UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Jože Gobar

UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

Diplomsko delo

Maribor, oktober 2015

UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

Diplomsko delo

Študent:	Jože Gobar
Študijski program:	Univerzitetni študijski program Informatika in tehnologije komuniciranja
Smer:	Informacijski sistemi
Mentor:	red. prof. dr. Vili Podgorelec
Somentor:	asist. Sašo Karakatič
Lektor:	Ivana Vekjet Frančeškin, univ. dipl. slov.

Univerza v Mariboru

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

FERI

Številka: E1064711
Datum in kraj: 19. 06. 2015, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 46/2012)
izdajam

SKLEP O DIPLOMSKEM DELU

1. **Jožetu Gobarju,** študentu univerzitetnega študijskega programa INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA, smer Informacijski sistemi, se dovoljuje izdelati diplomsko delo.

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

3. **Naslov diplomskega dela:**
   UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

4. **Naslov diplomskega dela v angleškem jeziku:**
   EFFICIENCY OF LEARNING ALGORITHMS ON BIG DATA

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. 2015 v referatu za študentske zadeve.

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

Dekan:
red. prof. dr. Borut Žalik

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

ZAHVALA

Iskreno se zahvaljujem svojemu mentorju, red. prof. dr. Vilju Podgorelcu, in somentorju, asist. Sašu Karakatiču, za vso pomoč pri izdelavi diplomske naloge in čas, ki sta mi ga posvetila.

Zahvaljujem se tudi družini, ki mi je ves čas študija stala ob strani in me je na tej poti podpirala.

UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

Ključne besede: veliki podatki, karakteristike velikih podatkov, strojno učenje, klasifikacija, učinkovitost učnih algoritmov

UDK: 004.421:004.65(043.2)

Povzetek

Tematika te diplomske naloge so veliki podatki, karakteristike velikih podatkov in učni algoritmi, ki jih uporabljamo za klasifikacijo. V diplomski nalogi predstavljam tudi rezultate eksperimenta, s katerim sem ugotavljal učinkovitost učnih algoritmov na velike podatke. Učinkovitost algoritmov sem ovrednotil s klasifikacijsko točnostjo in časovnim izvajanjem učnih algoritmov na podatkovnih množicah. Iz pridobljenih rezultatov lahko sklepam, da se algoritmi glede na dane podatkovne množice različno obnašajo ter da je izbira učnega algoritma za analizo podatkovnih množic odvisna predvsem od problema in zastavljenega cilja.

EFFICIENCY OF LEARNING ALGORITHMS ON BIG DATA

Keywords: big data, big data characteristics, machine learning, classification, learning algorithms efficiency

UDK: 004.421:004.65(043.2)

Summary

This thesis presents the subject of big data and its characteristics, learning algorithms applied in classification, as well as the results of an applied experiment in order to determine learning algorithms efficiency on big data. Algorithm efficiency has been assessed by classification accuracy and timely implementation of learning algorithms on datasets. The results indicate that algorithms perform differently considering given dataset and that preference of a learning algorithm intended for dataset analysis depends upon the posed problem and objective.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

UPORABLJENE KRATICE

WEKA (angl. Waikato Environment for Knowledge Analysis)

SVM (angl. Support vector machines)

k-NN (angl. *k*-Nearest Neighbors algorithm)

MGI (angl. McKinsey Global Institute)

1 Uvod

Količina podatkov na svetu nenehno narašča. Podjetja zajemajo bilijone bajtov podatkov o svojih kupcih, dobaviteljih in procesih z mrežnimi senzorji, vgrajenimi v fizičen svet preko naprav, kot so mobilni telefoni in avtomobili, ki zaznavajo, ustvarjajo in posredujejo podatke [1]. Hitro naraščajoči podatki, v strukturirani in nestrukturirani obliki, različnih formatih, kot so video, avdio, slike, tekst, ki dosegajo meje nepredstavljivega, predstavljajo značilnost dobe velikih podatkov (angl. Big data era), v kateri se trenutno nahajamo. Kljub temu velika količina podatkov ni vzrok za takšno poimenovanje, saj je vedno obstajalo več podatkov, kot smo jih bili zmožni obdelati [2]. Razlika med dobo velikih podatkov in prejšnjimi obdobji je preobrat, ki ga je doživelo gospodarstvo in podjetja. Ta želijo danes uporabljati vse podatke, ki jih imajo na voljo ali pa jih lahko zberejo, za obdelavo in analizo, da bi iz njih pridobili novo znanje, vzorce, zgradili nove modele, ter jih izkoristili za boljše sprejemanje odločitev sedaj in v prihodnosti (napovedovanje, diagnostika, nadzor, preverjanje, simulacije itd.).

Podatkov danes ni težko pridobiti, pojavi pa se izziv pridobivanja pravilnih, se pravi primernih podatkov, ki jih lahko analiziramo s pomočjo računalnikov. Analiziramo jih z namenom pridobivanja novega znanja in identifikacije vzorcev v njih, ki jih brez pomoči računalnikov ne bi mogli odkriti oz. bi za to uporabili predolgo časa.

Analiza podatkov mora biti dovolj hitra, da bo pridobljeno znanje v danem trenutku za nas koristno, saj lahko pridobljeno znanje postane nekoristno, če ga ne moremo pravočasno uporabiti. S tem namenom uporabljamo različne metode strojnega učenja, kot so klasifikacija (uvrščanje), regresija, učenje asociacij in logičnih relacij, učenje sistemov (diferencialnih) enačb in razvrščanje [3], ki nam omogočajo hitro analizo podatkov in grajenja modelov iz njih. Učinkovitost učnih algoritmov pa se na velikih podatkih lahko zelo razlikuje, zato se bom v diplomski nalogi posvetil metodi klasifikacije, na kratko bom predstavili najpogostejše učne algoritme, kot so odločitvena drevesa z algoritmoma C4.5 in naključni gozdovi implementacije orodja WEKA, naivni Bayesov klasifikator, k-najbližjih sosedov (k-NN), klasifikator po metodi podpornih vektorjev (SVM) in AdaBoost.m1. S

pomočjo meritev bom poskušal ugotoviti, kateri učni algoritmi so najučinkovitejši glede na zastavljene probleme, kot so hitra analiza podatkov, velika točnost podatkov in kompromis med točnostjo in hitrostjo klasifikacije.

Osnovni namen diplomske naloge je analiziranje učinkovitosti učnih algoritmov na velikih podatkih. V drugem poglavju sem na kratko predstavil, kaj so veliki podatki, kakšna je njihova definicija in katere so karakteristike, ki podatke naredijo velike. V tretjem poglavju sem predstavil metodo klasifikacije in pripadajoče algoritme.

V praktičnem delu sem predstavil izbrane podatkovne množice, pridobljene iz UCI [4] in mldata [5] shrambe podatkov. Nad njimi sem izvedel izbrane učne algoritme z orodjem WEKA [6]. Učinkovitost učnih algoritmov sem meril s časom, ki ga algoritem uporabi za učenje modela in izvajanje klasifikacije, ter z merami točnosti učnega algoritma. Za mere točnosti sem izbral klasifikacijsko točnost in F-measure ali F-score. Rezultate sem analiziral in jih predstavil v grafični in tabelarični obliki.

2 Veliki podatki

Z večanjem količine digitalnih podatkov, ki se zbirajo in nastajajo v vsakem sektorju našega gospodarstva, vsaki organizaciji in vsakem uporabniku sodobnih tehnologij, se je pojavil izraz »veliki podatki« (angl. Big Data), ki opisuje fenomen shranjevanja, zbiranja, nastajanja ter uporabe velikih količin podatkov. Fenomen velikih podatkov si najpreprosteje razlagamo kot množice velikih količin podatkov, nad katerimi izvajamo analitike, kot so strojno učenje, podatkovno rudarjenje ter druge statistične metode za pridobivanje vrednosti iz podatkov. Ta sposobnost pridobivanja, zajemanja in analiziranja podatkov z nižjimi stroški, kot je bilo v preteklosti mogoče, je tisto, kar predstavlja takšno zanimanje za velike podatke. Iz te razlage lahko sklepamo, da izraz »veliki podatki« ni zgolj ustvarjanje in uporaba velikih količin podatkov, ampak predstavlja analizo vseh podatkov, ki so okoli nas [7].

Podjetja zbirajo podatke, da bi iz njih pridobili čim večjo vrednost, ki jim bo koristila pri poslovanju in sprejemanju odločitev zdaj in v prihodnosti. Da je najpomembnejša lastnost »velikih podatkov« ravno vrednost oz. znanje, ki ga pridobimo z uporabo analitičnih metod, kot je strojno učenje, predstavlja primer velikih podatkov, ki ga je podal David Feinleib v knjigi Big Data Bootcamp [8]. Feinleib v primeru s potresi na enostaven način prikaže, da niso dovolj zgolj podatki, s pomočjo katerih lahko napovemo potres, ampak je bistveno, da te podatke hitro analiziramo. Saj nam podatki, ki lahko napovejo potres en dan vnaprej, ne koristijo nič, če njihova obdelava traja dva dni. Iz tega enostavnega primera je razvidno, da se pojem »veliki podatki« navezuje na veliko več kot pa zgolj na količino podatkov. Sami podatki nam lahko dajo pravo vrednost, ki je lahko za nas koristna, samo v primeru če jo pridobimo pravočasno.

V tem poglavju bom predstavil pojem »veliki podatki« in njegove karakteristike ter nekaj primerov uporabe velikih podatkov v našem vsakdanu.

