3.9 Implementacija in validacija metode klasifikacije z odločitvenim drevesom

Celoten proces sestoji iz štirih operaterjev: »retrieve car«, »set role«, »validation« in »log«. Trije izmed štirih operatorjev so preprosti, operater validacije pa ima podprocese, kot to nakazuje ikonica v spodnjem delu kvadratka. Operatorji od leve proti desni so [24]:

- **Retrieve car**: V repozitoriju poišče in vrne zahtevano podatkovno množico. V našem primeru je to prej uvoženi car.csv. Vhodnih vrat operater nima, preko izhodnih vrat »out« pa posreduje operaterju »Set Role« podatkovno množico. Edini parameter operaterja je pot do množice v repozitoriju.
- **Set role**: Vhodni podatkovni množici glede na nastavitve parametrov predpiše razred. Operator ima dvoje izhodnih vrat. Vrata »exa« primernemu operatorju validation pošljejo rezultat operacije, opcijska vrata »ori« pa originalno nespremenjeno množico.
- **Validation**: Je sestavljen operator, ki vsebuje dva podprocesa: »Training« in »Testing«. Podproces »Training« vsebuje operatorje, ki izvajajo operacije na učni množici. V našem primeru je to operator »Decision Tree«, ki kot vhod prejme učno množico, kot izhod pa vrne model, ki ga je algoritem ustvaril na podlagi parametrov operatorja. Sistem izhodni model posreduje procesu »Testing« preko vhodno/izhodnih vrat »mod«. Proces »Testing« z operatorjem »Apply Model« model uporabi na testni množici, z operaterjem »Performance« pa meri uspešnost učenja. Končni rezultat posameznega testiranja je raven uspeha validacije. Podprocesa operatorja sta prikazana na sliki 3.10. Validacija ima tri različne tipe izhodnih vrat. »Mod« posreduje model, uporabljen pri validaciji, »tra« posreduje učno množico in »ave« povprečje vseh iteracij validacij. RapidMiner s pomočje omenjenih treh elementov prikaže rezultate procesa. Tip vzorčenja in število validacij določimo preko parametrov operaterja.



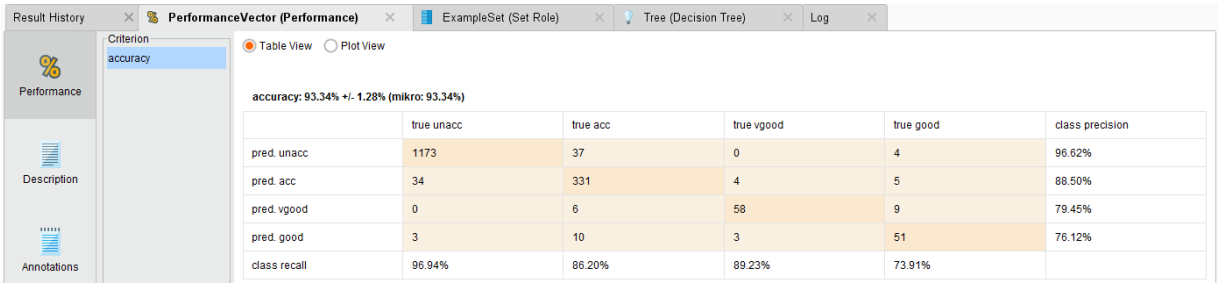
Slika 3.10 Podprocesa »Training« in »Testing«

- **Log**: Je opcijski operator, ki zbira podatke o izvajanju procesa. Preko parametrov lahko uporabnik med drugim spreminja tip sortiranja podatkov (*bottom-k* in *top-k*) ter navede ime in lokacijo shranjevanja datoteke.

S klikom na gumb z ikono modre puščice se proces izvede (slika 3.2).

Rezultati

Rezultate procesa najdemo v pogledu »Results«, ki je drugi izmed dveh privzetih pogledov. Sestavljen je iz dveh delov. Privzeto je na levi strani prostor za prikaz rezultatov, na desni pa panel »Repository«. Prikaz repozitorija v tem pogledu je praktično enak kot pri pogledu »Design«, zato se osredotočimo na delovanje prikaza rezultatov. Rezultati analize so odvisni od nastavitev posameznih operatorjev in priključkov v vrata »res«. Vsak priključek predstavlja en zavihek, kar dobro demonstrira slika 3.11.



	true unacc	true acc	true vgood	true good	class precision
pred. unacc	1173	37	0	4	96.62%
pred. acc	34	331	4	5	88.50%
pred. vgood	0	6	58	9	79.45%
pred. good	3	10	3	51	76.12%
class recall	96.94%	86.20%	89.23%	73.91%	

Slika 3.11 Rezultati procesa

V našem primeru imamo pet zavihkov:

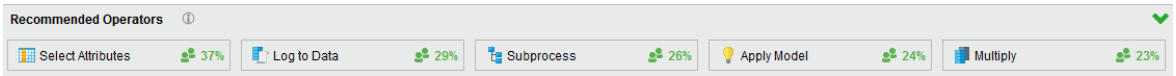
- **Result History**: Zavihek »Result History« je privzet in se pojavi vedno. Prikazuje strnjen povzetek vseh izvedb procesa, organiziran glede na posamezne zavihke.
- **PerformanceVector:** Je rezultat operatorja »Performance«. V našem primeru prikazuje le točnost in matriko zmede, vendar pa lahko v parametrih operatorja vklopimo več kot 15 različnih meril.
- **ExampleSet**: Vsebuje rezultate operaterja »SetRole«. V praksi to pomeni, da zavihek prikaže tabelo z vnosi podatkovne množice. Razred, kot ga je definiral uporabnik s pomočjo operatorja, je obarvan z zeleno.

- **Tree**: Vsebuje prikaz modeliranega drevesa na človeku intuitiven način. Možnih je več načinov prikaza drevesa.
- **Log**: Vsebuje grafe in tabele o metapodatkih izvajanja procesa. Vodi podatke, kot so skupni čas izvajanja, čas izvajanja posamezne iteracije in raba procesorja.

Pomoč uporabnikom znotraj grafičnega vmesnika

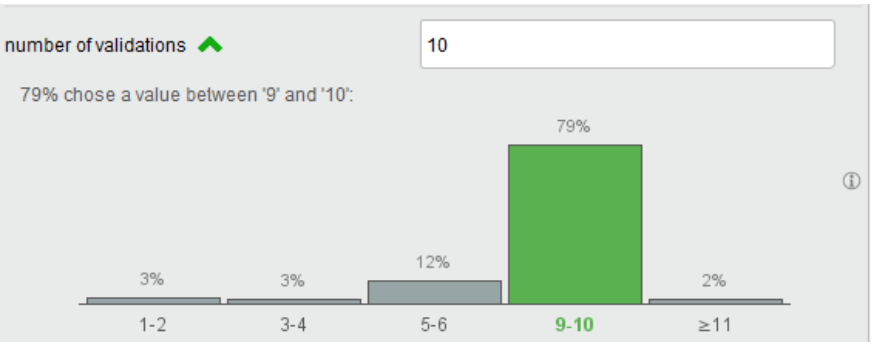
Ker podjetje RapidMiner svoje rešitve podatkovnega rudarjenja razvija tudi za netehnične profile, veliko sredstev vlaga v vizualno delo. Lahko bi rekli, da je način grajenja procesov pravzaprav neke vrste bločno programiranje. Za dobro začetno izkušnjo uporabnika skrbi veliko elementov pomoči znotraj samega orodja. Začetniku so znotraj same aplikacije na voljo vaje, ki ga vodijo skozi osnove uporabe vseh aspektov orodja od predprocesiranja podatkovne množice do grajenja procesov in interpretacije rezultatov [19].

Ena izmed najpomembnejših in najunikatnejših oblik pomoči v RapidMinerju je prav gotovo sistem »Wisdom of the Crowds«. Z anonimno analizo uporabe orodja so razvijalci ugotovili, kateri operatorji se največkrat uporabljajo v katerem kontekstu in s kakšnim naborom nastavitev parametrov. To znanje so implementirali v uporabniški vmesnik na diskreten in enostavno razumljiv način. Na ravni procesa je »Wisdom of the Crowds« implementiran v obliki panela »Recommended Operators« (slika 3.12), ki na podlagi trenutnega nabora operatorjev in nastavitev njihovih parametrov predlaga operatorje, ki so jih drugi uporabniki v tem kontekstu najpogosteje uporabili [29].



Slika 3.12 Panel »Recommended Operators«

Na ravni parametrov operatorja je »Wisdom of the Crowds« implementiran v obliki informacij o pogostosti uporabe nastavitev najpomembnejših parametrov. Slika 3.13 prikazuje priporočilo izbire števila validacij za operator »Validation«.



Slika 3.13 Priporočilo »Wisdom of the Crowds« za število validacij

Poleg omenjenega je v RapidMiner Studiu pomemben panel »Help«, ki za izbrani operator poda najpomembnejše informacije za uspešno uporabo operatorja v procesu. Za vsak operator je na voljo razdelek, ki vsebuje splošen opis, seznam in razlago vhodnih in izhodnih tipov ter seznam in razlago nastavitev parametrov. Za kompleksnejše operatorje pridejo v poštev razdelki, kot so kratki vodič uporabe in praktični primeri.

3.2.2.2 Implementacija procesa v lastno rešitev z uporabo RapidMiner API

Platformo RapidMiner lahko v lastno rešitev z jezikom Java implementiramo na dva načina; z nalaganjem procesa iz datoteke xml in z dinamičnim veriženjem operatorjev. Preden lahko začnemo z dejanskim grajenjem procesov, moramo orodje inicializirati. Postopek inicializacije je odvisen od tega, v kakšnem okolju se bo RapidMiner izvajal. Procedura inicializacije preko ukazne vrstice je prikazana na sliki 3.14. RapidMiner definira naslednje enumeratorje, namenjene definiciji tipa izvajanja [30]:

- APPLET: Izvajanje kot applet znotraj brskalnika
- APPSERVER: Izvajanje znotraj aplikacijskega strežnika
- COMMAND_LINE: Izvajanje preko funkcije *RapidminerCommandLine.main(String[])*
- TEST: Izvajanje za namene testiranja
- EMBEDDED_AS_APPLET: RapidMiner je vgrajen kot applet
- EMBEDDED_WITH_UI: RapidMiner je vgrajen znotraj drugega programa z uporabniškim vmesnikom
- EMBEDDED_WITHOUT_UI: RapidMiner je vgrajen znotraj drugega programa brez uporabniškega vmesnika

- UI: Izvajanje preko metode *RapidminerGUI.main(String[])*
- WEBSTART: Izvajanje znotraj tehnologije Java Web Start
- UNKNOWN: Način zagona ni znan

```
//Inicializacija knjižnice RapidMiner
RapidMiner.setExecutionMode(RapidMiner.ExecutionMode.COMMAND_LINE);
RapidMiner.init();
```

Slika 3.14 Inicializacija preko ukazne vrstice

Metoda nalaganja procesa iz datoteke xml

Kot smo že omenili, je vsak proces RapidMiner shranjen v obliki dokumenta xml, v katerem so shranjeni vsi podatki o procesu. To vključuje tudi uporabljeno podatkovno množico in posamezne nastavitve parametrov operatorjev. Za uspešno izvedbo algoritma, prikazanega na sliki 3.15, moramo navesti tip inicializacije in v objekt razreda Process navesti lokacijo. Ob klicu *process.run()* se celotni proces izvede. Rezultate izpišemo v tekstovni obliki v konzolo. Metoda nalaganja procesov iz datoteke xml je priporočeni način integracije platforme RapidMiner v lastno rešitev [31].

```
public static void xmlCrossValidation() {
    //Inicializacija knjižnice RapidMiner
    RapidMiner.setExecutionMode(RapidMiner.ExecutionMode.COMMAND_LINE);
    RapidMiner.init();
    try {
        Process process = new Process(new File("E:/Program
Files/Dropbox/ITK/6.Semester/Diplomska naloga/validationxml.xml"));
        IOContainer ioResult = process.run();
        for (int i = 0; i < ioResult.size(); i++) {
            System.out.println(ioResult.getElementAt(i));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Slika 3.15 Algoritem izvajanja navzkrižne validacije z uporabo datoteke xml

Metoda dinamičnega veriženja operatorjev

Če želimo proces graditi dinamično, ga moramo v celoti zgraditi v jeziku Java. RapidMiner tega načina gradnje procesov ne priporoča, posledično je za metodo na voljo relativno malo

po večini že zastarele dokumentacije [31]. Operatorje ustvarjamo in verižimo na podoben način kot v uporabniškem vmesniku. Parametre operatorjev spreminjamo preko za to namenjenih metod.

Algoritem začnemo z ustvarjanjem procesa, ki je odgovoren za shranjevanje podatkov o vseh operatorjih, njihovih parametrih in podprocesih. Nadaljujemo z instanciranjem in nastavitvijo vrednosti parametrov operatorjev »csv_reader« (slika 3.16), »Set Role« (slika 3.17) in »Validation« (slika 3.18).