2.1 Definicija izraza »veliki podatki«

Kljub temu, da se pojem »veliki podatki« v današnjem času vse pogosteje uporablja, kar je razvidno iz njegove popularnosti iskanja v Googlovem iskalniku [2], ki od leta 2011 narašča, sem med prebiranjem literature naletel na različne definicije, ki ta pojem opisujejo. V eni izmed največjih poslovnih študij z naslovom »Big data: The next frontier for innovation, competition, and productivity« MGI definira velike podatke kot podatkovne množice, katerih velikost presega zmožnosti tipične programske opreme za podatkovne baze, orodij za zajem, shranjevanje, upravljanje in analizo [1]. MGI v definiciji poudarja, da ne obstaja konkretna meja v količini podatkov, ki so uvrščeni med velike podatke, ampak je to odvisno od okoliščin in podatkov samih. Kljub temu v definiciji kot edini kriterij za razvrščanje uporabljajo količino podatkov ali obseg. Takšna uporaba pojma »veliki podatki« je lahko zavajajoča, saj kaže predvsem na to, da je pojem odvisen od količine podatkov. V tem primeru pojem ne bi predstavljal nič novega, saj tema, kako ravnati z velikimi količinami podatkov na določeni točki oz. na določenem trenutku, že dolgo obstaja na področju raziskovanja podatkovnih baz.

Torej, če upoštevam aktualnost tematike »veliki podatki«, mora izraz predstavljati veliko več kot pa zgolj količino podatkov. Večina publikacij širi takšno definicijo. Ena od teh je podana v IDC študiji »The Digital Universe«. IDC velike podatke opredeljuje kot novo generacijo tehnologij in arhitektur, zasnovanih za pridobivanje vrednosti iz zelo velikih in raznolikih množic podatkov z omogočanjem njihovega hitrega zajemanja, odkrivanja in analiziranja. Pri tem obstajajo tri glavne značilnosti velikih podatkov: podatki sami, analiza podatkov in predstavitev rezultatov analize [9].

Ta definicija temelji na 3 V modelu (angl. 3V's model), ki ga je ustvaril Doug Laney leta 2001 [10]. Laney v svojem delu ni uporabil izraza »veliki podatki«, ampak je napovedal, da bo obvladovanje podatkov v elektronskem trgovanju postalo čedalje bolj pomembno in zahtevno. Laney je 3 V-je opredelil kot največje izzive za upravljanje s podatki. Prvi V je količina podatkov (angl. data volumen), drugi je hitrost podatkov (angl. data velocity) in tretji je raznolikost podatkov (angl. data variety). Količina podatkov predstavlja velikost podatkovne množice, hitrost podatkov predstavlja hitrost, s katero podatki prispejo oz. pridobimo, ustvarjamo in analiziramo nove podatke, raznolikost se nanaša na različne vire, iz katerih podatke pridobimo, pri tem pa so podatki lahko strukturirani ali pa nestrukturirani. Ko so se začele razprave o velikih podatkih, je industrija, predvsem s področja poslovanja, sprejela Laneyjev 3 V model za opredelitev pojma »veliki podatki«.

V literaturi pa se pojavljajo še druge definicije, ki bodisi slonijo na 3 V modelu in ga nekoliko preoblikujejo ali pa se od njega nekoliko odmaknejo. Edd Dumbill, analist pri O'Reilly Media, v članku »What is big data? An introduction to the big data landscape.« [11] velike podatke opiše kot podatke, ki presegajo zmogljivosti obdelave podatkov konvencionalnih sistemov za upravljane s podatkovnimi bazami. Pri tem pa poudari, da so podatki preveliki, se premikajo prehitro (s tem mislimo na hitrost nastajanja, pridobivanja in analiziranja podatkov) ali pa ne ustrezajo strukturi, ki jo uporabljamo v podatkovni bazi. Za pridobitev vrednosti iz takšnih podatkov je treba izbrati alternativne načine za njihovo obdelavo. Eden teh, ki se je nekoliko odmaknil od 3 V modela, je Tim Kraska. Kraska kljub temu prizna, da so »veliki podatki« veliko več kot zgolj velika količina podatkov. »Velike podatke« opiše kot podatke, katerih normalna uporaba sedanjih tehnologij uporabnikom ne omogoča, pravočasnega, stroškovno eficientnega in kakovostnega odgovora na podatkovno usmerjena vprašanja [12]. Odprto pa pusti vprašanje, katere karakteristike podatkov gredo preko »normalnih« zmožnosti sedanjih tehnologij. Ostali avtorji se še vedno osredotočajo predvsem na količino podatkov, kot edino karakteristiko [13] ali pa formalne definicije ne podajo [12].

Nasploh so 3 V model ali njegove razširitve največkrat uporabljeni za opis pojma »veliki podatki«. Zraven treh osnovnih karakteristik se največkrat pojavlja še verodostojnost oz. zaupanje v podatke (angl. data veracity) in v neki meri predstavlja rezultat med hitrostjo in raznolikostjo podatkov.

Kljub svoji točnosti in pravilnosti naštete definicije ne zajamejo resnične vrednosti izraza »veliki podatki«. Velike podatke bi morali meriti z vplivom, ki ga imajo oz. ga ustvarijo, ne glede na količino prostora za shranjevanje podatkov in procesorske moči, ki ga porabijo. Vse pogosteje se razprave o »velikih podatkih« usmerjajo na karakteristike vnosnih podatkov (koliko terabajtov, petabajtov bomo porabili za shranjevanje in obdelavo) in ne v znanje in vrednost, ki ga iz njih lahko pridobimo. Zato se zraven naštetih karakteristik velikih podatkov pojavi še vrednost podatkov (angl. data value). S tem izrazom bom v diplomski nalogi zajel 5 V-jev za opis velikih podatkov, katerih opisi sledijo v naslednjem podpoglavju.

2.2 Karakteristike velikih podatkov

Pri karakteristiki velikih podatkov ni potrebe po tem, da je vseh pet karakteristik prisotnih v vsakem trenutku, da bi podatke označili kot »velike podatke«. Velja pa, da lahko obstaja

vsaka varianta karakteristik, ki naredi problem težek ali nemogoč za obdelavo s tradicionalnimi tehnologijami, da podatke smatramo za velike podatke. Pri tem pa nam morajo veliki podatki zagotoviti vrednost podatkov in pomoč pri doseganju zastavljenega cilja, ki je lahko napovedovanje, postavljanje diagnostike, simulacije itd..
Najprej bom definiral tri osnovne karakteristike 3 V modela, nato pa še verodostojnost in vrednost podatkov kot novejši karakteristiki velikih podatkov.

2.2.1 Količina podatkov

Količina podatkov (angl. data volume) se navezuje na velikost ustvarjenih podatkov iz različnih virov ter na koristi, pridobljene s sposobnostjo za njihovo obdelavo in analizo. Ravno njihova analiza predstavlja glavno zanimanje za velike podatke. Več podatkov, kot imamo na razpolago, je bolje, kot pa najboljši modeli ali algoritem za njihovo obdelavo. Primer: z večjo količino podatkov lahko upoštevamo tudi več faktorjev kot prej, s tem pridobimo boljše rezultate glede poizvedb. Količina predstavlja največji izziv pri starejših strukturah podatkovnih baz, saj potrebuje prilagodljivo in razširljivo podatkovno skladišče ter porazdeljen pristop po povpraševanju [14].

Veliko podjetij že ima veliko količino shranjenih podatkov, ki so ustvarjeni iz različnih virov, kot so besedila, video in avdio posnetki, raziskovalne študije, družbena omrežja, medicina itd.. Predpostavljamo, da je takšna količina podatkov prevelika in neprimerna za obdelavo s klasičnimi relacijskimi podatkovnimi strukturami. Že sama struktura podatkov je lahko neprimerna za shranjevanje, saj so podatki lahko nestrukturirani in jih tako ne moremo normalizirati za shranjevanje na tak način, kot smo ga navajeni z delom z Oracel ali SQL serverji za upravljanje z relacijskimi podatkovnimi bazami [15]. Zato so se pojavile rešitve, kot je Apache Hadoop [16] in podobne rešitve, ki temeljijo na njem. Zanje se odloča čedalje več uporabnikov in se tudi najpogosteje uporabljajo za obdelavo velikih podatkov. Ta odločitev se večinoma izoblikuje glede na vpliv, ki ga ima karakteristika hitrosti. Apache Hadoop je bolj primeren za hitro spreminjajoče in naraščajoče podatkovne množice ter ni odvisen od strukture podatkov. Zmožnost obdelave hitro naraščajočih podatkov, ki imajo različne strukture, ga naredi primernejšega, kot pa tipične rešitve podatkovnih skladišč.

Apache Hadoop je bil razvit kot odprtokodna rešitev podjetja Yahoo in implementira infrastrukturo MapReduce [17], ki ga je razvilo podjetje Google. Omogoča distribucijo oz. razdelitev velikih podatkovnih množic na več serverjev, nad katerimi lahko izvajamo

operacije, kot so podatkovno rudarjenje, strojno učenje, sortiranje …. S tem Hadoop ni samo podatkovno skladišče, ampak deluje tudi kot analitično orodje za podatke [14]. Omogoča cenovno alternativo za shranjevanje in analizo podatkov, pri čem so podatki vedno dostopni za analizo, kadar se enkrat naložijo [18].

Eden izmed največjih uporabnikov Hadoop pristopa je Facebook, ki shranjuje ključne podatke v MySql podatkovnih bazah. Te podatke nato prenese v Hadoop za njihovo analizo, kjer se nad njimi izvajajo določene operacije, kot so kreiranje priporočil, novic za uporabnike, glede na njihove interese, priporočila prijateljev, oglaševanje glede na uporabnikove interese. Po opravljeni analizi rezultate shranijo nazaj v MySql, kjer podatke uporabijo za prikaz uporabniku [14].

2.2.2 Hitrost

Hitrost podatkov (angl. data velocity) je še eden od pomembnejših karakteristik velikih in kompleksnih podatkov. Predstavlja hitrost, s katero podatki prihajajo v organizacijo, se spreminjajo in naredijo podatkovne množice prevelike za njihovo obdelavo. Doba interneta, mobilnih naprav in družbenih omrežij, kot so Facebook, Twitter, LinkedIn je razlog, da v organizacije prihajajo količine podatkov, ki jih je težko obvladovati.

Hitrost podatkov s tem direktno vpliv na količino podatkov, ki sem jo omenil v prejšnjem podpoglavju. Pomembna pa ni samo hitrost, s katero podatki prispejo, se spreminjajo in jih shranjujemo v podatkovna skladišča, ampak je pomembna tudi hitrost povratnih informacij, ki jih pridobimo v času obdelave podatkov, od vnosa podatkov pa vse do sprejetja odločitve [15]. Zmožnost hitrega izkoristka in obdelave podatkov v času prihoda nam daje prednost pred konkurenti in s tem večjo vrednost.

2.2.3 Raznolikost podatkov

Kot sem že omenil, se podatki pojavljajo v različnih strukturah in formatih. V preteklosti smo se osredotočali zgolj na strukturirane podatke, kot so finančni podatki, ki so bili enostavni za shranjevanje v tabelah relacijskih podatkovnih baz. Večina podatkov v dobi »velikih podatkov« pa je nestrukturiranih (avdio, video, tekst, glas itd.) in večinoma niso primerno oblikovani za takojšno procesiranje. S tem se doda dodatna kompleksnost k že tako velikim in hitro naraščajočim podatkom in naredi podatke še bolj neprimerne za shranjevanje v

relacijske podatkovne baze [14]. S tehnologijami, namenjenimi velikim podatkom lahko sedaj enostavno analiziramo in združimo podatke različnih tipov, kot so sporočila, slike, pogovori, podatki iz družbenih omrežij, podatki iz senzorjev itd.. Združevanje teh podatkov, ki so na prvi pogled nezdružljivi, je prednost uporabe Hadoopa, ki podatke združi, obdela ter omogoča njihovo vizualizacijo [19].

Pri tem pa ne smemo zanemariti, da podatki prihajajo iz različnih virov, kot so brskalniki, različni operacijski sistemi, prenosne naprave (mobilni telefoni, tablični računalniki), uporabniški vmesniki, za katerimi so ljudje, zato se ne moremo izogniti napakam v podatkih. To ima tudi direkten vpliv na integriteto oz. verodostojnost podatkov. Z drugimi besedami čim večja je raznolikost podatkov (angl. data variety), tem več napak bodo vsebovali [15].

2.2.4 Verodostojnost podatkov

Z verodostojnostjo (angl. data veracity) mislimo na verodostojnost podatkov oz., kako gotovi smo glede podatkov, če jim lahko zaupamo in so dostopni, kadar jih potrebujemo. Pri tem se osredotočamo predvsem na pomen podatkov, ki ga imajo glede na dan problem, ki ga raziskujemo. Preden so dodali ta kriterij, se je za podatke smatralo, da prihajajo brez napak in točni. Ta predpostavka je sledila iz tradicionalnih podatkovnih skladišč. Ker prihajajo podatki iz različnih virov, kot so objave na Facebooku, Twitterju, LinkedInu, iz pametnih telefonov, raznih senzorjev itd., in so v večini primerov nestrukturirani, se moramo najprej vprašati, če lahko zaupamo takim podatkom. Takšni podatki so lahko zanimivi za samo branje, nimajo pa prave vrednosti za poslovna in kritična odločanja.
Zato je pomembno, da same podatke prečistimo z namenskimi orodji in algoritmi ter definiramo, kako bomo takšnim podatkom zaupali [9]. Če podatkom slepo zaupamo, se pravi jih ne preverimo in očistimo, lahko ob analizi podatkov pridobimo prave vrednosti glede na podatke, naše odločitve pa bodo kljub temu napačne.

2.2.5 Vrednost

Vrednost (angl. data value) je zadnja karakteristika, ki jo omenjam v tej diplomski nalogi, in predstavlja želeno vrednost izhodnih podatkov, ki jih dobimo s procesiranjem velikih podatkov. Če podatki za nas nimajo nobene vrednosti, potem postanejo prej našteti kriteriji nepomembni [20]. Zato si želimo iz velikih podatkovnih množic pridobiti čim večjo vrednost.

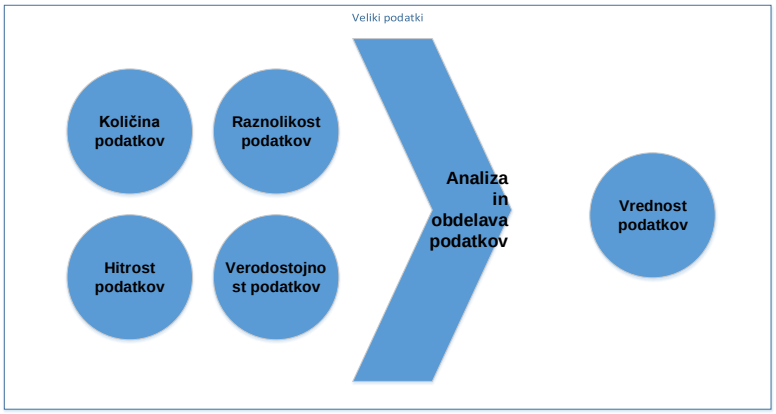
Večja vrednost podatkov predstavlja tudi večji vpliv, ki ga ima podjetje pred konkurenco ter pripomore k boljšemu sprejemanju odločitev.

Kot je prikazano s primerom v 2. poglavju, se lahko vrednost podatkov s časom spreminja. Zato skušajo podjetja z neprestanim zbiranjem in analizo podatkov pridobiti primerno vrednost za trenutne in prihajajoče odločitve.

2.3 Grafični prikaz velikih podatkov

Slika 1 prikazuje karakteristike velikih podatkov, tako kot jih demonstrira večina literature. Se pravi, da so veliki podatki sestavljeni iz količine, raznolikosti, hitrosti, verodostojnosti in vrednosti podatkov. Takšen prikaz, čeprav je pravilen, se mi ne zdi najbolj primeren za opis pojma veliki podatki, zato sem s sliko 2 prikazal velike podatke na način, ki se mi zdi bolj primeren za opis samega pojma. Iz slike 2 je razvidno, da moramo kljub temu, da je vrednost del velikih podatkov, podatke obdelati in analizirati za pridobivanje vrednosti, ki se skriva v njih.

Slika 2.1: Karakteristike velikih podatkov

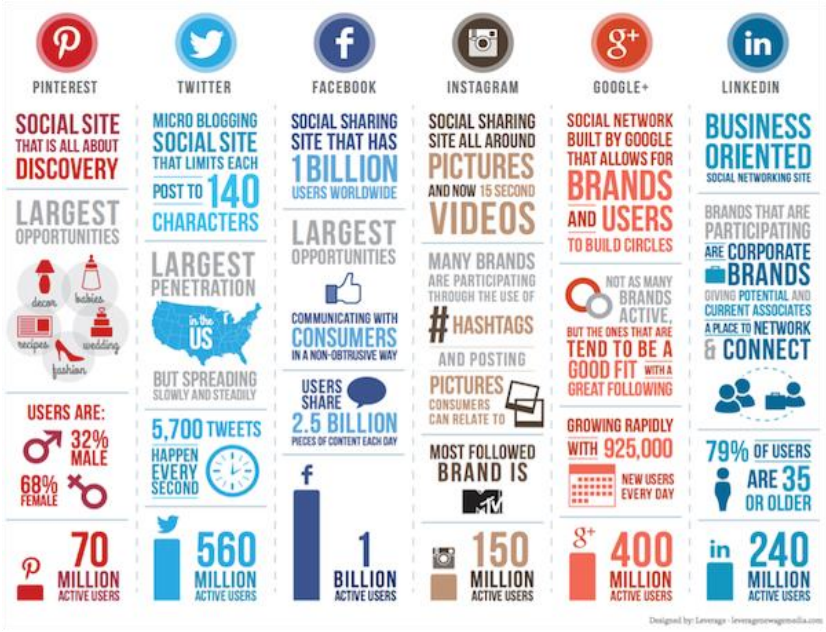


Slika 2.2 Primer primernejšega prikaza velikih podatkov

2.4 Primeri uporabe velikih podatkovnih množic

Kot že omenjeno v uvodu, veliki podatki zajemajo vsa področja našega gospodarstva (medicina, trgovina, politika, industrija, družbena omrežja). V nadaljevanju je navedenih nekaj primerov uporabe velikih podatkov, ki sem jih našel na spletu in v literaturi.

Twitter, LinkedIn in Facebook so primeri podjetij iz domene družbenih omrežij. Ravno družbena omrežja spadajo med tista podjetja, ki v današnjem času ustvarjajo in uporabljajo največje količine podatkov. Iz slike 2.3 je razvidno, da družbena omrežja uporablja velika količina uporabnikov, Facebook celo 1 milijarda, ki dnevno ustvarjajo veliko količino podatkov, kot so slike, posti, twitti itd. Ta podjetja velike podatke koristijo z namenom ponujanja boljših storitev končnim uporabnikom. LinkedIn pridobljene podatke uporablja za storitvi, kot sta »Osebe, ki jih morda poznaš« in »Prikaz novic, ki zanimajo uporabnika«. Pri podjetju Twitter pridobljene množice velikih podatkov uporabljajo za prikaz predlog med iskanje v realnem času in popravljanje slovničnih napak v iskalnih nizih. Facebook uporablja velike podatke za prepoznavanje ljudi na slikah, prikaz strani, ki bi te zanimale, prikaz ciljnih oglasov. Če podrobneje spremljamo oglase, ki se prikazujejo na družbenih omrežjih, predvsem na Facebooku, hitro opazimo, da so prikazani oglasi ravno tisto, kar nas trenutno zanima. Primer, če smo iskali informacije o določenem predmetu, področju (npr. razvoj aplikacij za Android naprave), nam bo Facebook zagotovo prikazal oglase, povezane s to temo.