```
//Inicializacija operatorja csv_reader
Operator csv reader =
OperatorService.createOperator(CSVExampleSource.class);
csv_reader.setParameter(CSVExampleSource.PARAMETER_CSV_FILE, "E:/Program
Files/Dropbox/ITK/6.Semester/Diplomska naloga/car.csv");
csv_reader.setParameter(CSVExampleSource.PARAMETER_COLUMN_SEPARATORS, ",");
glavniProces.getRootOperator().getSubprocess(0).addOperator(csv_reader);
```

Slika 3.16 Instanciranje in dodajanje operatorja »csv_reader« v proces

```
//Inicializacija operatorja SetRole, ki pridobljeni podatkovni množici
pripiše razred
ChangeAttributeRole nastaviRazred =
OperatorService.createOperator(ChangeAttributeRole.class);
nastaviRazred.setParameter(ChangeAttributeRole.PARAMETER_NAME, "class");
nastaviRazred.setParameter(ChangeAttributeRole.PARAMETER_TARGET_ROLE,
Attributes.LABEL_NAME);
glavniProces.getRootOperator().getSubprocess(0).addOperator(nastaviRazred);
```

Slika 3.17 Instanciranje in dodajanje operatorja »SetRole« v proces

```
//Inicializacija operatorja Validation
XValidation xvalidation =
OperatorService.createOperator(XValidation.class);
xvalidation.setParameter(XValidation.PARAMETER_NUMBER_OF_VALIDATIONS,
"10");
xvalidation.setParameter(XValidation.PARAMETER_SAMPLING_TYPE,
"stratified");
```

Slika 3.18 Instanciranje in dodajanje operatorja »Validation« v proces

Seveda zgolj dodajanje operatorjev v proces ni dovolj, temveč jih moramo med seboj povezati. Do vrat posameznih operatorjev dostopamo z metodama *getOutputPorts( ) in getInputPorts( ),* s katerima povežemo neki operator z drugim. Na sliki 3.19 je prikazana izvorna koda, ki tri prej omenjene operatorje med seboj poveže.

```
//Operatorje, ki so za izvedbo procesa potrebni, med seboj povežemo
csv_reader.getOutputPorts().getPortByName("output").connectTo(
        nastaviRazred.getInputPorts().getPortByName("example set input"));
nastaviRazred.getOutputPorts().getPortByName("example set
output").connectTo(
        xvalidation.getInputPorts().getPortByName("training"));
xvalidation.getOutputPorts().getPortByName("averagable 1").connectTo(

glavniProces.getRootOperator().getSubprocess(0).getInnerSinks().getPortByIn
dex(0));
```

Slika 3.19 Povezava definiranih operatorjev

Ker je operator »Validation« sestavljen iz dveh podprocesov (»Training« in »Testing«), moramo operatorje podprocesov prav tako instacirati, jim nastaviti parametre in jih dodati v proces. Izvorna koda za učni del validacije (*getSubProcess(0)*) je na sliki 3.20, za testni del (*getSubProcess(1)*) pa na sliki 3.21.

```
//Definiramo razred klasifikatorja in povežemo operatorje prvega podprocesa
Operator klasifikator = OperatorService.createOperator(NaiveBayes.class);
xvalidation.getSubprocess(0).addOperator(klasifikator); //
xvalidation.getSubprocess(0).getInnerSources().getPortByName("training").co
nnectTo(
        klasifikator.getInputPorts().getPortByName("training set"));
klasifikator.getOutputPorts().getPortByName("model").connectTo(

(xvalidation).getSubprocess(0).getInnerSinks().getPortByName("model"));
```

Slika 3.20 Učni del validacije

```
//Definiramo operatorje drugega podprocesa "Testing"
Operator applyModel = OperatorService.createOperator(ModelApplier.class);
xvalidation.getSubprocess(1).addOperator(applyModel);
Operator perf =
OperatorService.createOperator(PolynominalClassificationPerformanceEvaluato
r.class);
xvalidation.getSubprocess(1).addOperator(perf);
// Povežemo operatorje podprocesa "Testing"
xvalidation.getSubprocess(1).getInnerSources().getPortByName("model").conne
ctTo(
        applyModel.getInputPorts().getPortByName("model"));
xvalidation.getSubprocess(1).getInnerSources().getPortByName("test
set").connectTo(
        applyModel.getInputPorts().getPortByName("unlabelled data"));
applyModel.getOutputPorts().getPortByName("labelled data").connectTo(
        perf.getInputPorts().getPortByName("labelled data"));
perf.getOutputPorts().getPortByName("performance").connectTo(