Slika 2.3: Veliki podatki v družbenih omrežjih
(vir: http://searchenginewatch.com/sew/how-to/2339110/digital-data-trends-search-social-content-fusion)

Iz člankov [21] in [22] je razvidno, da lahko Twitter in ostala družbena omrežja uporabljamo za napovedovanje izbruha epidemij bolezni, kot je gripa, ter za napovedovanje naravnih nesreč (potresi, požar, poplave …). Pridobljene informacije o izbruhih epidemij ali naravnih nesrečah nam koristijo, da se pravočasno pripravimo z njimi.

Trgovine velike podatke, pridobljene iz navad kupcev, uporabljajo za priporočila o nakupu glede na interese uporabnika. Amazon in Walmart sta podjetji, ki ta način uporabljata za ustvarjanje vrednosti podjetja. Vsakič, ko uporabnik išče po spletni strani Amazona, bodisi za nakup izdelka ali za informacijami o knjigi, seriji, Amazon pridobi nekaj informacij o tem, kaj nas zanima in kaj je v tistem času popularno. Te informacije jim omogočajo, da prilagodijo cene izdelkov trenutnim trendom in razmeram ter že predhodno poskrbijo, da imajo vedno dovolj zaloge [8].

3 Klasifikacija in metode strojnega učenja na velikih podatkih

Strojno učenje je veja raziskav umetne inteligence, ki se uporablja za analizo podatkov in odkrivanje zakonitosti med njimi. Osnovni princip strojnega učenja je avtomatsko opisovanje pojavov (grajenje modelov) iz podatkov. Naučeni modeli nam pomagajo pri razlaganju podatkov, pridobivanju vrednosti iz podatkov in napovedovanju.

Omogoča nam analizo podatkov, ki jih zaradi njihove kompleksnosti (različni viri in tipi podatkov) drugače ne bi mogli analizirati. Ravno v teh nedirektnih povezavah med podatki se pokaže moč strojnega učenja [23].

3.1 Klasifikacija (uvrščanje)

Ena najpogostejših uporab strojnega učenja je klasifikacija ali uvrščanje (angl. classification). Naloga učnega algoritma (klasifikatorja) je za objekt (problem), opisan z množico atributov določiti, kateremu izmed možnih razredov pripada. Razredi so v primeru klasifikacije vnaprej definirani na podlagi že znanih atributov. Atributi so neodvisne zvezne spremenljivke, s katerimi opisujemo objekte, razred pa je odvisna diskretna spremenljivka, ki ji določimo vrednost (izberemo razred) glede na vrednosti atributov [knjiga]. Da lahko učni algoritem določi razred, mora na neki način imeti predstavljeno diskretno funkcijo oz. model, ki preslika prostor atributov v razred. Ta funkcija je lahko podana vnaprej ali pa je naučena iz podatkov.

Klasifikacijo podatkov lahko razdelimo v tri dele. Prvi del je grajenje modela. Ta se zgradi iz primerkov učne množice (angl. training set), pri tem zahtevamo, da čim bolj ustreza vhodnim podatkom (atributom). Drugi del je uporaba tako zgrajenega modela za razvrščanje primerkov iz testne množice (angl. test set) v ustrezne razrede. Razvrščene primerke iz testne množice nato uporabimo za ovrednotenje zgrajenega modela. Tretji del pa je uporaba zgrajenega modela na novih podatkih.

Primer uporabe klasifikacije je diagnostika pacientov, kjer na podlagi opisa znanih diagnoz pacientov zgradimo model, ki ga bomo lahko uporabili za diagnostiko novih pacientov [3].

3.1.1 K-kratna navzkrižna validacija

K-kratna navzkrižna validacija (angl. k-fold cross-validation) je ena izmed najpogostejših metod za validacijo modela. Z njo izboljšamo učinkovitost naučenega modela, ki ni odvisen le od učnega algoritma, temveč tudi od nekaterih drugih dejavnikov kot so: porazdelitev razredov, velikost učne in testne množice, podobnost med množicama, obtežitev napačnih klasifikacij.
V splošnem velja, čim večja je učna množica, tem boljši je naučeni model. Ker imajo veliki podatki na voljo veliko število primerkov, za njih v večini primerov ni potrebe po uporabi navzkrižne validacije, ampak lahko iz množice izberemo poljubno število primerkov, ki jih bomo uporabili za učno in testno množico. Pri izvedbi praktičnega dela, kjer bom uporabil male in velike množice podatkov, bom za izboljšanje učinkovitosti modela, zgrajenega na »malih« podatkovnih množicah uporabil metodo navzkrižne validacije.

K-kratna navzkrižna validacija nam omogoča, da v več zaporednih iteracijah razdelimo izvorno množico na učno in testno množico. Pri tem celotno množico razdelimo na k-različnih podmnožic (približno) enake velikosti. V vsaki iteracij uporabimo eno izmed podmnožic za tesno množico, preostalih k-1 podmnožic pa uporabimo za učenje modela. To ponovimo n-krat oz. dokler ne uporabimo vseh k-podmnožic. Skupna napaka naučenega modela se izračuna kot povprečje vseh k-podmnožic. Z uporabo k-navzkrižne validacije prav tako izločimo možnost neprimerne delitve na dve množici.

3.1.2 Mere točnosti klasifikacije

Za merjenje učinkovitosti učnih algoritmov v praktičnem delu bom uporabil dve meri točnosti klasifikacije: klasifikacijska točnost in F-measure.

Klasifikacijska točnost

Uspešnost reševanja klasifikacije ocenjujemo s klasifikacijsko točnostjo (angl. classification accuracy). Klasifikacijska točnost je merilo, ki nam pove, kolikšen delež primerkov je bil pravilno klasificiran in je definirana z:

$$T = \frac{N^{(p)}}{N} \times 100 \tag{3.1}$$

Tu je:
T – točnost (%)
$N^{(p)}$ – število pravilno razvrščenih primerkov
N – število vseh primerkov iz množice

Interpretiramo jo lahko kot verjetnost, da bo naključno izbran primer pravilno klasificiran.

F-measure

F-measure se, prav tako kot klasifikacijska točnost, uporablja za merjenje točnosti klasifikacije. Pri tem pa F-measure upošteva priklic in preciznost. Priklic ocenjuje odstotek pravilno klasificiranih pozitivnih primerov, preciznost pa ocenjuje odstotek primerkov, ki so bili klasificirani kot pozitivni. F-measure si lahko predstavljamo kot tehtno povprečje med priklicem in preciznostjo, pri čem doseže najboljšo vrednost pri 1 in najslabšo vrednost pri 0. Vse tri mere F-measure, priklic in preciznost pri tem ne upoštevajo števila pravilno klasificiranih negativnih primerov, saj nas pri iskanju oz. razvrščanju bolj zanima, kako dobro bodo primerki razvrščeni v pravilne razrede.
Priklic je definiran kot:

$$Priklic = \frac{TP}{TP+FN} = \frac{TP}{POS} \tag{3.2}$$

Tu je:
TP – število pravilno klasificiranih pozitivnih primerov (angl. true positives)
FN – število napačno klasificiranih pozitivnih primerov (angl. false negatives)
POS – število pozitivnih primerov

Preciznost je definirana kot:

$$Preciznost = \frac{TP}{TP+FP} = \frac{TP}{PP} \quad (3.3)$$

Tu je:

TP – število pravilno klasificiranih pozitivnih primerov (angl. true positives)

FP – število napačno klasificiranih negativnih primerov (angl. false positives)

PP – število primerov, klasificiranih kot pozitivni (angl. predicted positives)

F-measure pa je definirana kot:

$$F = \frac{2 \times Priklic \times Preciznost}{Priklic+Preciznost} = \frac{2TP}{2TP+FP+FN} \quad (3.4)$$

Tu je:

F = F-measure

TP – število pravilno klasificiranih pozitivnih primerov (angl. True Positives)

FP – število napačno klasificiranih negativnih primerov (angl. False Positives)

FN – število napačno klasificiranih pozitivnih primerov (angl. False Negatives)

3.2 Učni algoritmi na velikih podatkih

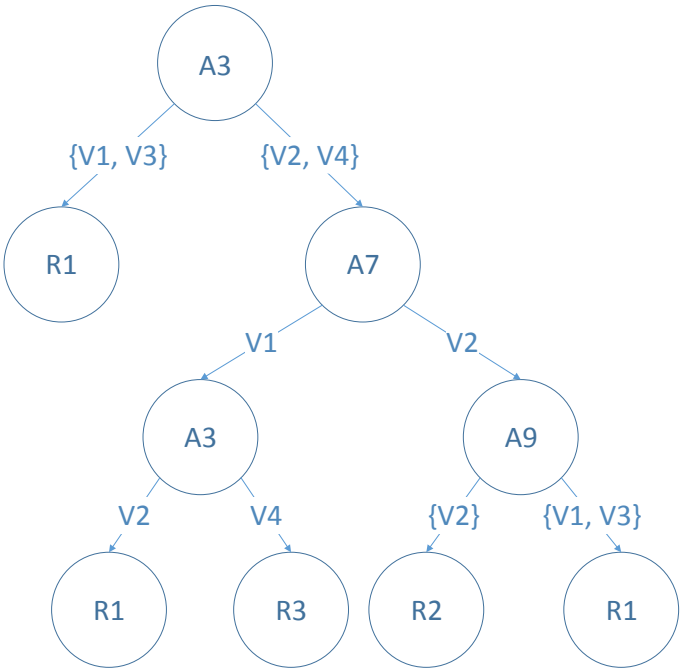
Kot že omenjeno v uvodnem poglavju, obstaja več učnih algoritmov, ki so namenjeni klasifikaciji. V tem poglavju bom predstavil najpogostejše učne algoritme, ki jih bom pozneje uporabil v praktičnem delu diplomske naloge.

3.2.1 Odločitvena drevesa

Odločitvena drevesa so idealna za filtriranje velikih količin podatkov. Prav tako so eden izmed osnovnih načinov klasifikacije in imajo zadovoljivo učinkovitost in točnost klasifikacije na podanih množicah. Imajo dobro razmerje med klasifikacijsko točnostjo in hitrostjo učenja, prav tako pa nam dajo zadovoljive rezultate [24].

Odločitveno drevo se zgradi na podlagi primerkov v učni množici. Pri tem je vsak primerek opisan s pripadajočimi atributi in razredom, v katerega spada. Notranja vozlišča drevesa ustrezajo atributom, listi drevesa pa so pripadajoči razredi [25]. Pri tem ena pot v drevesu ustreza enemu odločitvenemu pravilu za uvrstitev primerka v razred. Vsako vozlišče

predstavlja pogoj za določen atribut, podmnožice pa predstavljajo vrednosti, ki jih pogoj lahko zavzame. Primer odločitvenega drevesa povzetega po [3] je podan na sliki 3.1.



Slika 3.1 Primer odločitvenega drevesa

Algoritem učenja odločitvenega drevesa je naslednji:

Če je izpolnjen ustavitveni pogoj,
potem postavi list, ki vključuje vse učne primere;
sicer
izberi »najboljši« atribut Ai;
označi naslednike z vrednostmi atributa Ai;
za vsako vrednost V_j atributa A_i ponovi:
rekurzivno zgradi poddrevo z ustrezno
podmnožico učnih primerov;

Pri tem je ustavitveni pogoj lahko:

- bodisi dovolj »čista« učna množica (npr. vsi ali večina primerov iz iste množice),
- bodisi je premalo učnih primerov za zanesljivo nadaljevanje gradnje drevesa,
- ali pa je zmanjkalo (dobrih) atributov

Ključnega pomena je izbira »najboljšega atributa«. Za izbiro atributa se najpogosteje uporabljajo mere: informacijski prispevek, razmerje informacijskega prispevka, Gini–indeks in ReliefF, ki nam povejo kakšna je pomembnost atributa [3].

Zaradi nezanesljivosti nižjih nivojev drevesa, ki lahko nastane zaradi manjšega števila učnih primerkov ali pa prevelikega prileganja podatkom (angl. overfitting), se uporabljata dve metodi rezanja drevesa. Z rezanjem naprej (angl. pre-pruning) ali pa z rezanjem nazaj (angl. post-pruning) pridobimo manjše drevo, kar nam da večjo preglednost in razumljivost, običajno pa povečamo še točnost modela na testni množici.

Z rezanjem naprej predčasno ustavimo gradnjo drevesa in s tem drevesu ne dovolimo, da se zgradi do končne velikosti. Rezanje nazaj, ki se uporablja najpogosteje, zgradi drevo do konca, saj ne moremo vnaprej določiti nezanesljivosti drevesa. Po končani gradnji drevesa pa manj zanesljive dele odrežemo.

V praktičnem delu bom testiral učinkovitost dveh učnih algoritmov, ki temeljita na principu odločitvenih dreves: C4.5 (J48 v programu WEKA) in naključni gozdovi (angl. Random Forest). Algoritma bom predstavil v nadaljevanju.

C4.5

C4.5 je algoritem, ki ga v programu WEKA najdemo pod imenom J48, je eden izmed najpogosteje uporabljenih algoritmov, ki temeljijo na odločitvenih drevesih. C4.5 razširja Quinlanov ID3 [26]. Odločitveno drevo se zgradi na podoben način, kot je opisano v prejšnjem poglavju. Značilnost C4.5 algoritma je izboljšana učinkovitost v primerjavi z osnovnim principom odločitvenih dreves.

Naključni gozdovi

Metoda naključni gozdovi (angl. Random Forest) je bila razvita za odločitvena drevesa [27]. Omogoča, da zgradimo zaporedje odločitvenih dreves, tako da se pri izbiri najboljšega atributa v vsakem vozlišču naključno izbere relativno majhno število atributov. Število tako zgrajenih dreves je po navadi 100.

Vsako drevo, ki je zgrajeno na učni množici, se kasneje uporabi za klasifikacijo novega primera po metodi glasovanja. Vsako drevo ima en glas, ki ga nameni razredu, v katerega bi to drevo klasificiralo primerek. Na koncu se izbere razred, ki je dobil največ glasov.
S tem se poveča točnost oz. učinkovitost modela v primerjavi z enim drevesom.

V praktičnem delu diplomske naloge bom uporabil implementacijo naključnih gozdov programa WEKA.

3.2.2 AdaBoost.M1

AdaBoost.M1 [28] je Freundova in Schapirova implementacija algoritma AdaBoost [29], ki teoretično lahko zmanjša možnost napake vsakega učnega algoritma, katerega učinkovitost je boljša kot naključno ugibanje. Značilnost algoritma AdaBoost.M1 je, da uporablja metodo »boosting«, ki bistveno zmanjša napake vsakega »šibkega« učnega algoritma. Šibki učni algoritmi so v tem primeru lahko Naivni Bayes, odločitvena drevesa itd. [30].

Shapire je zapisal delovanje algoritma »boosting« kot učenje modela, ki izvede šibki[1] algoritem večkrat oz. v več iteracijah, pri tem uporabi vsakič drugo podskupino učne množice. V vsaki iteraciji se zgradi šibko pravilo oz. model, ki se nato z drugimi modeli združi v končni model, ki je bolj točen, kot posamezno šibko pravilo. Končni model je sestavljen na podlagi glasovanja šibkih pravil, se pravi, da klasifikacija v razrede poteka glede na razred, ki je dobil največ glasov oz. je bil največkrat izbran med šibkimi pravili [30].

Za izvedbo praktičnega dela bom uporabili AdaBoost.M1 s šibkima algoritmoma J48 in DecisionStump.

3.2.3 Metoda podpornih vektorjev

Metoda podpornih vektorjev (angl. Support Vector Machines) ali SVM [31] spada med najbolj uspešne metode za klasifikacijo. Model SVM je predstavitev primerkov iz učnih množic v prostoru. Algoritem nato poišče optimalno hiperravnino, ki loči primerke med sabo. Pri tem si želimo, da je prostor med ravninama čim večji. Vektorjem oz. primerkom, ki so

[1] Pojem »šibki« uporabljamo kljub temu, da se »boosting« lahko kombinira s precej močnim oz. učinkovitim učnim algoritmom, kot je npr. C4.5.

najbližji primerkom optimalne hiperravnine, pravimo podporni vektorji. Klasifikacija novega primerka poteka glede na to, na kam spada nov primerek. V nekaterih primerih pa lahko nelinearna ravnina bolje določa klasifikacijo primerkov. Pri tem metode SVM, poleg uporabe podpornih vektorjev, uporabljajo jedrne funkcije, ki nelinearno transformira atributni prostor v kompleksnejši atributni prostor. S tem postane prostor primernejši za linearno ločitev hiperravnine.

Za izvajanje praktičnega dela bom uporabil algoritem SMO, ki je implementacija metode SVM, z linearnim jedrom (angl. linear kernal) v orodju WEKA.

3.2.4 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator temelji na Bayesovem pravilu s predpostavkami o pogojni neodvisnosti vrednosti različnih atributov pri danem razredu. Grajenje modela je enostavno, brez zapletenih ponavljajočih ocenjevanj parametrov, kar ga naredi primernega za velike podatke. Kljub svoji enostavnosti, nam klasifikacija z Naivnim Bayesovim da dobre rezultate, pri tem pa je njegovo izvajanje velikokrat hitrejše, kot pa izvajanje drugih algoritmov [32].

Za klasifikacijo primerka v določen razred se za nov primerek izračuna njegova pogojna verjetnost glede na podane atribute. Pogojna verjetnost, kot jo je izpeljal [3] se izračuna z:

$$P(r_k|V) = P(r_k) \prod_{i=1}^{\alpha} \left(\frac{P(r_k|v_i)}{P(r_k)}\right) \quad (3.5)$$

Tu je:
$P(r_k|V)$ – verjetnost, da bo nov primerek na podlagi V klasificiran kot r_k
$P(r_k)$ – verjetnost razreda r_k
$P(r_k|v_i)$ – pogojna verjetnost razreda r_k pri danem atributu v_i

Klasifikacija novega primerka poteka z izračunom pogojnih verjetnosti za vsak razred posebej na podlagi podanih atributov. Novemu primerku se dodeli razred, ki je imel največjo pogojno verjetnost.

3.2.5 K-najbližjih sosedov

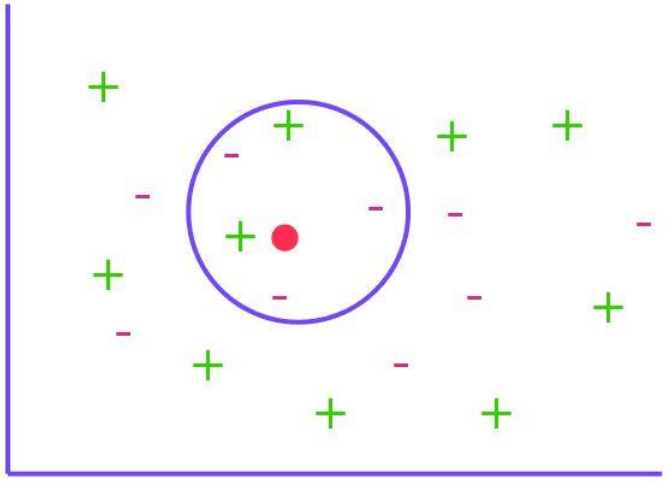
K-najbližjih sosedov (kNN) je numerična metoda klasifikacije. Metoda je zelo preprosta, saj enostavno shrani vse ali pa podmnožice učnih primerov. Pri klasifikaciji novih primerov poiščemo med učnimi primerki k-najbližje, pri tem je k-število učnih primerkov, s katerimi primerjamo nov primerek, in pri klasifikaciji napovemo večinski razred, t. j. razred, ki mu pripada največ izmed k-najbližjih sosedov. Za število k se po navadi uporabi liho število, npr. 1, 3, 7, 15 ali 31. Ker učenja pri tej metodi skoraj ni, pravimo tej vrsti učenja tudi leno učenje (angl. lazy learning) [3].