xvalidation.getSubprocess(1).getInnerSinks().getPortByName("averagable
1"));
```

Slika 3.21 Testni del validacije

3.2.3 Razširljivost

RapidMiner uporabniku omogoča, da razvija lastne rešitve in jih implementira v platformo v obliki razširitev (angl. extension). Te lahko objavi na spletnem tržišču RapidMiner Marketplace. Po funkcionalnosti jih delimo na naslednje tipe [32]:

- **Podatkovni viri in formati**: Implementacija novih podatkovnih virov vključno z razčlenjevalniki, generatorji in pretvorniki podatkovnih formatov.
- **Domensko-specifični operatorji**: Algoritmi in operatorji, razviti za uporabo v specifičnih aplikacijskih domenah in primerih uporabe.
- **Strojno učenje**: Algoritmi strojnega učenja, ki niso vključeni v jedro platforme.
- **Operatorji**: Sem sodijo operatorji, ki ne sodijo v druge kategorije.
- **Učno gradivo**: Gradivo, namenjeno izobraževanju, vključno z vodiči.
- **Knjižnice**: Adapterji za knjižnice drugih platform (Weka, R).
- **Razvojne**: Pripomočki, ki pomagajo razvijalcem pri gradnji novih priključkov.

Nekateri priključki uživajo uradno podporo razvijalcev. Posledično so redno posodabljani, uporabniki s profesionalnimi licencami pa imajo na voljo še tehnično podporo.

3.3 Weka

Weka je ena izmed najstarejših še aktualnih platform za podatkovno rudarjenje. Razvijajo jo raziskovalci Univerze v Waikatu, Nova Zelandija, in sicer v jeziku Java. Delo na platformi opravljamo preko uporabniškega vmesnika ali programiranja lastnih rešitev s pomočjo Javanskih knjižnic. Zaščitena je pod odprtokodno licenco GNU General Public Licence. Za namene diplomskega dela smo uporabili Weko 64-bitne različice 3.8.0. Ker je platforma napisana v programskem jeziku Java, jo lahko poganjamo v vseh modernih okoljih; Windows, Mac OSX in Linux.

3.3.1 Vidik podatkovnih virov in funkcionalnosti

Predprocesiranje podatkov

Za uvoz podatkovnih množic v Weko lahko uporabimo vrsto nepovezanih formatov in vse podatkovne baze, ki nudijo gonilnike JDBC (Java Database Connector). Podprti nepovezani podatkovni formati so [33]: Arff, C4.5, CSV, JSON, libsvm, Matlab, Binary serialized instances in XRFF.

Posebej velja omeniti format Arff, ki so ga razvili znanstveniki na univerzi v Waikatu specifično za platformo Weka. Format nastopa v obliki tekstovnega dokumenta, ki vsebuje seznam primerkov, ki jim je skupen neki nabor atributov. Atributi so definirani v glavi, podatki pa v telesu dokumenta.

Z namenom boljšega razumevanja podatkovne množice za vsak atribut program generira statistične podatke, kot na primer mere srednjih vrednosti, mere variance, maksimume in minimume. V primeru uporabe v uporabniškem vmesniku podatke tudi vizualizira. Uporabniku orodje omogoča uporabo filtrov za transformacijo podatkov, kot so diskretizacija, vzorčenje, agregacija, določanje obravnave manjkajočih vrednosti in številni drugi [33].

Integracija različnih tehnik

Wekina dokumentacija definira in opredeli podporo trem splošnem tipom metod podatkovnega rudarjenja [34]:

- Klasifikacija
    - Bayes
    - Funkcije
    - Odločitvena drevesa
    - Leni algoritmi
    - Pravila
    - Meta
- Gručenje
- Asociacijska pravila

Weka za vsako izmed navedenih metod ponuja praktično vse najbolj znane algoritme. Za klasifikacijo in gručenje so na voljo standardne metode validacije, kot so navzkrižna validacija, validacija »vrni enega« in validacija na izbrani predpripravljeni množici. Tako pridobljene modele lahko poljubno shranjujemo, prenašamo in nalagamo. Napredni uporabniki lahko pišejo in na spletu objavljajo filtre in algoritme, ki Wekine osnovne zmogljivosti razširijo [33].

Skalabilnost

Napogostejši način ustvarjanja modela, ki ga Weka ponuja, je, da celotno množico podatkov naloži v spomin računalnika, kar pomeni, da smo pri obdelovanju velikih podatkovnih množic omejeni z velikostjo pomnilnika. To še posebej velja za uporabniški vmesnik »Explorer«, saj dodatni prostor v spominu zasedejo še orodja za vizualizacijo. Optimalen način generiranja modela na velikih podatkih je uporaba ukazne vrstice, bločno programiranje s pomočjo vmesnika »Knowledge Flow« ali neposredna implementacije v Javi. Z omenjenimi načini lahko uporabimo klasifikacijske tehnike, ki podatkovno množico v spomin nalagajo inkrementalno, na primer s tehniko naključnega vzorčenja z rezervoarjem (angl. Reservoir Sampling) [35].

Od različice Weka 3.7 naprej je v določeni meri zmožnost porazdeljenega procesiranja preko platform Hadoop [36] in Spark [37]. Za uspešno delo je potrebna dodatna distribucija, ki omogoči platformno-neodvisne metode baziranja in reduciranja podatkovnih množic ter distribucija za vsako platformo, s katero nameravamo delo opravljati. Naloga platformno-specifične distribucije je prilaganje platformno-neodvisnih metod na določeno platformo.

3.3.2 Uporabniški vidik

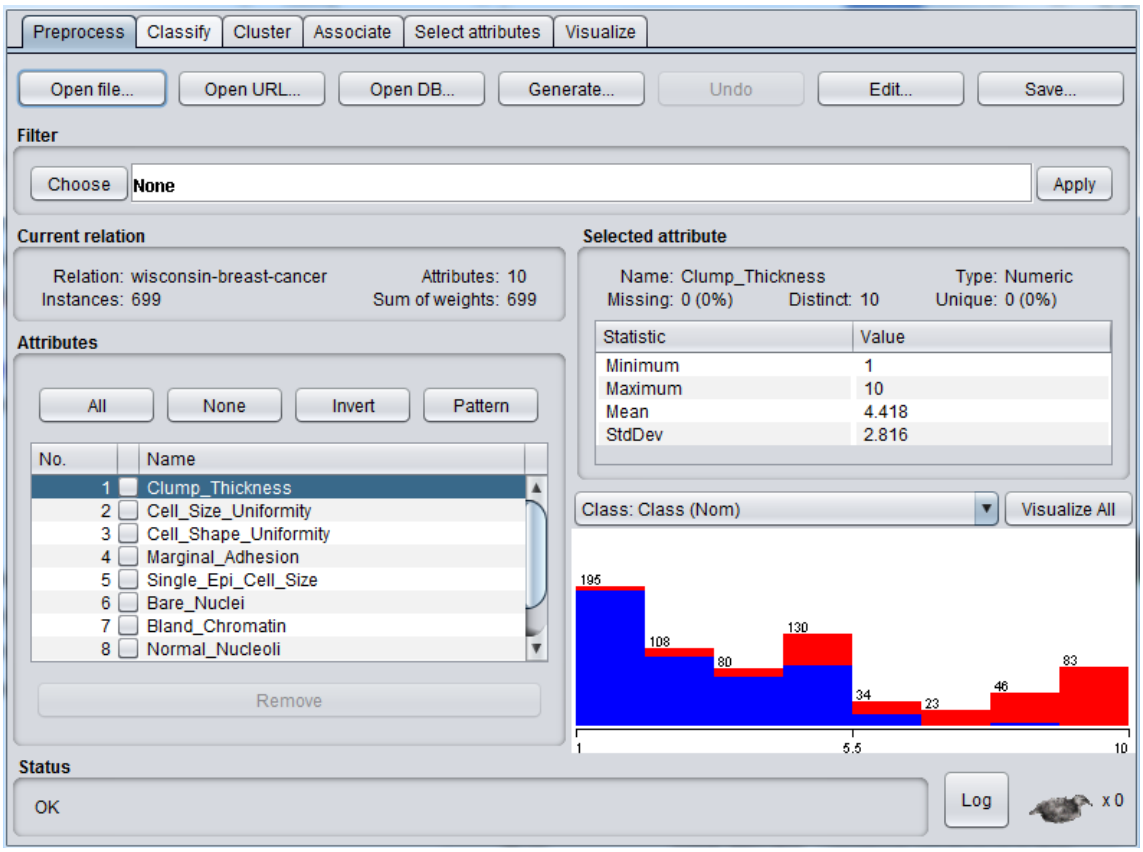
Platforma Weka omogoča izvajanje strojnega učenja preko uporabniškega vmesnika ali s pisanjem Javanske kode s pomočjo za to namenjenih knjižnic. V naslednjem poglavju bomo predstavili aspekte uporabniškega vmesnika in razložili implementacijo validacije modela, pridobljenega z algoritmom J48 v programskem jeziku Java.

3.3.2.1 Grafični uporabniški vmesnik

Wekin uporabniški vmesnik sestavlja pet aplikacij, ki skupaj predstavlja polno funkcionalnost platforme. Imena aplikacij so »Explorer«, »Experimenter«, »KnowledgeFlow«, »Workbench« in »Simple CLI«. V aplikaciji »Explorer« so na voljo samo algoritmi, ki podatkovne množice v spomin naložijo v celoti.

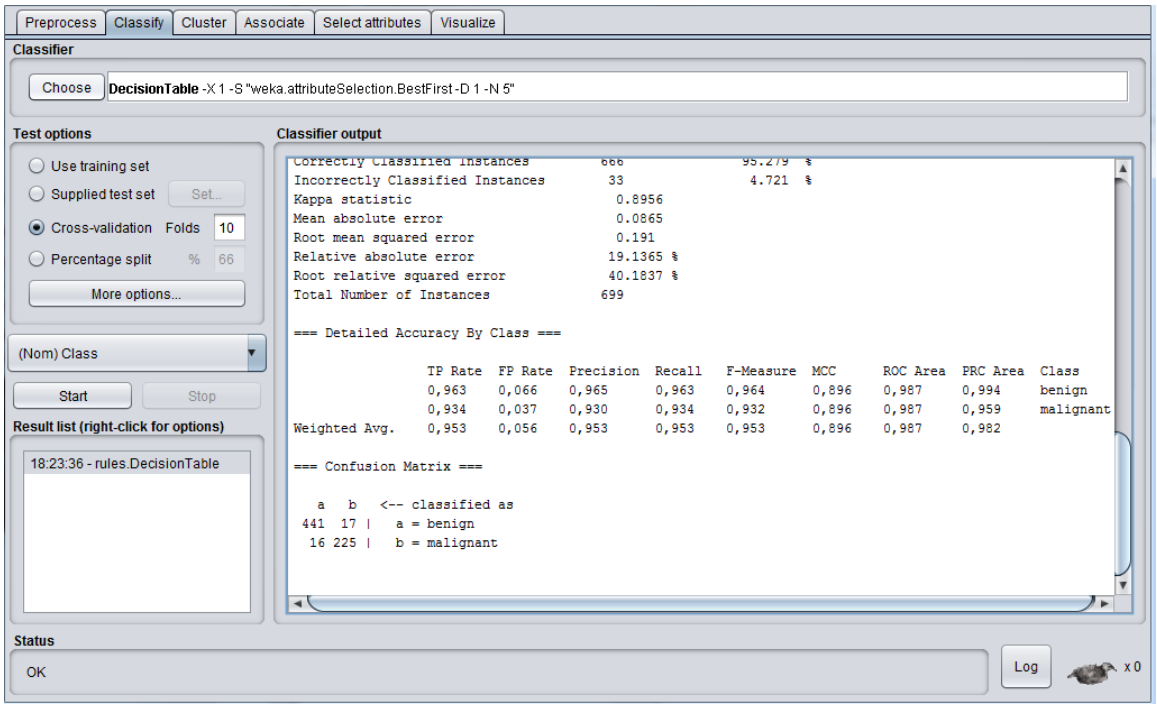
Explorer

Kot že ime pove, je »Explorer« orodje za spoznavanje in raziskovanje podatkovnih množic. Vmesnik ponuja možnost predprocesiranja, klasifikacije, gručenja, asociacije in izbora atributov. Zavihek »preprocessing« (slika 3.22) omogoča nalaganje podatkovnih množic iz dokumentov na lokalnem stroju, s spleta ali iz podatkovne baze. Splošni podatki o atributih naložene množice se pojavijo pod razdelkom »Current relation«, v razdelku »Attributes« pa nam je na voljo seznam vseh atributov, preko katerega lahko izberemo atribute, ki nas posebej zanimajo. Orodje omogoča vpogled v osnovne statistične podatke izbranega atributa pod razdelkom »Selected Attributes«. V razdelku »Filters« najdemo filtre, ki jih lahko uporabimo za transformacijo uvožene množice [33].



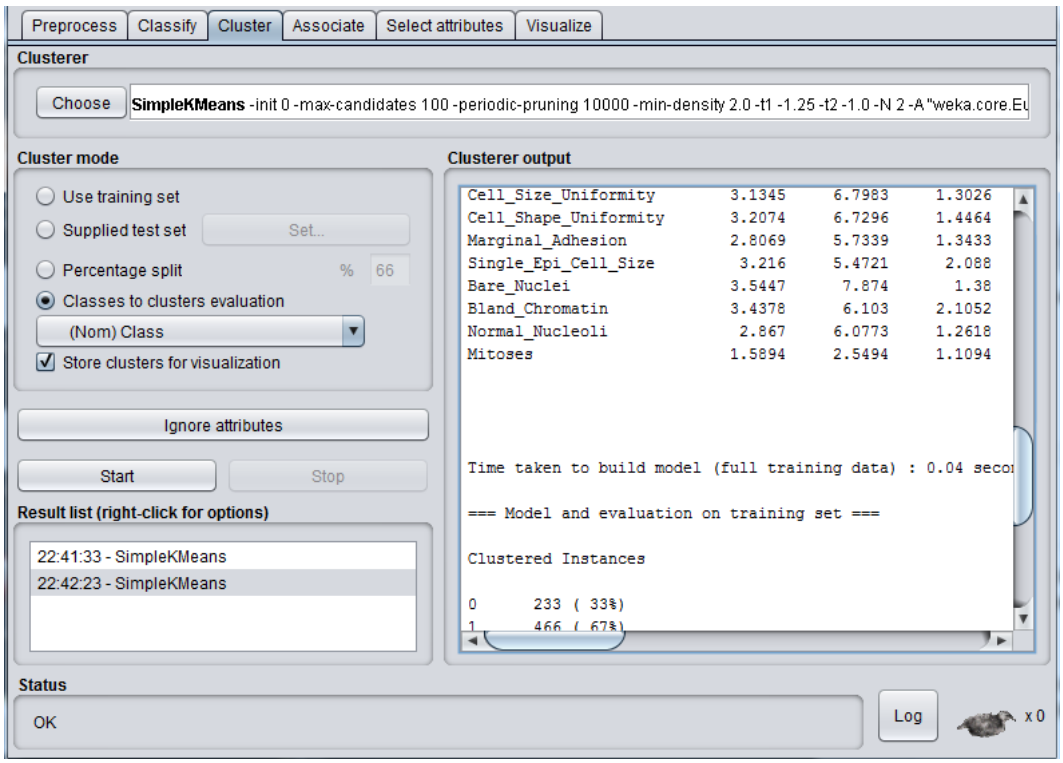
Slika 3.22 Zavihek "Preprocessing"

Preko zavihka »Classify« (slika 3.23) dostopamo do nastavitev gradnje klasifikatorjev. Preko spustnega menija lahko izbiramo med klasifikacijskimi in meta-klasifikacijskimi algoritmi. V razdelku »Test options« izberemo način validacije in izberemo atribut, ki bo prevzel vlogo razreda. Za boljše razumevanje rezultatov dostopamo do vizualnih orodij tako, da z desnim klikom izberemo proces iz razdelka »Results List« in iz spustnega menija izberemo enega izmed možnosti vizualizacije.



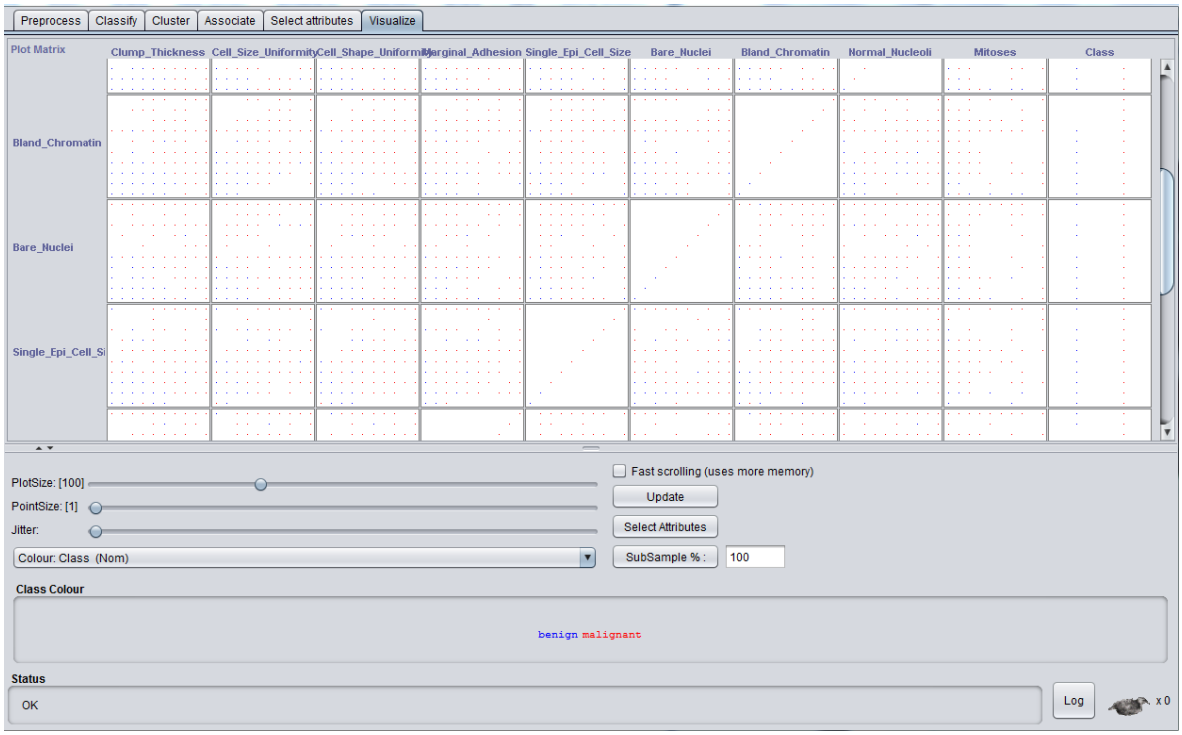
Slika 3.23 Zavihek "Classify"

Z zavihkom »Cluster« (slika 3.24) dostopamo do nastavitev gručenja. Podobno kot pri klasifikaciji izberemo želeni algoritem in testne možnosti. S klikom na gumb »Start« proces začnemo. Do možnosti vizualizacije dostopamo podobno kot pri zavihku klasifikacije.



Slika 3.24 Zavihek "Cluster"

Z možnostmi v zavihku »Associate« ugotavljamo povezave med atributi podatkovne množice. V primeru, da je naložena množica primerna za izvajanje algoritma, lahko pritisnemo gumb »Start«. Rezultati se prikažejo v okencu »Output«. Če želimo izvedeti, kateri atributi najbolj vplivajo na izvedene napovedi, uporabimo možnosti pod zavihkom »Select Attributes«. To storimo tako, da izberemo merila ocenjevanja atributov, tip validacije in razredni atribut. Podobno kot v prejšnjih zavihkih se nam rezultati izpišejo pod razdelkom »Attribute selection output«, do vizualizacijskih možnosti pa dostopamo preko seznama izvajanj procesa. Zavihek »Visualize« je namenjen vizualizaciji odnosa med posameznimi atributi. Na voljo je veliko prikaznih možnosti, kot so spreminjanje velikosti podatkovnih točk, spreminjanje velikosti grafa, prikaz odstopanj in tako naprej. Primer vizualizacije odnosov med atributi vidimo na sliki 3.25 [33].

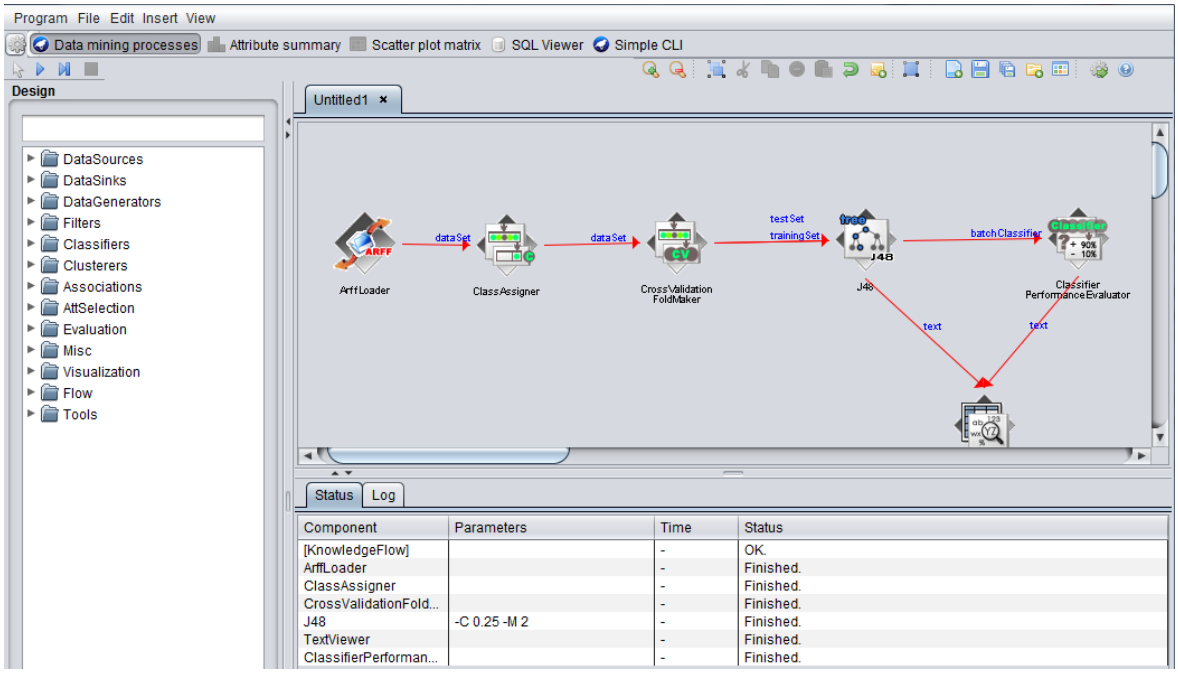


Slika 3.25 Zavihek "Visualize"

KnowledgeFlow

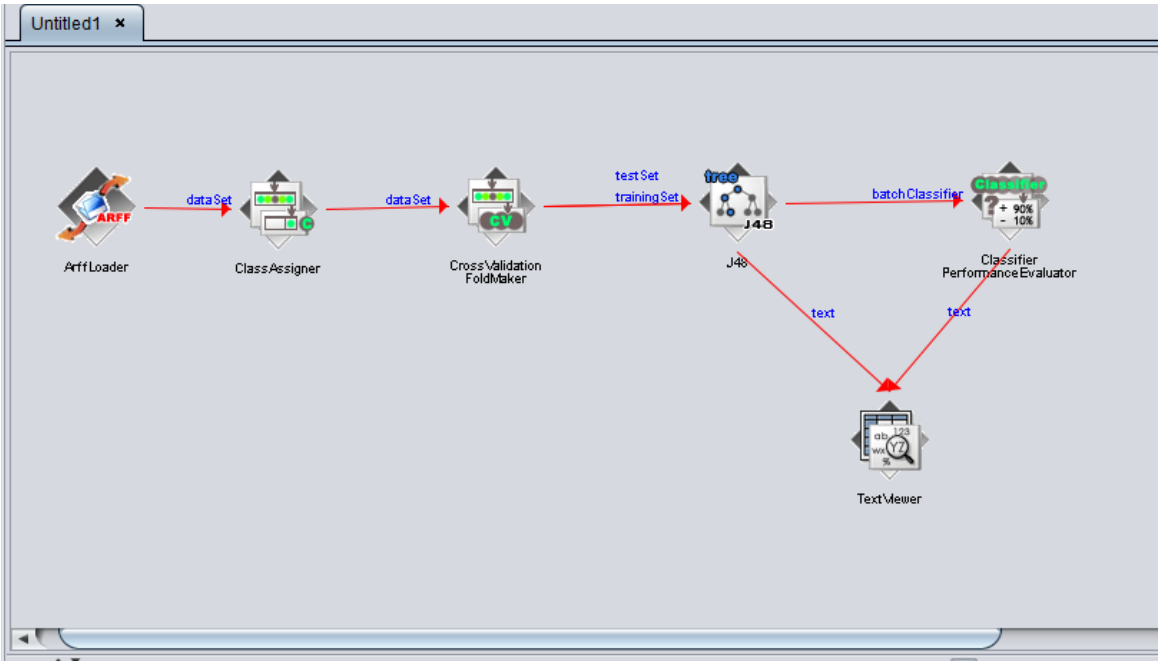
Knowledge Flow, prikazan na sliki 3.26, je alternativa vmesniku »Explorer«. Izvajane procese predstavlja kot tok podatkov, ki jih uravnavajo povezane komponente. Le-te na način »vleci in spusti« postavimo iz menija v delovno okolje. Z desnim klikom na

komponento dostopamo do nastavitve komponente. Nastavitve so odvisne od tipa komponent. V splošnem so na voljo možnosti za izbris, preimenovanje, spremembe konfiguracije delovanja in kopiranje. Komponente med seboj povežemo tako, da v nastavitvah pod menijem »Connections« izberemo tip izhoda in z levim klikom na primerno komponento tako usmerjamo tok [33].



Slika 3.26 Aplikacija "KnowledgeFlow"

Vzemimo primer klasifikacije z odločitvenim drevesom, kot je prikazan na sliki 3.27. Podatkovno množico naložimo s komponento »ArffLoader« in ji določimo razred s komponento »ClassAssigner«. Tako spremenjeno podatkovno množico pošljemo komponenti »CrossValidationFoldMaker«. V konfiguracijah komponente določimo podrobnosti navzkrižne validacije in tako učno kot testno množico pošljemo v obdelavo algoritmu v komponento »J48«. Rezultate klasifikacije pošljemo komponenti »ClassifierPerformanceEvaluator«, ki izvede analizo uspešnosti v skladu s konfiguracijo nastavitev. Rezulate v tekstovni obliki pošljemo komponenti »TextViewer«, komponenta »TextViewer« pa prav tako v tekstovni obliki pošlje iste podatke komponenti »J48«. Proces začnemo s klikom na modro puščico. Po zagonu procesa do podatkov dostopamo preko individualnih komponent. Če želimo dostopati do podatkov uspešnosti klasifikacije, tako na meniju komponente »TextViewer« izberemo možnost »Show results«. Ker smo iz »TextViewer« poslali rezultate v tekstovni obliki komponenti »J48«, lahko preko »Show results« dostopamo do modela ASCII generiranega drevesa [33].



Slika 3.27 Podatkovni tok klasifikacije z uporabo vmesnika KnowledgeFlow

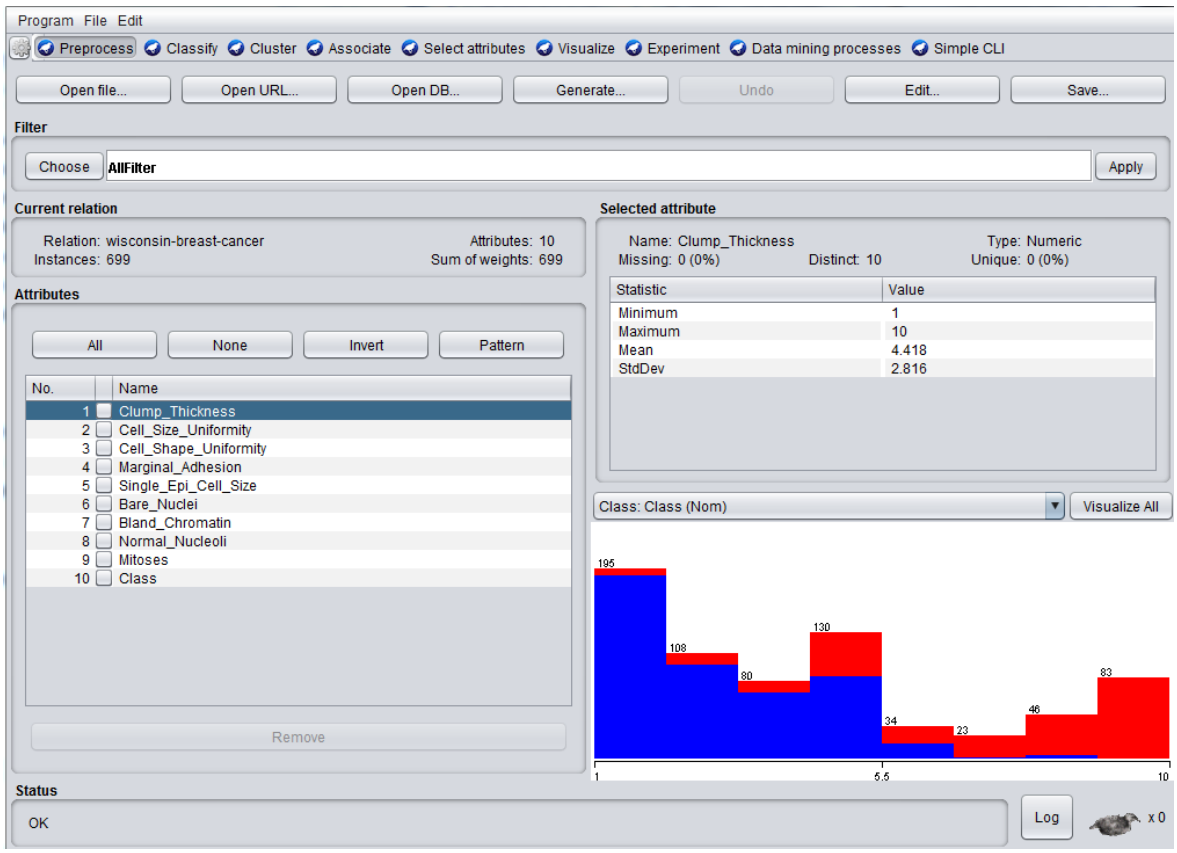
Experimenter

Weka »Experimenter« je okolje, ki uporabniku omogoča, da ustvari, poganja, modificira in analizira eksperimente na bolj učinkovit način, kot če bi vsak algoritem v določeni skupini podatkovnega rudarjenja zagnali posebej. Razdeljen je na tri zavihke: »Setup«, »Run« in »Analyse«. V zavihku »Setup« definiramo nastavitve eksperimenta. To vključuje nastavitve validacije, število iteracij eksperimenta, uvoz podatkovnih množic in izbiro algoritmov, učinkovitost katerih bomo primerjali. V naprednem načinu imamo možnost ročne izbire in spreminjanja lastnosti generatorja rezultatov. Prav tako imamo možnost izvajanja enega eksperimenta na več sistemih, kjer vsi stroji jemljejo podatke iz iste podatkovne baze. V zavihku »Run« zaženemo in spremljamo stanje izvajanja eksperimenta. V zavihku »Analyse« dostopamo do rezultatov testiranj, prav tako imamo možnost izvajanja statističnih testov nad rezultati [33].

Workbench

Workbench (slika 3.28) je združitev vseh prej omenjenih aplikacij in preproste ukazne vrstice v en uporabniški vmesnik. Glavna prednost uporabe tega načina je, da enkrat

naložene podatkovne množice ni treba ponovno nalagati, če želimo uporabiti funkcionalnosti druge aplikacije.



Slika 3.28 Workbench

3.3.2.2 Implementacija klasifikacije v lastno rešitev z uporabo Weka API

Uporabnik lahko z uporabo Weka API na preprost način integrira funkcionalnosti v svoje rešitve. Na voljo je veliko knjižnih in spletnih virov, primernih tako za osnovne uporabnike kot za eksperte. Za reševanje problema je skoraj vedno relevantna uporaba naslednjih razredov [3]:

- **Instance:** Objekti razreda »Instance« shranjujejo uporabnikove podatke,
- **Filter:** Metode predprocesiranja podatkov,
- **Classifier/Clusterer:** Metode ustvarjanja modelov klasifikacije/gručenja,
- **Evaluating:** Metode ocenjevanja,
- **Options:** Vmesnik za določanje lastnosti operacij (klasifikacije, gručenje, filtri),
- **AttributeSelection:** Vmesnik, ki definira metode izbire atributov na podlagi podanih meril.

Na preprostem primeru, prikazanem na sliki 3.29, si poglejmo gradnjo in ocenjevanje modela klasifikacije. Algoritem klasifikacije začnemo z definicijo lokacije izvora. Iz lokacije izvora z razredom *Instance* iz vira ekstrahiramo podatkovno množico in nastavimo razredni atribut, če ta še ni nastavljen kot del množice. Za izvedbo navzkrižne validacije moramo kot parametre metode *crossValidateModel()* razreda *Evaluation* podati objekt drevesa, podatkovno množico, število rezov in seme generatorja števil, za namene generiranja naključnih particij podatkov. Z metodo *toSummaryString()* izpišemo povzetek rezultatov validacije klasifikacije, z metodo *toClassDetailsString()* pridobimo informacije o statističnih značilnostih razredov, z metodo *toMatrixString( )* pa izpišemo matriko zmede [3].