Kot je zapisal Sutton v svojem delu [33], si lahko primerke iz učne množice predstavljamo kot točke v prostoru. Pri tem si podobnosti med primerki predstavljamo kot razdalje med njimi. Za računanje razdalje se po navadi uporabi evklidska razdalja. Algoritem delovanja je povzel iz del [34] in [35] in ga lahko zapišemo kot:

1. Izbran je nov primerek in število k (število najbližjih sosedov).
2. Izberemo k-primerkov iz učne množice, ki so najbližje novemu primerku.
3. Poiščemo večinski razred.
4. Nov primerek klasificiramo kot večinski razred.

Enostaven primer klasifikacije prikazuje slika 3.2. Pri izbiri 1-najbližjega soseda bi nov primerek klasificirali kot +, pri izbiri 2-najbližjih sosedov je klasifikacija neznana, zato se v večini primerov uporablja liho število. Pri izbiri 5-najbližjih sosedov bi primerek klasificirali kot -.

V praktičnem delu smo uporabili algoritem IBk orodja WEKA, ki je implementacija metode kNN.



Slika 3.2: primer kNN klasifikacij

4 PRAKTIČNI DEL

Kot že omenjeno v uvodnem poglavju, je osnovni namen diplomske naloge analizirati učinkovitost učnih algoritmov na velikih podatkih. V tem poglavju se bom posvetil analizi učnih algoritmov, ki sem jih opisal v prejšnjem poglavju (odločitvena drevesa, AdaBoost ...) ter poskušal ugotoviti, kateri izmed algoritmov je za podan problem na različnih podatkovnih množicah najučinkovitejši. S tem bom dobil algoritme, ki bodo izmed izbranih algoritmov najučinkovitejši, za uporabo na velikih podatkih. Podatkovne množice sem pridobil iz UCI [4] in mldata [5] repozitorija, učinkovitost izvajanja algoritmov pa sem izvedel s pomočjo orodja WEKA [6].

4.1 Izvedba eksperimenta

Za izvedbo eksperimenta sem uporabil prenosni računalnik ASUS K-53S, z Intel Core i5-2410M 2.30 GHz procesorjem, 8 GB in nameščenim 64 bitnim operacijskim sistemom Windows 8.1 Pro. Za izvajanje meritev in uporabo učnih algoritmov sem uporabil orodje WEKA 3.6.

WEKA je odprtokodno programsko okolje, ki omogoča uporabo orodij in algoritmov za podatkovno rudarjenje in modelirano napovedovanje. Algoritme lahko izvajamo direktno nad podatkovnimi datotekami ali pa s pomočjo lastne Java kode. Prav tako omogoča standardna opravila na področju podatkovnega rudarjenja, kot so predobdelava, klasifikacija, regresija, vizualizacija podatkov.

Iz dostopnih množic, namenjenih klasifikaciji sem izbral 7 velikih podatkovnih množic. Množice sem izbral na podlagi treh kriterijev: števila primerkov v množici, števila atributov in števila razredov za razvrščanje, pri tem sem se osredotočil na podatkovne množice, ki so imele največje vrednosti pri posameznem kriteriju. Z upoštevanjem teh kriterijev sem hotel doseči, da bodo množice kar najbolj podobne velikim podatkom. Prav tako pa sem izbrali še 7 manjših podatkovnih množic, s katerimi sem ugotavljal, kako dobro so

posamezni učni algoritmi skalabilni tudi na več primerkov ali atributov in ali so algoritmi, ki so hitri/točni na manjših množicah, enako učinkoviti tudi na velikih množicah. Preverjanje učinkovitosti algoritmov na malih podatkih je za velike podatke ravno tako pomembno, kot učinkovitost na velikih podatkovnih množicah, saj lahko z novejšimi pristopi, kot je že omenjeni Apache Hadoop, velike podatkovne množice analiziramo na večjem številu računalnikov, se pravi, da velike podatkovne množice razdelimo na več manjših, ki pa s časom lahko postanejo spet velike podatkovne množice. V ta namen sem preverjal skalabilnost učnih algoritmov.

Značilnosti podatkovnih množic so navedene v tabelah 4.1 in 4.2. Iz tabele je razvidno, da sem za izvedbo eksperimenta uporabil tudi podatkovne množice, ki vsebuje več kot 100.000 primerkov (census-income, diabetic_data, kddcup99) in množici, ki vsebujeta več kot 1.000 atributov (cmu-newsgroup-clean-1000_sanitized, dlbcl-tumor-from-harvard).
Podatkovne množice so dostopne na spletni strani http://archive.ics.uci.edu/ml/ in http://mldata.org/repository/.

Tabela 4.1: Značilnosti velikih podatkovnih množic

Ime podatkovne množice	**št. primerkov**	**št. atributov**	**št. razredov**
arrhythmia	452	280	16
census-income	299.285	42	2
cmu-newsgroup-clean-1000_sanitized	19.747	21.632	20
diabetic_data	101.766	50	3
dlbcl-tumor-from-harvard	77	7.130	2
kddcup99	494.021	42	23
letter	20.000	17	26

Tabela 4.2: Značilnosti manjših podatkovnih množic

Ime podatkovne množice	**št. primerkov**	**št. atributov**	**št. razredov**
audiology	226	70	24
kdd_synthetic_control	600	62	6
mushroom	81.214	23	2
nursery	12.960	9	5
sonar	208	61	3
splice	3.190	62	3
waveform-5000	5.000	41	3

Nad podatkovnimi množicami sem z orodjem WEKA izvedel učne algoritme namenjene klasifikaciji, ki sem jih opisal v poglavju 3.2. Za validacijo naučenega modela pri manjših podatkovnih množicah in množicah arrhythmia, dlbcl-tumor-from-harvard, letter sem uporabili metodo desetkratne navzkrižne validacije. Se pravi, da sem pred vsakim izvajanjem algoritma izbral možnost »Cross-validation«, število foldov pa sem nastavil na 10. Pri množicah census-income, cmu-newsgroup-clean-1000_sanitized, diabetic_data, kddcup99 sem zaradi velikega števila primerkov in atributov za validacijo naučenega modela uporabil metodo razdelitve podatkov (angl. percantage split), s pomočjo katere seo za učno množico izbral naključnih 66 % primerkov, za testno množico pa preostalih 34 % primerkov. Za uporabo razdelitve podatkov sem se odločil na podlagi rezultatov iz tabele 4.17, iz katere je razvidno, da sem z uporabo te metode izboljšal čas izvajanj algoritmov, pri čem je točnost ostala skoraj enaka.

Zaradi velikih množic, ki sem jih uporabil za eksperiment, sem moral orodju WEKA dovoliti, da za izvajanje programa lahko uporabi več pomnilnika. Vrednost pomnilnika, ki ga lahko uporabi sem nastavil iz privzete vrednosti 1024 MB oz. 1 GB na 8192 MB oz. 8 GB.

4.2 Meritve in analiza podatkov

V tabeli 4.3 je prikazan potreben čas, ki so ga izbrani algoritmi potrebovali za učenje modela na manjših podatkovnih množicah. Iz tabele je razvidno, da je za učenje najmanj časa potreboval algoritem IBk, ki je v povprečju porabil 0,01 sekunde. Kot pa sem že omenil, pri IBk ne gre za učenje, ampak se vsi podatki iz učne množice naložijo v pomnilnik računalnika in se nato uporabljajo za določanje najbližjega soseda. Drugi najboljši čas je dosegel algoritem NaiveBayes, ki je za učenje modela v povprečju potreboval 0,02 sekunde. Največ časa je potreboval algoritem SMO, ki je za učenje modela v povprečju potreboval 6,14 sekunde. Prav tako je dvakrat dosegel najslabši čas, in sicer na podatkovni množici nursery in splice.

Tabela 4.3: Čas v sekundah, potreben za učenje/gradnjo modela na manjših podatkovnih zbirkah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (Decision stump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
audiology	0	0,16	0	0,19	0,7	0	0
kdd_synthetic_control	0,2	0,76	0,04	1,05	0,65	0,04	0
mushroom	0,09	0,81	0,43	0,11	1,97	0,02	0,02
nursery	0,05	1,22	0,09	0,94	19,51	0,02	0,02
sonar	0,02	0,15	0,05	0,42	0,01	0	0
splice	0,11	4,09	0,23	0,33	19,66	0	0
waveform-5000	0,61	6,2	0,31	7,87	0,48	0,05	0
Povprečje:	**0,15**	**1,91**	**0,16**	**1,56**	**6,14**	**0,02**	**0,01**

Algoritmi so v povprečju najmanj časa porabili na podatkovni množici sonar, in sicer 0,09 sekunde, kar je razvidno iz tabele 4.4, ki prikazuje povprečni čas, potreben za analizo posamezne manjše podatkovne množice v sekundah. Največ časa so porabili na podatkovnih množicah nursery, splice, waveform-5000.