```
public static void treeClassifier() throws Exception {
    //Uvoz podatkovne množice
    DataSource source = new DataSource("E:/Users/Aleksej/breast-w.arff");
    Instances data = source.getDataSet();
    //Nastavimo razredni atribut
    if (data.classIndex() == -1) {
        data.setClassIndex(data.numAttributes() - 1);
    }
    //Instanciranje odločitvenega drevesa
    J48 tree = new J48();
    tree.setUnpruned(true);
    //Izvedba klasifikacije
    Evaluation eval = new Evaluation(data);
    eval.crossValidateModel(tree, data, 10, new Random(1));
    //Izpis povzetka rezultatov klasifikacije
    System.out.println(eval.toSummaryString("\nResults\n======\n", false));
    //Podrobna statistika klasifikacije
    System.out.println(eval.toClassDetailsString());
    //Prikaz matrike zmede
    System.out.println(eval.toMatrixString());
}
```

Slika 3.29 Algoritem klasifikacije in navzkrižne validacije z uporabo Weka API

3.3.3 Razširljivost

Weka omogoča razširitev obstoječega nabora značilk. Tako lahko razširjamo ali prilagajamo metode klasificiranja, gručenja, metode izbora atributov, filtrov in veliko več. Vse na novo dodane lastnosti zazna grafični vmesnik brez dodatnega programiranja, kar omogoča enostavno in hitro primerjavo novih rešitev z obstoječimi implementacijami. Poleg razširitev

lahko izboljšamo osnovne funkcionalnosti z implementacijo uporabniško definiranih možnosti.

Od različice Weka 3.7.2 naprej se posamezne funkcionalnosti ločujejo na module, imenovane paketi (angl. Package). Paketi so nadgradnja osnovnih funkcionalnosti platforme. Vsebujejo datoteke jar, dokumentacijo, meta podatke in potencialno tudi izvirno kodo. Uvedba paketov je omogočila poenostavitev jedra platforme, saj uporabniki namestijo le pakete, ki jih potrebujejo. Prav tako razvijalcem ponujajo enostaven mehanizem za distribucijo lastnega dela. Na voljo so paketi, ki osnovnim funkcionalnostim dodajajo učne sheme ali razširijo funkcionalnost osnovne platforme. Veliko izbiro obstoječih vtičnikov najdemo v repozitoriju Weka na strani SourceForge [38].

3.4 Teoretična primerjava RapidMinerja in Weke

V tem poglavju bomo primerjali prej analizirani platformi. Primerjali ju bomo po metodologiji, podrobneje opisani v poglavju 3.1. Ker bi bilo nepošteno primerjati plačljiv izdelek z zastonjskim, bomo z Weko primerjali RapidMiner Studio Community Edition.

3.4.1 Splošne značilnosti

Obe platformi sta napisani v jeziku Java in ponujata tako grafični uporabniški vmesnik, ukazno vrstico kot tudi API. Uporabljata se za splošno namensko podatkovno rudarjenje. Glavna razlika med orodjema je v načinu dela, naboru funkcionalnosti, možnosti dostopa do razširitev in licenci, pod katero sta zaščiteni. Omenjena dejstva prikazuje tabela 3.5. RapidMiner vsakemu uporabniku ponuja do 20 MB prostora na repozitoriju na oblačni tehnologiji RapidMiner Cloud, preko katerega lahko na strežnikih podjetja izvajamo svoje procese.

Tabela 3.5 Značilnosti platform

Lastnost	**RapidMiner Studio**	**Weka**
Programski jezik	Java 7 (Java 8 za Linux)	Java 7
Licenca	Business Source / AGP v3	GNU General Public
Najnovejša stabilna različica	7.1	3.8.0
Uporabniški vmesnik / ukazna vrstica	Oboje	Oboje
Operacijski sistem	Kjer je na voljo JRE	Kjer je na voljo JRE

3.4.2 Vidik podatkovnih virov in funkcionalnosti

Predpriprava podatkov

V tabeli 3.6 je prikazana primerjava podprtih formatov podatkovnih virov med platformama. RapidMiner Studio je zaradi poslovne odločitve na tem področju hudo okrnjen, saj dovoljuje samo vire formatov csv, txt in odprtokodnih baz. Plačljiva različica to težavo seveda odpravi. Če hočemo v RapidMiner Studio Community naložiti podatkovno množico v formatu arff, jo moramo iz Weke izvoziti v obliki formata csv. Weka se v celoti pri uvozu podatkovnih virov odreže dosti bolje, saj je ekstremno fleksibilna. Dejstvo, da ne podpira formata Excel, ni težava, ker se da razpredelnice enostavno uvoziti kot format csv. V primerjavi z Weko je RapidMiner boljši pri razčlenjevanju datotek csv. To se opazi predvsem v kakovosti predlogov pri čarovniku ekstrahiranja podatkovnih množic iz vira.

Tabela 3.6 Podprti formati podatkovnih virov

Format	**RapidMiner Studio Community**	**Weka**
Tekstovne datoteke	CSV, TXT	Da
Nezvezni formati za podatkovno rudarjenje	Ne	Da
Tabele iz PB	Samo odprtokodne PB	Samo PB z JDBC gonilnikom
Excel	Da	Ne
Podatki iz povezave (URL)	Ne	Da

Integracija raznolikih tehnik

Zahvaljujoč dolgoletnemu razvoju in odprtokodni naravi obe platformi ponujata veliko različnih shem za podatkovno rudarjenje, vseeno pa je izbira algoritmov RapidMinerja proti Weki za spoznanje bogatejša. Količina in fleksibilnost platform sta primerljivi le pri algoritmih klasifikacije. Za veliko večino ostalih tipov obdelave podatkov in ustvarjanje modelov RapidMiner prednjači tako po številu algoritmov kot po fleksibilnosti gradnje procesov. Prav tako ima močnejša orodja klasične statistike in simulacij, saj omogoča posebne operatorje s skriptami jezika Python in R. RapidMiner ima možnost uvoza večine Wekinih operacij (uvoz podatkov, validacija, modeliranje) v obliki operatorjev.

Skalabilnost

Kar se tiče skalabilnosti, se platformi razlikujeta predvsem v načinu, kako skalabilnost dosežeta. Weka dobro skalabilnost doseže z inkremetalnim vzorčenjem in z možnostjo uporabe tehnologij Hadoop in Spark. Zastonjska različica RapidMinerja določeno mero skalabilnosti doseže s pomočjo tehnologije RapidMiner Cloud, vendar je le-ta omejena zaradi poslovnega modela. To pomeni, da je Weka veliko bolj primerna za obdelovanje velikih podatkov (angl. Big Data). RapidMiner Studio Professional omogoča popolno integracijo tehnologije Radoop, močno podporo izvajanja procesov na Oblaku in izvajanje modeliranja neposredno na podatkovni množici v podatkovni bazi.

3.4.3 Uporabniški vidik

Primerjava grafičnih vmesnikov

Najbolj primerljivi metodi dela preko grafičnega vmesnika sta RapidMiner in Wekin »KnowledgeFlow«. Pri obeh vmesnikih procese strukturiramo tako, da izberemo gradnike in jih med seboj logično povežemo. Razlika je v načinu delovanja in posledično intuitivnosti grajenja in modifikacije procesov. V RapidMinerju v bistvu gradimo drevesa (en začetni element in poljubno veliko končnih), kar omogoča implementacijo prekinitvenih točk in enostavne definicije ponovne uporabe posameznih gradnikov. Wekin »KnowledgeFlow« je v bistvu graf, katerega točke so posamezne operacije. To pomeni, da proces poženemo, do rezultatov pa dostopamo preko posameznih komponent. Velika razlika je prav tako v količini pomoči in informacij, do katere lahko uporabnik dostopa iz glavnega vmesnika. Do pomembnih funkcionalnosti lahko večinoma v RapidMinerju dostopamo hitreje kot v Weki. RapidMiner ima interaktivne vodiče, integrirano tehnologijo »Wisdom of the Crowds« in podrobno razlago tako operatorjev (vključno z interaktivnimi primeri uporabe) kot posameznih parametrov operatorja. V primeru težave s procesom nam orodje na enostaven način sporoči, kaj je narobe s pomočjo simbolov in predlogi za rešitev. Pomoč, ki je na voljo v Weki, je proti omenjenemu omejena.

Aplikaciji »Explorer« in »Experimenter« ponujata alternativni način dela z uporabniškim vmesnikom, ki v RapidMinerju ni na voljo. Omenjeni uporabniški vmesnik blesti, kadar izvajamo preproste operacije podatkovnega rudarjenja. Poleg enostavnosti so nam možnosti predprocesiranja v teh aplikacijah na voljo nemudoma, v enem vmesniku. Prav tako lahko neposredno v meniju uporabljamo orodja vizualizacije, ne da bi morali dostopati do za to posebnega menija. Za kompleksnejše procese je RapidMiner boljši, ker lahko poleg boljše vizualizacije problema in bolj kakovostne pomoči posamezne dele procesa enostavno odstranjujemo, dodajamo in spreminjamo. Posledično so rezultati bolje prilagojeni problemu, ki ga rešujemo.

Ena izmed pomembnejših lastnosti RapidMinerja je koncept repozitorija. Repozitorij centralizira podatke in procese, tako da so uporabniku vedno na voljo. To je posebej pomembno pri grajenju kompleksnejših procesov, kjer uporabljamo več podatkovnih

množic ali nam posamezni procesi služijo kot podprocesi. Weka funkcionalnosti repozitorija ne ponuja, ampak moramo datoteke ob vsakem zagonu naložiti s pomočjo raziskovalca.

Primerjava Java API

Z metodo veriženja operatorjev je zaradi pomanjkanja dokumentacije izredno težko napisati že osnoven algoritem klasifikacije na podatkovni množici. Način implementacije operatorjev v procese ima smisel na ravni uporabniškega vmesnika, vendar je na ravni kode ne intuitiven in nepotrebno obširen. Wekin API poleg več dokumentacije za uporabnika ponuja učinkovitejši način pisanja kode.

Nalaganje procesa iz xml je v primerjavi z Weka API veliko preprostejše, saj le uvozimo proces xml, ki smo ga ustvarili v uporabniškem vmesniku, in iz njega poberemo rezultate. Na tak način lahko v kodo implementiramo zelo kompleksne procese, ki z uporabo Wekinega API niso mogoči ali pa so težje izvedljivi. Slabost nalaganja procesov iz datotek xml je, da je treba uporabljati razčlenjevalnike xml, če želimo dinamično nalagati podatkovne množice. Weka nam to možnost ponuja neposredno z API. Omenjena dejstva so razvidna iz tabele 3.7.

Tabela 3.7 Primerjava Weka in RapidMiner API

Lastnost	**RapidMiner API – Nalaganje procesa iz xml**	**RapidMiner API – metoda veriženja operatorjev**	**Weka API**
Količina dokumentacije	Malo	Malo	Veliko
Zahtevnost	Enostavno	Težko	Srednje
Dinamično nalaganje podatkovnih množic	Da, z razčlenjevalnikom XML	Da	Da
Priporočilo razvijalca	Da	Ne	Da

Vizualizacija podatkov in modelov

Weka je proti RapidMinerju na področju vizualizacije podatkov in rezultatov omejena, kar vidimo po tabeli 3.8. Velja omeniti, da ima RapidMiner proti Weki praviloma več možnosti prikaza posameznega grafa. Dobra plat tega je fleksibilnost za naprednega uporabnika, vendar se lahko neizkušeni uporabnik kaj hitro izgubi v obilici nastavitev.

Tabela 3.8 Primerjava kategorij vizualizacije

Kategorija vizualizacije	**RapidMiner Studio**	**Weka**
Histogrami	Da	Da
Raztrosni grafi	Da	Da
Ostali grafi (škatla z brki, povprečje/napaka ...)	Da	Ne
3D-grafi	Da	Ne

Razširljivost