Tabela 4.4: Povprečni čas za učenja modela na posameznih podatkovnih množicah v sekundah

Ime podatkovne množice	**Povprečni čas na podatkovno množico**
audiology	0,15
kdd_synthetic_control	0,39
mushroom	0,49
nursery	3,12
sonar	0,09
splice	3,49
waveform-5000	2,22

Tabela 4.5 nam prikazuje čas, potreben za učenje modela na velikih podatkovnih množicah. V njej najdemo tudi polja, ki nimajo vrednosti, v teh primerih sem prekinil izvajanje algoritma, saj v več kot 12 urah nisem pridobil meritve. Glede na ta čas sem se odločil, da ti algoritmi niso dovolj učinkoviti za izbrano podatkovno množico, kljub temu, da bi ob koncu izvajanja morda dosegli malo boljšo klasifikacijsko točnost. Najboljši rezultat je v povprečju dosegel

IBk z zgolj 0,15 sekunde. Drugi najboljši čas je dosegel NaiveBayes z 42,42 sekundami. Najslabši čas pa je dosegel algoritem AdaBoost.M1 s šibkim algoritmom J48, ki je za učenje potreboval 379,77 sekunde. Pri izračunu povprečnega časa za učenje modela pa nisem upošteval časov algoritmov J48, RandomForest in AdaBoost.M1 na podatkovni množici cmu-newsgroup-clean-1000_sanitized, ter algoritma SMO na podatkovni množici census-income in diabetic_data, saj so omenjeni algoritmi potrebovali za učenje več kot 12 ur. Če primerjam rezultate velikih in manjših podatkovnih množic je jasno razvidno, da so algoritmi za učenje na množicah z več primerki ali atributi v povprečju potrebovali več časa. Vrstni red glede na čas, potreben za učenje pa je ostal skoraj nespremenjen, le SMO je v primeru velikih podatkov dosegel boljši rezultat kot pa AdaBoost.M1 (J48). Iz primera podatkovne množice cmu-newsgroup-clean-1000_sanitized, ki ima zelo veliko število atributov (21.632), lahko sklepamo, da je učinkovitost algoritmov J48, AdaBoost.M1(J48) in Random Forest zelo odvisna tudi od števila atributov, in ne le od števila primerkov.

Tabela 4.5: Čas v sekundah, potreben za učenje/gradnjo modela na večjih podatkovnih množicah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (Decision stump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
arrhythmia	0,61	0,98	0,17	3,99	0,84	0,05	0
census-income	70,08	259,13	44,83	897,67	/	0,99	0,14
cmu-newsgroup-clean-1000_sanitized	/	/	402,39	/	466,77	290,54	0,60
diabetic_data	40,21	82,47	5,42	370,00	/	0,49	0,09
dlbcl-tumor-from-harvard	0,3	0,44	1,47	0,51	0,09	0,08	0
kddcup99	94,27	661,93	72,81	984,84	548,06	4,64	0,19
letter	3,61	18,17	0,17	21,58	7,45	0,13	0
Povprečje:	**34,85**	**170,52**	**75,32**	**379,77**	**204,64**	**42,42**	**0,15**

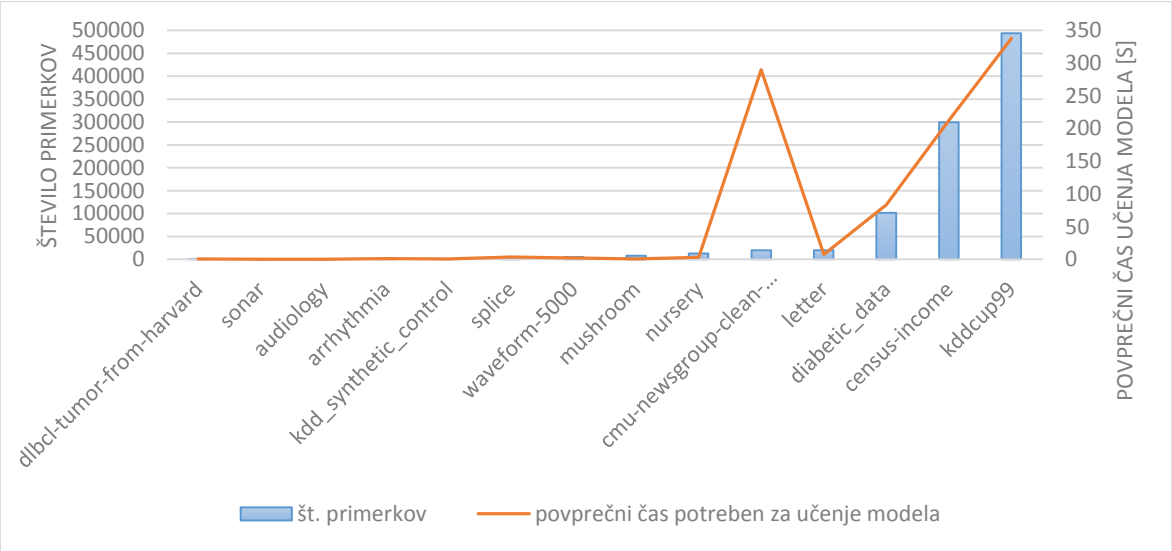
Iz rezultatov v tabeli 4.6, kjer so prikazani povprečni časi, ki so jih algoritmi potrebovali za učenje modela, lahko sklepam, da učni algoritmi za učenje modela iz velikih količin podatkov potrebujejo tudi več časa. Časovna razlika med podatkovnima množicama z

najmanj primerki (sonar) in največ primerki (kddcup99), je kar 338,02 sekunde, kar je več kot 5 minut razlike med učenjem modela.

Tabela 4.6: Povprečni čas za učenja modela na posameznih podatkovnih množicah v sekundah

Ime podatkovne množice	**Povprečni čas na podatkovno množico**
arrhythmia	0,95
census-income	212,14
cmu-newsgroup-clean-1000_sanitized	290,08
diabetic_data	83,11
dlbcl-tumor-from-harvard	0,41
kddcup99	338,11
letter	7,30

Slika 4.1 nam prikazuje odvisnost povprečnega časa učenja od števila primerkov v podatkovni množici. Iz slike je razvidno, da se je čas povečeval s številom primerkov. Razlika med povprečnim časom se je začela zelo povečevati, ko so množice dosegle več kot 100.000 primerkov. Prav tako se iz slike vidi, da je bilo za množico z največ atributi potrebno veliko časa za učenje, kljub temu, da je vsebovala malo manj kot 20.000 primerkov.



Slika 4.1: Graf odvisnosti povprečnega časa učenja od števila primerkov

V tabeli 4.7 so podatki o klasifikacijski točnosti naučenih modelov za manjše podatkovne množice. Kot sem že omenil v prejšnjem poglavju, nam klasifikacijska točnost pove odstotek pravilno razvrščenih primerkov v razrede glede na število vseh primerkov v podatkovni množici.

Pri klasifikaciji malih podatkov je bil najučinkovitejši algoritem RandomForest, ki je dosegel 91,46 % točnost. Drugi najboljši je bi algoritem AdaBoost.M1 (J48) z 90,24 %, tretji pa SMO z 90,03 % klasifikacijske točnosti. Dobljeni rezultati nam kažejo na zelo dobro klasifikacijo izbranih algoritmov, saj so vsi razen AdaBoost.M1 (DecisionStump) dosegli povprečno klasifikacijsko točnost, večjo od 85 %. AdaBoost.M1 (DecisionStump) je v povprečju dosegel najslabše rezultate z zgolj 66,75 % klasifikacijske točnosti. Nizke točnosti v primerih podatkovnih množic audiology in kdd_synthetic_control je dobil, ker algoritem ni mogel nad podatkovnima množicama izvesti metode »boosting«, zato se je izvedel zgolj šibki algoritem. To nam nakazuje na to, da je pri analizi podatkov pomembno poznavanje podatkovne množice, s katerimi imamo opravka, ter pravilno izbiranje algoritma, ki ga bomo uporabili. Saj zgolj s spremembo šibkega algoritma v primeru AdaBoost.M1 lahko zelo izboljšamo učinkovitost klasifikacije na podatkovni množici. Če ne upoštevamo vrednosti, kjer se ni izvedla metoda »boosting« za algoritem AdaBoost.M1 (DecisionStump), bi ta dosegel 77,49 % klasifikacijsko točnost.

Tabela 4.7: Rezultati klasifikacijske točnosti na manjših podatkovnih množicah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
audiology	77,88	79,20	46,46	84,96	81,86	73,45	77,88
kdd_synthetic_ control	91,67	97,83	33,33	95,67	99,17	94,67	96,50
mushroom	100	100	96,20	100	100	95,83	100
nursery	97,05	99,07	66,25	99,51	93,08	90,32	98,38
sonar	71,15	85,10	71,63	77,88	75,96	67,79	86,54
splice	94,08	93,86	86,74	93,17	93,45	95,30	74,67
waveform-5000	75,08	85,16	66,64	80,48	86,68	80,00	73,62
Povprečje:	**86,70**	**91,46**	**66,75**	**90,24**	**90,03**	**85,34**	**86,80**

V povprečju so algoritmi na manjših podatkovnih množicah dosegli najboljše rezultate klasifikacijske točnosti na podatkovni množici mushroom in to s kar 98,86 %. Vsi izbrani algoritmi so na tej podatkovni množici dosegli rezultate boljše kot 95 %. Algoritmi J48, RandomForest, Adaboost.M1 (J48), SMO in IBk so klasificirali primerke 100 %. Algoritmi so dobre rezultate dosegli še na množici nursery z 91,95 % in splice z 90,18 %. Najslabšo klasifikacijsko točnost so v povprečju imeli na podatkovni množici audiology, kjer so dosegli zgolj 74,53 %. Rezultati povprečne klasifikacijske točnosti na posamezni podatkovni množici so vidni v tabeli 4.8.