V obe platformi je tako obstoječe kot nove funkcionalnosti enostavno implementirati, le da izvedba poteka na različen način. RapidMiner ponuja ogrodje, ki je fasada RapidMiner API,

s pomočjo katerega implementiramo elemente lastnega operatorja. Tako ustvarjene operatorje lahko s pomočjo prej omenjenega ogrodja testiramo, jim spremenimo videz in jim dodamo dokumentacijo. Pri razširjanju Weke ne dostopamo do posebnega ogrodja, ampak razširimo funkcionalnosti jedra programa z uporabo vmesnikov. Tako razvite rešitve Weka zazna samodejno in temu primerno populira vmesnik.

Skupnost

Tako RapidMiner kot Weka imata veliko skupnost, ki razvija rešitve in pomaga pri reševanju težav na forumih platforme. Na posebej pereče ali zahtevne probleme se velikokrat odzovejo tudi razvijalci. Weka se trenutno sooča s trendom upadanja števila uporabnikov, medtem ko se je RapidMiner uveljavil kot eden izmed voditeljev na področju programske opreme za podatkovno rudarjenje. To lahko sklepamo iz rezultatov letnih anket strani KDNuggets. Tabela 3.9 povzema rezultate štirih letnih anket [39] [40] [41] [42]. Posamezne vrednosti predstavlja delež anketirancev, ki so orodje uporabljali za projekt, in delež anketirancev, ki so platformo za projekt za namene podatkovnega rudarjenja uporabljali izključno. Opazimo, da dobršen delež uporabnikov RapidMinerja orodje uporablja izključno. Iz tega lahko sklepamo, da je orodje dovolj vsestransko, da za majhne projekte specialistična programska oprema ni potrebna. Tega za Weko iz danih podatkov ne moremo reči. Za Weko je zaradi odprtokodne licence posebej pomembno, da obdrži ali poveča delež uporabnikov, saj so nove funkcionalnosti in izboljšave odvisne tudi od njih.

Tabela 3.9 Delež uporabe orodij

Leto	**RapidMiner (%)**	**RapidMiner izključno (%)**	**Weka (%)**	**Weka izključno (%)**
2013	39,2	30,9	14,3	5,6
2014	44,2	35,1	17,0	0,0
2015	31,5	13,7	11,2	0,0
2016	32,6	11,7	10,9	0,0

4 PRIMERJAVA IMPLEMENTACIJ ALGORITMOV RAPIDMINER IN WEKA

V naslednjem poglavju smo primerjali učinkovitost v drugem poglavju opisanih algoritmov glede na platformo. Cilj je ugotoviti, katera platforma ima boljše implementacije algoritmov podatkovnega rudarjenja pri predpostavki, da so specifike izvajanja le-teh čim bolj primerljive.

4.1 Izvedba eksperimenta

Za izvajanje eksperimenta smo uporabili namizni računalnik s procesorjem Intel Core i5 2500K 3.3 GHz in pomnilnikom DDR3 1600 MHz z 8 GB prostora. Primerjali smo različice algoritmov, ki jih ponujata orodja Weka 3.8.0 in RapidMiner 7.1.1. Orodja smo poganjali v 64-bitni različici operacijskega sistema Windows 7.

V vsakem eksperimentu smo uporabili deset podatkovnih množic različnih velikosti, števil atributov in tipov atributov. Lastnosti posamezne podatkovne množice so prikazane v tabeli 4.1. Za vsako podatkovno množico smo z uporabo klasifikacijskih algoritmov odločitveno drevo, k-najbližjih sosedov in naivni bayes ustvarili prediktivni model ter izračunali točnost, preciznost, priklic in F-mero vsake množice z uporabo k-kratne navzkrižne validacije z desetimi rezi.

Platformi uteženo povprečno preciznost in uteženi povprečni priklic računata drugače zaradi drugačne metode pripisovanja uteži posameznim razredom. Weka pripiše večjo utež razredom, ki se v podatkovni množici pojavljajo pogosteje, medtem ko RapidMiner vsem razredom pripiše utež 1. Normalizacijo preciznosti in priklica smo bili primorani izvesti v orodju Microsoft Excel, ker nobena izmed platform ne ponuja računanja uteži na način, ki ga uporablja druga platforma. Izvedli smo jo tako, da smo izračunali povprečje vrednosti preciznosti in priklica razredov podatkovne množice brez aplikacije Wekinih uteži, iz katerih smo izračunali F-mero po enačbi 2.9. Ker RapidMiner ne podpira neposrednega izračuna F-mere za podatkovne množice s polinominalnimi razredi, smo morali le-to prav tako izračunati s pomočjo orodja Excel.

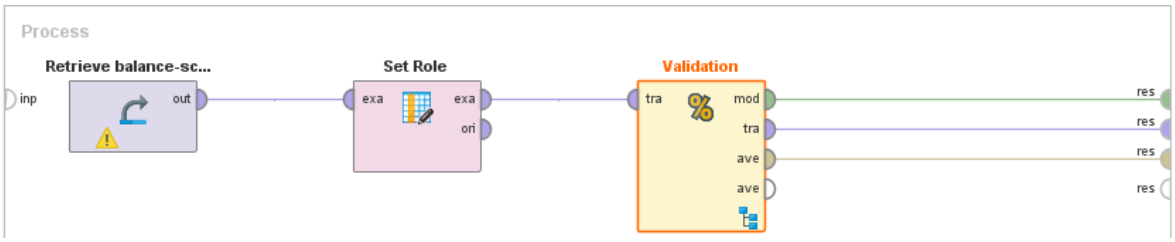
Na platformi Weka smo eksperiment izvajali preko aplikacije »Explorer«. V eksperiment smo vključili Wekine implementacije prej omenjenih algoritmov. To so iBK kot implementacija k-NN algoritma, RandomForest kot implementacija Naključnega gozda in NaiveBayes kot implementacija algoritma Naivni Bayes.

Tabela 4.1 Uporabljene podatkovne množice

Ime podatkovne množice	Število razredov	Število atributov	Število primerkov
iris	3	5	150
glass	7	10	214
breast-w	2	10	699
c-lens	3	17	57
diabetes	2	9	768
sonar	2	61	208
p-tumor	21	18	339
heart-s	2	14	270
audiology	24	70	226
b-scale	3	5	625

Izvedba eksperimenta na platformi RapidMiner

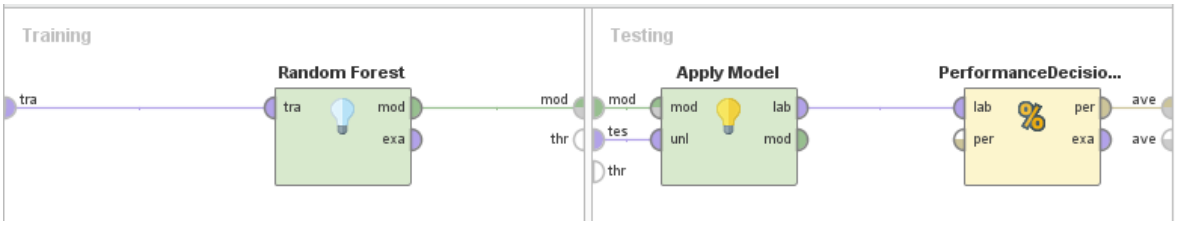
Za izvedbo eksperimenta na platformi RapidMiner smo v aplikaciji RapidMiner Studio zgradili splošen proces klasifikacije, sestavljen iz operatorjev »Retrieve«, »Set Role« in »Validation«. Proces je prikazan na sliki 4.1.



Slika 4.1 Proces klasifikacije v orodju RapidMiner Studio

Podproces »Learning« procesa »Validation« vsebuje operator relevantnega algoritma. V namene izvajanja eksperimenta so to operatorji »Naive Bayes«, »k-NN« in »Decision Tree«. Podproces »Training« vsebuje operator »Apply Model«, podproces »Testing« pa operatorja

»Apply Model« in »Performance«. Primer implementacije klasifikacije z metodo Naključni gozd je prikazan na sliki 4.2. Pri eksperimentih za preostala dva prej omenjena algoritma operator »Random Forest« zamenjamo z operatorjem »k-NN« ali »Naive Bayes«. Pri vseh operatorjih smo uporabili privzete nastavitve, razen pri operatorju »Naive Bayes«, ki smo mu izklopili parameter »Laplace transformation«. Wekina različica algoritma te funkcije namreč privzete ne podpira.



Slika 4.2 Podproces "Validation", kot je implementiran za izvedbo eksperimentov

4.2 Analiza rezultatov eksperimenta

V nadaljevanju bomo primerjali vrednosti točnosti in F-Mere primerljivih algoritmov obeh platform. Poudarimo, da so rezultati eksperimenta odvisni od značilnosti vzorčnih množic in specifičnih implementacij algoritmov. Dejstvo, da so implementacije različne, vidimo med drugim tudi iz dejstva, da imajo RapidMinerjeve različice algoritmov več možnih nastavitev. To pomeni, da lahko posamezne parametre algoritma optimiziramo glede na tip problema, ki ga rešujemo. Kot smo omenili zgoraj, smo pri obeh algoritmih uporabil privzete nastavitve, razen pri primerih, kjer smo navedli drugače.

4.2.1 Analiza točnosti

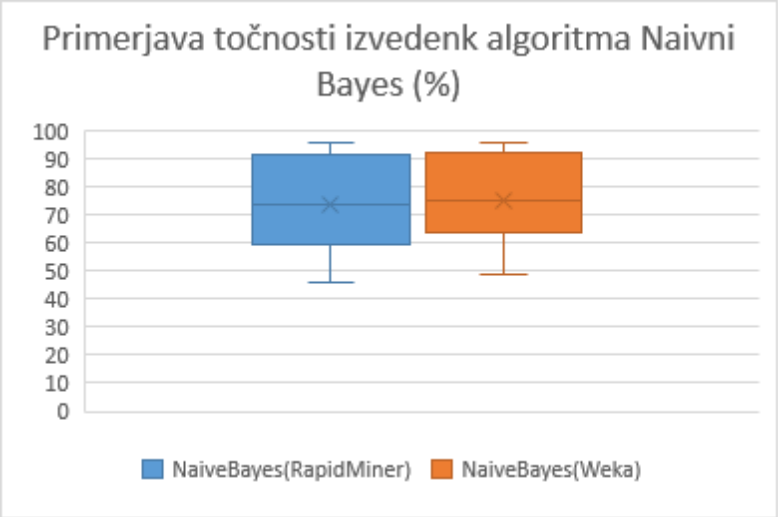
V tabeli 4.2 so zapisane vrednosti točnosti algoritmov glede na posamezne množice, izražene v odstotkih.

Tabela 4.2 Točnost klasifikacije orodij RapidMiner in Weka (%)

	RapidMiner			*Weka*		
	NaiveBayes	k-NN	RandomForest	NaiveBayes	iBK	RandomForest
iris	94,67	96,00	93,33	96,00	95,33	96,00
glass	47,66	72,43	64,95	48,59	70,56	66,82
breast-w	95,99	94,42	94,42	95,99	95,13	94,56
c-lens	70,83	75,00	75,00	70,83	79,16	83,33
diabetes	75,91	67,45	71,48	76,30	70,18	73,82
sonar	67,31	82,69	68,75	67,79	86,53	71,15
p-tumor	46,02	37,46	44,54	50,14	39,23	39,82
heart-s	83,70	57,78	70,74	83,70	75,18	76,66
audiology	63,72	74,78	65,93	73,45	77,87	77,87
b-scale	90,72	78,40	77,12	90,40	86,56	76,64
mediana	*73,37*	*74,89*	*76,37*	*74,88*	*78,52*	*80,58*

Primerjava točnosti izvedenk algoritma Naivni Bayes

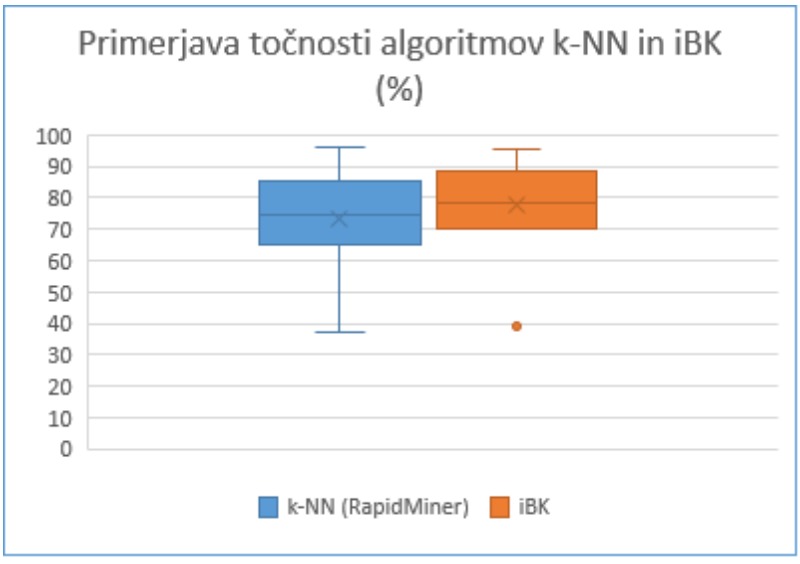
Na sliki 4.3 je prikazana škatla z brki, ki prikazuje in primerja podrobnosti rezultatov točnosti implementacij Naivni Bayes. Mediana Wekinih rezultatov je večja od RapidMinerjevih (M_{Weka} = 74,88; $M_{RapidMiner}$ = 73,37), prav tako pa je pri rezultatih Weke manjši kvartilni razmik. Minimuma in maksimuma sta primerljiva. Iz navedenih dejstev sklepamo, da je Wekina implementacija algoritma Naivni Bayes na izbranih podatkovnih množicah točnejša.



Slika 4.3 Primerjava točnosti izvedenk algoritma NaiveBayes

Primerjava točnosti izvedenk algoritma k-najbližjih sosedov

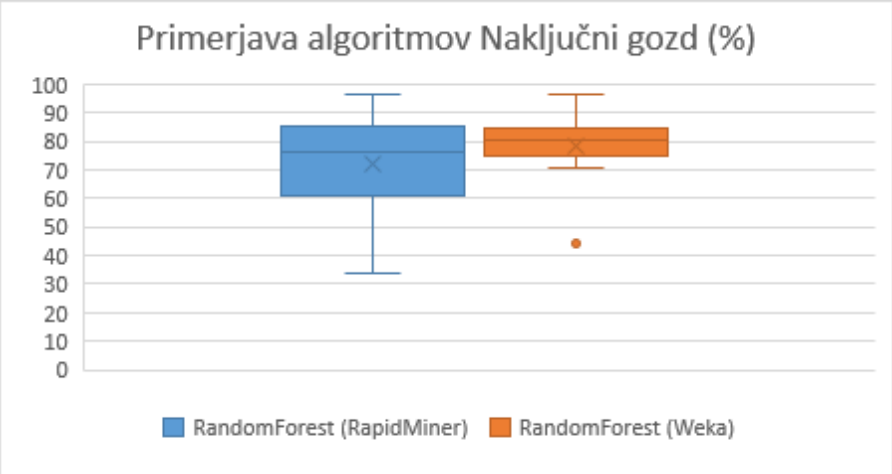
S slike 4.4 se da razbrati, da je algoritem iBK točnejši od RapidMinerjevega k-NN. Čeprav sta maksimuma primerljiva, se minimuma razlikujeta kar za 32,54 odstotnih točk v korist iBK. Omembe vredno je sicer, da ima iBK osamelca pri 39,23 % točnosti, kar je z RapidMinerjevim minimumom bolj primerljivo. iBK ima prav tako večjo mediano (M_{Weka} = 78,52; $M_{RapidMiner}$ = 74,89) in manjši kvartilni razmik, kar pomeni, da je bolj konsistenten pri točnem napovedovanju razredov.



Slika 4.4 Primerjava točnosti algoritmov k-NN in iBK

Primerjava točnosti izvedenk algoritma Naključni gozd

Škatla z brki na sliki 4.5 kaže veliko razliko v točnosti implementacij algoritmov Naključni gozd v korist Wekini implementaciji. Čeprav je razlika med medianami majhna (M_{Weka} = 80,58; $M_{RapidMiner}$ = 76,37), je razlika v kvartilnem razmiku opazna, saj so pri Wekini implementaciji algoritma točnosti množic zgoščene med 75 in 85 odstotkov, pri RapidMinerjevi implementaciji pa med 60 in 85 odstotkov. Razlika je velika tudi pri minimumih množic. Pri Weki opazimo osamelca, ki je vseeno občutno višje vrednosti od RapidMinerjevega minimuma.



Slika 4.5 Primerjava točnosti algoritmov RandomForest

4.2.2 Analiza F-mere

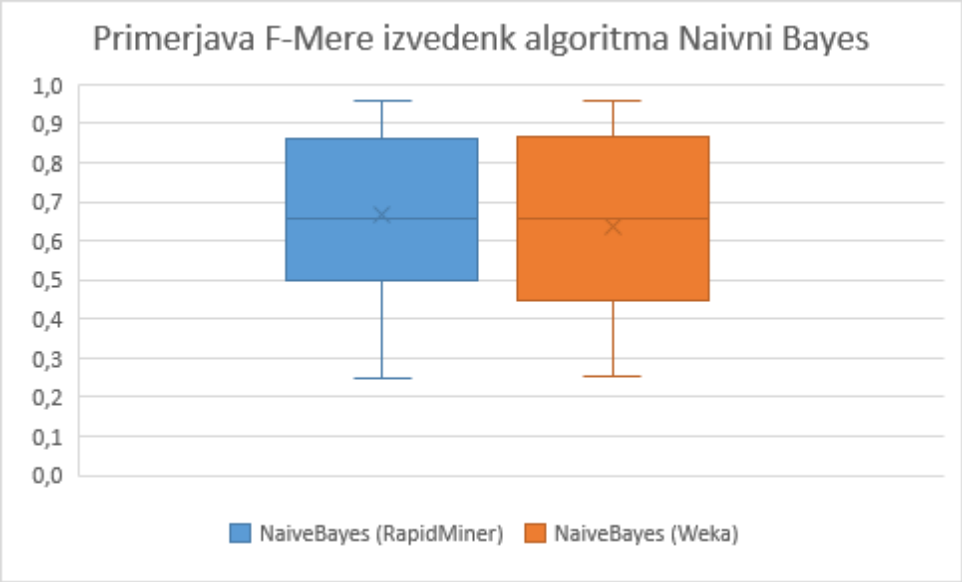
V tabeli 4.3 najdemo vrednosti F-Mere algoritmov glede na posamezne množice.

Tabela 4.3 F-Mera klasifikacije orodij RapidMiner in Weka

	RapidMiner			*Weka*		
	NaiveBayes	k-NN	RandomForest	NaiveBayes	iBK	RandomForest
iris	0,947	0,960	0,934	0,960	0,953	0,960
glass	0,501	0,700	0,625	0,512	0,661	0,654
breast-w	0,957	0,938	0,939	0,957	0,946	0,940
c-lens	0,622	0,727	0,717	0,622	0,768	0,816
diabetes	0,729	0,640	0,666	0,734	0,666	0,708
sonar	0,689	0,826	0,685	0,693	0,865	0,711
p-tumor	0,249	0,202	0,242	0,262	0,216	0,186
heart-s	0,834	0,570	0,703	0,834	0,749	0,763
audiology	0,493	0,501	0,337	0,251	0,510	0,381
b-scale	0,629	0,579	0,570	0,540	0,536	0,461
mediana	0,658	0,659	0,656	0,656	0,708	0,670

Primerjava F-Mere izvedenk algoritma Naivni Bayes

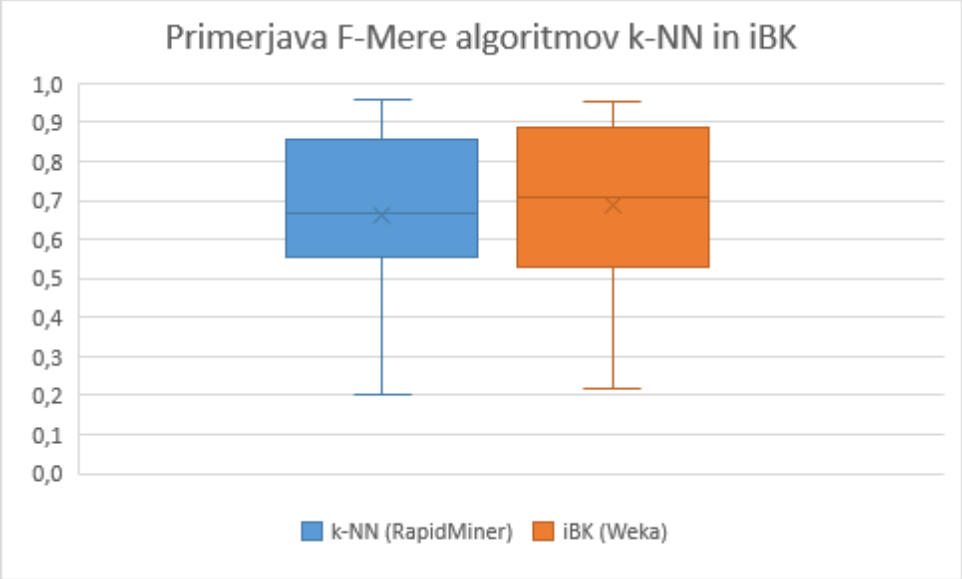
Na grafu, prikazanem na sliki 4.6, vidimo, da imata mediani F-Mere primerjanih algoritmov zelo majhne razlike (M_{Weka} = 0,656; $M_{RapidMiner}$ = 0,658). Maksimuma in minimuma sta prav tako primerljiva. Največja razlika med algoritmoma je v kvartilnem razmiku, saj je v primerjavi z Wekino različico RapidMinerjeva bolj zgoščena pri višjih vrednostih. To nam pove, da imata priklic in preciznost pri RapidMinerjevem Naivnem Bayesu pogosteje višje vrednosti kot pri Wekinem.



Slika 4.6 Primerjava F-Mere izvedenk algoritma NaiveBayes

Primerjava F-Mere izvedenk algoritma k-najbližjih sosedov

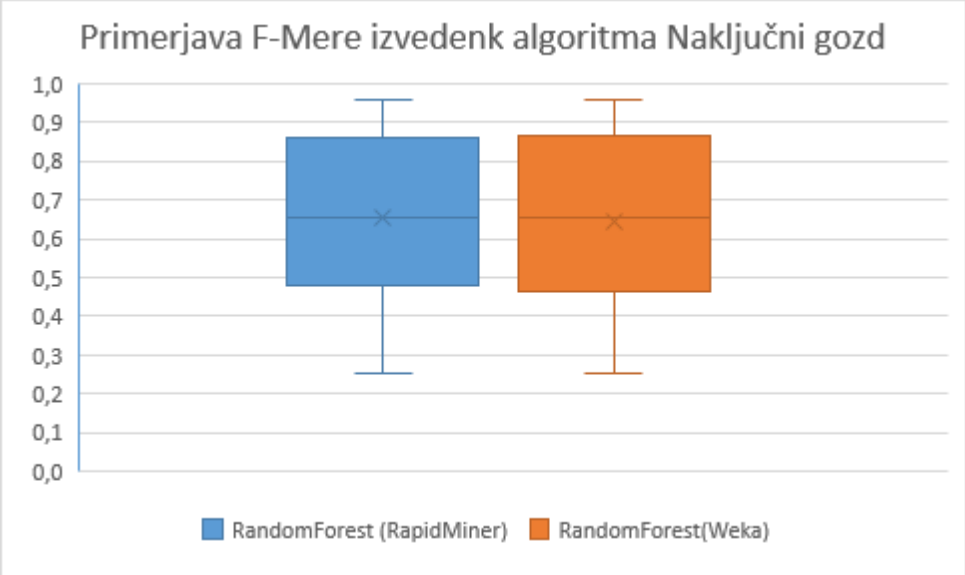
Mediana (M_{Weka} = 0,708; $M_{RapidMiner}$ = 0,659) F-Mere preizkušenih algoritmov na omenjenih podatkovnih množicah je večja pri Wekini različici algoritma, kvartilni razmik pa je manjši pri RapidMinerjevi različici. Pri Weki je tretji kvartil metrike F-Mera opazno večji od RapidMinerjevega, prvi kvartil pa opazno manjši. Iz tega sklepamo, da je RapidMinerjev algoritem bolj konsisten, pri Wekinem pa je večja verjetnost, da dosežemo višjo mero priklica in preciznosti. Graf opisanega je prikazan na sliki 4.7.



Slika 4.7 Primerjava F-Mere algoritmov k-NN in iBK

Primerjava F-mere izvedenk algoritma Naključni gozd

Iz grafa na sliki 4.8 vidimo, da so F-Mere množic, na katerih smo izvajali eksperiment z algoritmom Naključni gozd, primerljive. Mediana Wekinih meritev je malenkost večja od RapidMinerjevih (M_{Weka} = 0,670; $M_{RapidMiner}$ = 0,656), v korist RapidMinerjevi izvedenki pa kaže manjši kvartilni razmik. Maksimumi in minimumi so primerljivi.



Slika 4.8 Primerjava F-Mere izvedenk algoritma Random Forest

4.3 Rezultati statističnih testov

Ker smo želeli ugotoviti, katera izvedenka algoritma je učinkovitejša ne glede na uporabljeni vzorec, smo na eksperimentalnih rezultatih izvedli statistične teste. Najprej smo izvedli Shapiro-Wilkov test [43], s katerim smo preverili, ali vzorčne točnosti in F-Mere ustrezajo normalni porazdelitvi. V primeru, da so bile vzorčne vrednosti porazdeljene normalno, smo uporabili Studentov t-test za odvisne vzorce [44], sicer smo uporabili Wilcoxonov test predznačenih rangov [45]. Vse teste izvajamo s stopnjo zaupanja $\alpha = 0.05$.

4.3.1 Statistični testi točnosti

V tabeli 4.4 vidimo rezultate testa Shapiro-Wilk vzorčne normalne porazdelitve točnosti. Če je vrednost statistične značilnosti večja od stopnje zaupanja, je vzorec porazdeljen normalno, sicer hipoteze o normalni porazdeljenosti ne sprejmemo. Upoštevajoč rezultate smo pri primerjavi implementacij Naivnega Bayesa uporabili parametrični test, saj je Shapiro-Wilkov test pokazal, da sta oba vzorca porazdeljena normalno ($p > 0,05$ za oba vzorca). Pri primerjavi metod Naključnega gozda smo uporabili neparametrični test, saj je test porazdelitve za Wekino različico algoritma pokazal, da vrednosti točnosti ne ustrezajo normalni porazdelitvi ($p = 0,038$). Pri primerjavi metode k-NN smo ponovno uporabili parametrični t-test za odvisne vzorce za primerjavo obeh implementacij, saj sta oba vzorca porazdeljena normalno (pri obeh je $p > 0,05$).

Tabela 4.4 Rezultati testa Shapiro-Wilk vzorčne normalne porazdelitve točnosti

	Shapiro-Wilk		
	Statistična vrednost	Stopnje svobode	Vrednost statistične značilnosti *p*
NaivniBayes_Weka	0,923	10	0,384
NaivniBayes_RM	0,930	10	0,444
RandomForest_RM	0,879	10	0,127
RandomForest_Weka	0,834	10	0,038
iBK_Weka	0,866	10	0,089
kNN_RM	0,934	10	0,486

t-test za točnost izvedenk algoritma Naivni Bayes

Ker iz rezultatov testa, navedenih v tabeli 4.5, razberemo, da je dvostranska vrednost statistične značilnosti večja od stopnje zaupanja ($p = 0,124$), trdimo, da imata enako populacijsko povprečje točnosti. Hipoteze torej ne zavrnemo in trdimo, da sta Wekina in RapidMinerjeva izvedenka algoritma Naivni Bayes enako točna.

Tabela 4.5 Rezultati dvostranskega t-testa odvisnih vzorcev za točnost algoritma Naivni Bayes

	t	Stopnje svobode	Dvostranska vrednost statistične značilnosti *p*
NaivniBayes_Weka- NaivniBayes_RM	1,694	9	0,124

t-test za točnost izvedenk algoritma k-najbližjih sosedov

Dvostranska vrednost statistične značilnosti je pri statističnem testu enakosti populacijskih točnosti algoritmov k-najbližjih sosedov enaka stopnji zaupanja (p = 0,050), zato hipoteze ne zavrnemo. Podobno kot pri prejšnjem testu trdimo, da sta Wekina in RapidMinerjeva izvedenka algoritma k-najbližjih sosedov enako točna.

Tabela 4.6 Rezultati dvostranskega t-testa odvisnih vzorcev za točnost algoritma k- najbližjih sosedov

	t	Stopnje svobode	Dvostranska vrednost statistične značilnosti *p*
iBK_Weka- kNN_RM	2,264	9	,050

Wilcoxonov test predznačenih rangov za točnost izvedenk algoritma Naključni gozd

Iz tabele 4.7 razberemo, da je asimptotična dvostranska vrednost statistične značilnosti večja od stopnje zaupanja (p = 0,114). Na podlagi tega lahko trdimo, da je Wilcoxonov predznačni test pokazal, da med primerjanima izvedenkama algoritma Naključni gozd ni občutnih razlik v točnosti.

Tabela 4.7 Rezultati Wilcoxonovega testa predznačenih rangov odvisnih vzorcev za točnost izvedenk algoritma Random Forest

	RandomForest_RM - RandomForest_Weka
Z	-1,580
Asimptotična dvostranska značilnost *p*	0,114

4.3.2 Statistični testi F-Mere

Podobno kot pri točnosti smo tudi za F-Mero ugotovili, ali so vrednosti v danih vzorcih porazdeljeni normalno. Kot vidimo v tabeli 4.8, je vrednost statistične značilnosti pri vseh vzorcih večja od stopnje zaupanja (vsi vzorci ustrezajo normalni porazdelitvi), zato smo na vzorcih izvajali dvostranske t-teste odvisnih vzorcev.

Tabela 4.8 Rezultati Shapiro-Wilkovega testa vzorčne normalne porazdelitve F-Mere

	Shapiro-Wilk		
	Statistična vrednost	Stopnje svobode	Vrednost statistične značilnosti *p*
NaivniBayes_Weka	0,933	10	0,475
NaivniBayes_RM	0,960	10	0,789
RandomForest_RM	0,968	10	0,867
RandomForest_Weka	0,955	10	0,730
iBK_Weka	0,932	10	0,469
kNN_RM	0,948	10	0,648

t-test za F-Mere izvedenk algoritma Naivni Bayes

Rezultat dvostranskega t-testa odvisnih vzorcev nam da dvostransko vrednost statistične značilnosti, ki je večja od stopnje značilnosti testa (p = 0,264). Hipoteze ne moremo zavrniti, torej se F-Mere algoritmov Naivni Bayes statistično značilno ne razlikujejo. Navedena dejstva, iz katerih sklepamo, so prikazana v tabeli 4.9.

Tabela 4.9 Rezultati dvostranskega t-testa odvisnih vzorcev za F-Mere algoritma Naivni Bayes

	t	Stopnje svobode	Dvostranska vrednost statistične značilnosti *p*
NaivniBayes_Weka- NaivniBayes_RM	-1,192	9	0,264

t-test za F-Mere izvedenk algoritma k-najbližjih sosedov

Kot lahko razberemo iz tabele 4.10, hipoteze tudi pri tem testu ne moremo zavrniti, saj je dvostranska vrednost statistične značilnosti večja od stopnje značilnosti (p = 0,252).

Zaključimo, da se F-Mere izvedenk algoritma k-najbližjih sosedov statistično pomembno ne razlikujejo.

Tabela 4.10 Rezultati dvostranskega t-testa odvisnih vzorcev za F-Mere algoritma k-najbližjih sosedov

	t	Stopnje svobode	Dvostranska vrednost statistične značilnosti *p*
NaivniBayes_Weka- NaivniBayes_RM	1,223	9	0,252

t-test za F-Mere izvedenk algoritma Naključni gozd

Kot pri prejšnjih testih F-Mere je tudi tukaj iz tabele 4.11 razvidno, da je dvostranska vrednost statistične značilnosti večja od stopnje zaupanja (p = 0,373). Iz tega sledi, da hipoteze testa ne zavrnemo, kar pomeni, da je točnost vzorcev primerljiva.

Tabela 4.11 Rezultati dvostranskega t-testa odvisnih vzorcev za F-Mere algoritma Naključni gozd

	t	Stopnje svobode	Dvostranska vrednost statistične značilnosti *p*
NaivniBayes_Weka- NaivniBayes_RM	0,938	9	0,373

5 SKLEP

Neizprosnemu napredku strojnega učenja in podatkovnega rudarjenja morajo za preživetje na trgu zelo hitro slediti tako splošne kot specialistične platforme, saj podatkovni znanstveniki in analitiki zahtevajo vedno več funkcionalnosti. V diplomskem delu smo se glede na to omejili na primerjavo platform RapidMiner in Weka. Weko 3.8.0 smo v interesu poštenosti primerjali samo z RapidMiner Studio Community Edition 7.1.1, saj so druga razvojna okolja znotraj platforme plačljiva. Platformi smo primerjali tako teoretično kot eksperimentalno.

V teoretičnem delu smo definirali metodologijo, ki smo jo uporabili kot vodilo za identifikacijo in analizo najpomembnejših aspektov odprtokodnih platform podatkovnega rudarjenja. Ugotovili smo, da platforma RapidMiner v primerjavi z Weko prednjači po številu funkcionalnosti, pomoči uporabnikom, možnosti vizualizacije, intuitivnem uporabniškem vmesniku in predprocesiranju. Posebej velja omeniti zastonjske storitve, kot so RapidMiner Cloud in RapidMiner Repository, ki uporabniku omogočajo uporabo strežnikov podjetja za shranjevanje podatkov in procesov ter izvajanje operacij na oblaku. Zastonjska različica RapidMinerja je v primerjavi z Weko šibka na področju integracije platforme v lastno rešitev, preprostosti uporabe za začetnike in uvoza podatkovnih virov v sistem. Weka s pomočjo posebnih distribucij omogoča integracijo platform Apache Hadoop in Spark, s katerimi učinkoviteje obdelujemo velike podatke, medtem ko zastonjski RapidMiner te možnosti nima. Obe platformi imata dobre možnosti razširitve in dopolnitve obstoječih zmožnosti.

V eksperimentalnem delu smo primerjali metrike točnosti in F-mere platformnih implementacij algoritmov k-najbližjih sosedov, Naivni Bayes in Naključni gozd na desetih vzorcih. Rezultate smo primerjali s pomočjo škatel z brki. Ugotovili smo, da je vzorčna točnost pri vseh tipih algoritmov boljša pri Wekinih izvedenkah, vzorčne F-Mere pa so primerljive, saj so razlike v rezultatih majhne. Za vsako metriko smo izvedli Shapiro-Wilkov test za vsako implementacijo algoritma, s katerim smo ugotovili, ali so vrednosti implementacije porazdeljene normalno ali ne. V primeru, da so vzorčne vrednosti obeh implementacij algoritma porazdeljene normalno, smo uporabili dvostranski t-test odvisnih

vzorcev, sicer pa Wilcoxonov predznačni test. Za obe metriki pri vseh algoritmih velja, da razlike med rezultati niso statistično pomembne.

VIRI

[1] L. Kart , G. Herschel, A. Linden in J. Hare, „Magic Quadrant for Advanced Analytics Platforms,“ 2016.

[2] P. Domingos, "A Few Useful Things to Know about Machine Learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78-87, 2012.

[3] I. H. Witten, F. Eibe in M. Hall, Data Mining: Practical Machine Learning Tools And Techniques, Morgan Kaufmann, 2005.

[4] I. Kononenko, Strojno Učenje, Ljubljana: Založba FE in FRI, 2005.

[5] S. Kotsiantis, „Supervised Machine Learning: A Review of Classification Techniques,“ *Informatica*, Izv. 31, pp. 249-268, 2007.

[6] I. Hendrickx in A. van Den Bosch, „Hybrid Algorithms with Instance-Based Classification,“ v *Machine Learning: ECML 2005*, Porto, 2005.

[7] Z. Ghahramani, „Unsupervised Learning,“ v *Advanced Lectures on Machine Learning*, Berlin, Springer Berlin Heidelberg, 2014, pp. 72-112.

[8] C. Szepesvari, Synthesis lectures on Artificial Intelligence and Machine Learning, Alberta: Morgan & Claypool Publishers, 2009.

[9] L. Kozma, „k Nearest Neighbors algorithm (KNN),“ 2 20 2008. [Elektronski]. Available: http://www.lkozma.net/knn2.pdf. [Poskus dostopa 18 7 2016].

[10] I. Rish, „An empirical study of the naive bayes classifier,“ v *International Joint Conference for Artifical Intelligence*, 2001.

[11] T. Mitchell, „Decision Tree Learning,“ v *Machine Learning*, Pittsburgh, McGraw Hill, 1997, pp. 55-79.

[12] J. Ali, N. Ahmad, R. Khan in I. Maqsood, „Random Forests and Decision Trees,“ *International Journal of Computer Science Issues*, Izv. 9, št. 5, pp. 272-278, 2012.

[13] F. Livingston, „Implementation of Breiman's Random Forest Machine Learning Algorithm,“ *Machine Learning Journal Paper*, 2005.

[14] D. J. MacKay, „An Example Inference Task: Clustering,“ v *Information Theory, Inference and Learning Algorithms*, Cambridge, Cambridge University Press, 2003, pp. 284-292.

[15] J. Yadav in M. Sharma, „A Review of K-Mean Algorithm,“ *International Journal of Engineering Trends and Technology*, Izv. 4, št. 7, pp. 2972-2975, 2013.

[16] V. Kotu in B. Deshpande, Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner, Morgan Kaufmann, 2015.

[17] X. Chen, G. Williams in X. Xu, „A Survey of Open Source Data Mining Systems,“ v *Emerging Technologies in Knowledge Discovery and Data Mining*, Nanjing, Springer Berlin Heidelberg, 2007, pp. 3 - 14.

[18] M. Wildenius in L. Nyman, „Introducing "Business Source": The Future of Corporate Open Source Licensing?,“ *Technology Innovation Management Review,* Izv. 6, pp. 5-12, 2003.

[19] RapidMiner, „RapidMiner Studio Documentation,“ [Elektronski]. Available: http://docs.rapidminer.com/studio/. [Poskus dostopa 22 7 2016].

[20] RapidMiner, „RapidMiner Studio Pricing,“ [Elektronski]. Available: http://rapidminer.com/products/comparison/. [Poskus dostopa 22 7 2016].

[21] RapidMiner, „RapidMiner Server Administration,“ [Elektronski]. Available: http://docs.rapidminer.com/server/administration/. [Poskus dostopa 22 Julij 2016].

[22] RapidMiner, „RapidMiner Radoop Documentation,“ [Elektronski]. Available: http://docs.rapidminer.com/radoop/. [Poskus dostopa 22 Julij 2016].

[23] RapidMiner, „RapidMiner Cloud Documentation,“ [Elektronski]. Available: http://docs.rapidminer.com/cloud/. [Poskus dostopa 22 Julij 2016].

[24] Rapid-I GmBH, "RapidMiner Studio User Manual," 2014. [Online]. Available: https://rapidminer.com/wp-content/uploads/2014/10/RapidMiner-v6-user-manual.pdf.

[25] RapidMiner, „RapidMiner Fact Sheet,“ [Elektronski]. Available: http://www.rapid-i.com/downloads/brochures/RapidMiner_Fact_Sheet.pdf. [Poskus dostopa 22 7 2016].

[26] S. Kirstein, S. Land in D. Halfkann, „RapidMiner 7: How to extend RapidMiner,“ [Elektronski]. Available: http://docs.rapidminer.com/downloads/RapidMiner-extensions.pdf.

[27] RapidMiner, „RapidMiner Fact Sheet,“ [Elektronski]. Available: http://www.rapid-i.com/downloads/brochures/RapidMiner_Fact_Sheet.pdf. [Poskus dostopa 22 Julij 2016].

[28] RapidMiner, „RapidMiner - How Big Is Big Data,“ RapidMiner, [Elektronski]. Available: https://rapidminer.com/rapidminer-big-data-how-big-is-big/. [Poskus dostopa 23 Julij 2016].

[29] RapidMiner, „Wisdom of the Crowds: A Guiding Light,“ RapidMiner, [Elektronski]. Available: https://rapidminer.com/wisdom-crowds-guiding-light/. [Poskus dostopa 23 7 2016].

[30] RapidMiner, „RapidMiner Studio Doxygen documentation,“ Fossies, [Elektronski]. Available: https://fossies.org/dox/rapidminer-studio-7.0.1-src/annotated.html. [Poskus dostopa 23 7 2016].

[31] RapidMiner, „Integrating RapidMiner Into Your Application,“ RapidMiner, [Elektronski]. Available: https://rapid-i.com/wiki/index.php?title=Integrating_RapidMiner_into_your_application . [Poskus dostopa 23 Julij 2016].

[32] RapidMiner, „RapidMiner Marketplace,“ RapidMiner, [Elektronski]. Available: https://marketplace.rapidminer.com/UpdateServer/faces/index.xhtml. [Poskus dostopa 23 Julij 2016].

[33] R. R. Bouckaert, F. Eibe, M. Hall, P. Reutemann, R. Kirkby, A. Seewald in D. Scuse, Weka Manual for 3.8.0, Waikato: University of Waikato, 2016.

[34] Penataho, „Data Mining Algorithms and Tools in Weka,“ [Elektronski]. Available: http://wiki.pentaho.com/display/DATAMINING/Data+Mining+Algorithms+and+Tools+in+Weka. [Poskus dostopa 27 Julij 2016].

[35] Weka, „Mining Big Data Using Weka 3,“ [Elektronski]. Available: http://www.cs.waikato.ac.nz/ml/weka/bigdata.html. [Poskus dostopa 27 Julij 2016].

[36] M. Hall, „Weka and Hadoop,“ Weka, [Elektronski]. Available: http://markahall.blogspot.si/2013/10/weka-and-hadoop-part-1.html. [Poskus dostopa 27 Julij 2016].

[37] M. Hall, „Weka and Spark,“ Weka, [Elektronski]. Available: http://markahall.blogspot.si/2015/03/weka-and-spark.html. [Poskus dostopa 27 Julij 2016].

[38] R. R. Bouckaert, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann in I. H. Witten, „WEKA - Experiences with Java Open-Source Project,“ *Journal of Machine Learning Research,* št. 11, pp. 2533-2541, 2010.

[39] KDnuggets, „KDnuggets Annual Software Poll: RapidMiner and R vie for first place,“ [Elektronski]. Available: http://www.kdnuggets.com/2013/06/kdnuggets-annual-software-poll-rapidminer-r-vie-for-first-place.html. [Poskus dostopa 27 Julij 2016].

[40] KDnuggets, „KDnuggets 15th Annual Analytics, Data Mining, Data Science Software Poll: RapidMiner Continues To Lead,“ [Elektronski]. Available: http://www.kdnuggets.com/2014/06/kdnuggets-annual-software-poll-rapidminer-continues-lead.html. [Poskus dostopa 27 Julij 2016].

[41] KDNuggets, „R leads RapidMiner, Python catches up, Big Data tools grow, Spark Ignites,“ [Elektronski]. Available: http://www.kdnuggets.com/2015/05/poll-r-rapidminer-python-big-data-spark.html. [Poskus dostopa 27 Julij 2016].

[42] KDnuggets, „R, Python Duel As Top Analytics, Data Science software – KDnuggets 2016 Software Poll Results,“ [Elektronski]. Available: http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html/2. [Poskus dostopa 27 Julij 2016].

[43] S. S. Shapiro in M. Wilk, „An Analysis of Variance Test for Normality (Complete samples),“ *Biometrika,* Izv. III/IV, št. 52, pp. 591-611, 1965.

[44] B. Fadem, High-Yield Behavioral Science (High-Yield Series), Hagertwon: Lippincot, Williams & Wilkins, 2008.

[45] J. Demšar, „Statistical Comparisons of Classifiers over Multiple Data Sets,“ *Journal of Machine Learning Research,* št. 7, pp. 1-30, 2006.

ZA ŠTUDENTE E, ITK, MK, R-IT in TK

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

FERI

IZJAVA O AVTORSTVU

Spodaj podpisani/-a ALEKSEJ MILOŠEVIČ

z vpisno številko E10G88C5

sem avtor/-ica diplomskega dela z naslovom: ANALIZA IN PRIMERJAVA PLATFORM ZA PODATKOVNO RUDARJENJE RAPIDMINER IN WEKA

(naslov diplomskega dela)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

RED. PROF. DR. VILI PODGORELEC

in somentorstvom (naziv, ime in priimek)

ASIST. SAŠO KARAKATIČ

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela.
- soglašam z javno objavo elektronske oblike diplomskega dela v DKUM.

V Mariboru, dne 5.9.2016

Podpis avtorja/-ice:

Alek. Milošević

ZA ŠTUDENTE E, ITK, MK, R-IT in TK

Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

FERI

IZJAVA O USTREZNOSTI ZAKLJUČNEGA DELA

Podpisani mentor :

VILI PODGORELEC

(ime in priimek mentorja)

in somentor (eden ali več, če obstajata):

SAŠO KARAKATIČ

(ime in priimek somentorja)

Izjavljam (-va), da je študent

Ime in priimek: ALEKSEJ MILOŠEVIČ

Vpisna številka: E10688C5

Na programu: INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA

izdelal zaključno delo z naslovom:

ANALIZA IN PRIMERJAVA PLATFORM ZA POPATKOVNO PROCESIRANJE RAPIDMINER IN WEKA

(naslov zaključnega dela v slovenskem in angleškem jeziku)

ANALYSIS AND COMPARISON OF DATA MINING PLATFORMS RAPIDMINER AND WEKA

v skladu z odobreno temo zaključnega dela, Navodilih o pripravi zaključnih del in mojimi (najinimi oziroma našimi) navodili.

Preveril (-a, -i) in pregledal (-a, -i) sem (sva, smo) poročilo o plagiatorstvu.

Datum in kraj:
2.9.2016, MARIBOR

Podpis mentorja:

Datum in kraj:
2.9.2016, MARIBOR

Podpis somentorja (če obstaja):
Karakatič Jaš

Priloga:

- Poročilo o preverjanju podobnosti z drugimi deli.

Fakulteta za elektrotehniko, računalništvo in informatiko

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUČNEGA DELA IN OBJAVI OSEBNIH PODATKOV DIPLOMANTOV

Ime in priimek avtorja-ice: ALEKSEJ MILOŠEVIČ

Vpisna številka: E10B816B

Študijski program: INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA

Naslov zaključnega dela: ANALIZA IN PRIMERJAVA PLATFORM ZA PODATKOVNO RUDARJENJE RAPIDMINER IN WEKA

Mentor: VILI PODGORELEC

Somentor: SAŠO KARAKATIČ

Podpisani-a ALEKSEJ MILOŠEVIČ izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Zaključno delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija zaključnega dela je istovetna z elektronsko verzijo elektronski verziji, ki sem jo oddal za objavo v Digitalno knjižnico Univerze v Mariboru.

Zaključno delo zaradi zagotavljanja konkurenčne prednosti, varstva industrijske lastnine ali tajnosti podatkov naročnika: _____ ne sme biti javno dostopno do _____(datum odloga javne objave ne sme biti daljši kot 3 leta od zagovora dela).

Podpisani izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum zaključka študija, naslov zaključnega dela), na spletnih straneh in v publikacijah UM.

Datum in kraj: 5.9.2016, MARIBOR Podpis avtorja-ice: Alekj Milošč

Podpis mentorja: _____
(samo v primeru, če delo ne sme biti javno dostopno)

Podpis odgovorne osebe naročnika in žig:
(samo v primeru, če delo ne sme biti javno dostopno) _____