Tabela 4.8: Povprečna klasifikacijska točnost na manjših podatkovnih množicah

Ime podatkovne množice	**povprečna klasifikacijska točnost na podatkovno množico**
audiology	74,53
kdd_synthetic_control	86,98
mushroom	98,86
nursery	91,95
sonar	76,58
splice	90,18
waveform-5000	78,24

Iz tabele 4.9, kjer so prikazani rezultati klasifikacijske točnosti algoritmov na velikih podatkovnih množicah, je razvidno, da je bil najučinkovitejši algoritem SMO s 85,34 % klasifikacijsko točnostjo. Prav tako sta klasifikacijsko točnost večjo kot 83 % dosegla algoritma RandomForest s 83,45 % in AdaBoost.M1 (J48) z 81,84 %. Najslabšo klasifikacijsko točnost je imel algoritem Adaboost.M1 (DecisionStump) z 58,38 %, ki je najslabšo vrednost dosegel tudi med manjšimi podatkovnimi množicami. Razlog za to je isti kot prej, saj algoritem na podatkovnih množicah cmu-newsgroup-clean-1000_sanitized in letter ni mogel izvesti metode »boosting«.

Tabela 4.9: Rezultati klasifikacijske točnosti na velikih podatkovnih množicah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost. M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
arrhythmia	64,38	69,03	55,53	70,58	70,13	62,39	52,88
census-income	95,30	95,15	93,83	94,96	/	85,82	92,57
cmu-newsgroup-clean-1000_sanitized	/	/	7,33	/	78,22	58,28	40,69
diabetic_data	56,90	54,51	56,10	52,09	/	56,75	45,88
dlbcl-tumor-from-harvard	72,73	85,71	90,91	77,92	96,10	80,52	84,42
kddcup99	99,95	99,97	97,85	99,97	99,92	93,16	99,95
letter	87,98	96,30	7,09	95,54	82,34	64,12	96,03
Povprečje:	**79,54**	**83,45**	**58,38**	**81,84**	**85,34**	**71,58**	**73,20**

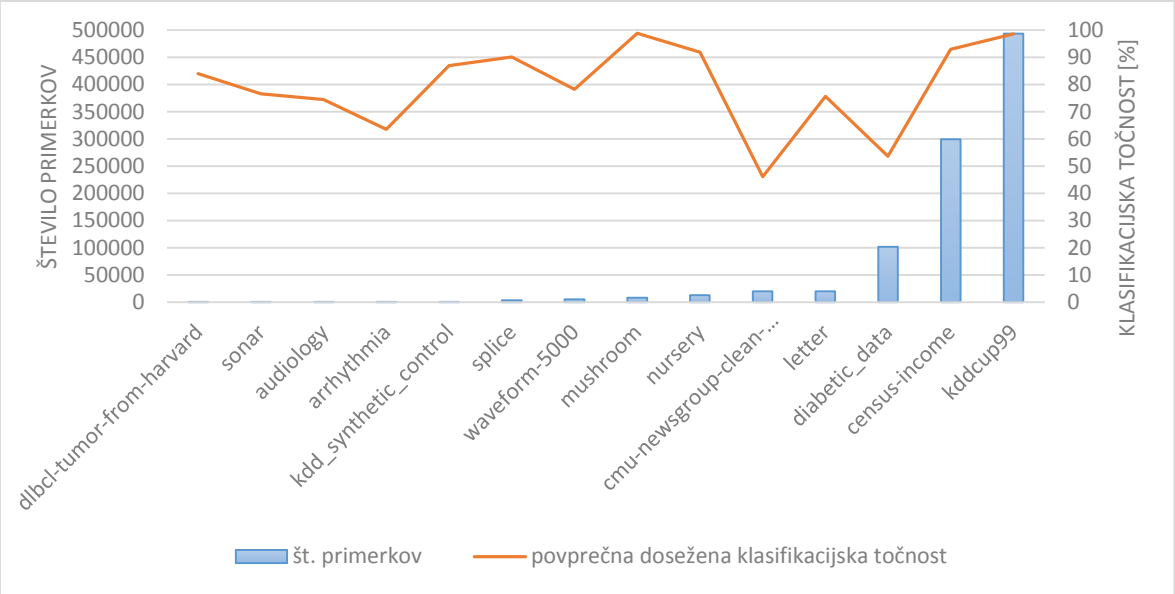
V povprečju so algoritmi na velikih podatkovnih množicah dosegli najboljšo klasifikacijsko točnost na podatkovni množici kddcup z 98,68 %, kar lahko razberemo iz tabele 4.10. Klasifikacijsko točnost, večjo od 90 % so dosegli še na podatkovni množici census-income z 92,94 %. Slabe rezultate pa so dosegli na podatkovni množici diabetic_data, kjer so algoritmi dosegli zgolj okrog 50 % klasifikacijske točnosti.

Prav tako sem slabe rezultate dobil v primeru podatkovne množice cmu-newsgroup-clean-1000_sanitized, kjer se je povprečje zmanjšalo predvsem zaradi algoritma AdaBoost.M1 (DecisionStump), ker ni mogel izvesti metode »boosting«. Na tej množici s tremi algoritmi nismo pridobili vrednosti, saj se izvajanje v primeru algoritmov J48 in AdaBoost.M1 (J48) ni zaključilo po več kot 12 urah. Po rezultatih, ki sta jih dosegla algoritma na manjših podatkovnih množicah, lahko predpostavljam, da bi algoritma dosegla podobne rezultate kot preostali algoritmi. V primeru RandomForest pa se algoritem ni mogel izvesti zaradi premajhne količine pomnilnika, ki ga je imel računalnik na voljo. Delovanje algoritma RandomForest na podatkovni množici diabetic_data, sem moral omejiti na zgolj 8 dreves, ki jih je algoritem lahko uporabil za izvajanje, saj analize drugače nisem mogel izvesti.

Tabela 4.10: Povprečna klasifikacijska točnost na velikih podatkovnih množicah

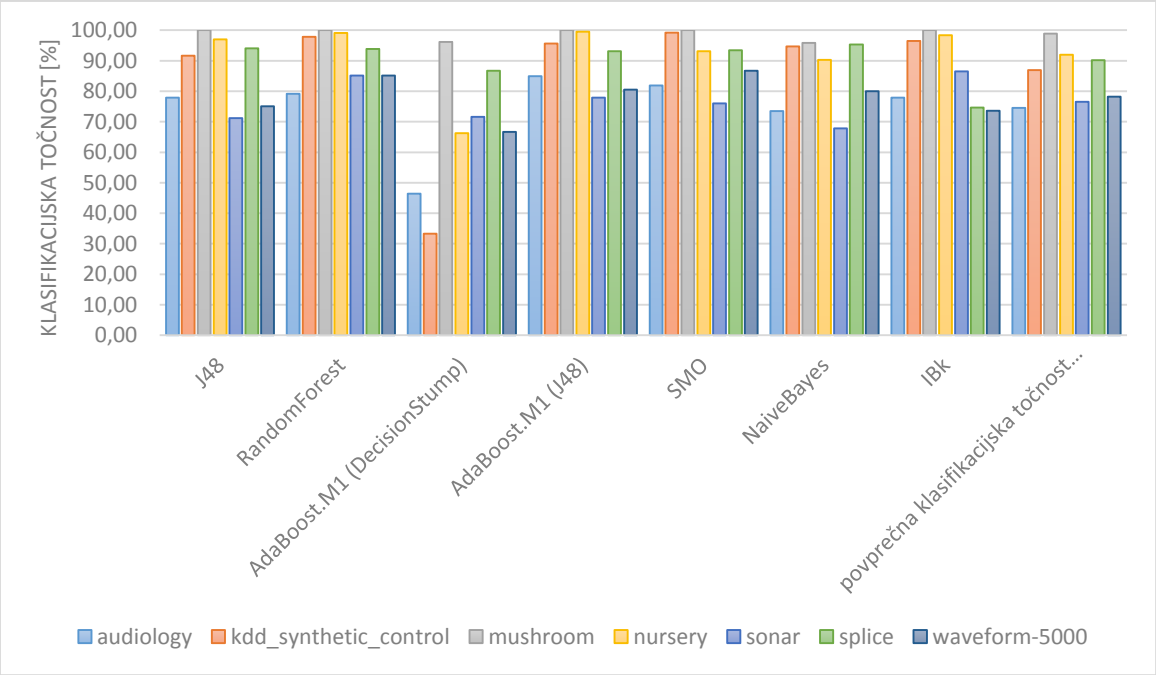
Ime podatkovne množice	**povprečna klasifikacijska točnost na posamezno podatkovno množico**
arrhythmia	63,56
census-income	92,94
cmu-newsgroup-clean-1000_sanitized	46,13
diabetic_data	53,71
dlbcl-tumor-from-harvard	84,04
kddcup99	98,68
letter	75,63

Slika 4.2 nam prikazuje povprečno klasifikacijsko točnost, doseženo na izbranih podatkovnih množicah v odvisnosti od števila primerkov v njih. Kot je razvidno iz grafa, v splošnem lahko predpostavljamo, da veliko število primerkov oz. podatkov omogoča boljše učenje modela in s tem doseganje boljše klasifikacijske točnosti, kar je razvidno iz rezultatov, pridobljenih na množici census_insome in kddcup99. Kljub temu pa smo najboljšo klasifikacijsko točnost dosegli na množici mushroom z zgolj 8124 primerki, na množici diabetic_data, ki ima več kot 10.000 primerkov, pa smo kljub velikem številu primerkov dosegli slab rezultat. Iz tega lahko sklepamo, da za doseganje velike klasifikacijske točnosti ni vedno pogoj veliko število podatkov, v primeru primerkov v podatkovnih množicah, ampak je veliko odvisno tudi od samih podatkov, ki jih klasificiramo. To se vidi predvsem iz tabele 4.7 in 4.9, kjer so vsi algoritmi na podatkovni množici mushroom dosegli najboljšo klasifikacijsko točnost in na podatkovni množici diabetic_data, kjer noben izmed izbranih algoritmov ni dosegel več kot 60 % klasifikacijske točnosti.



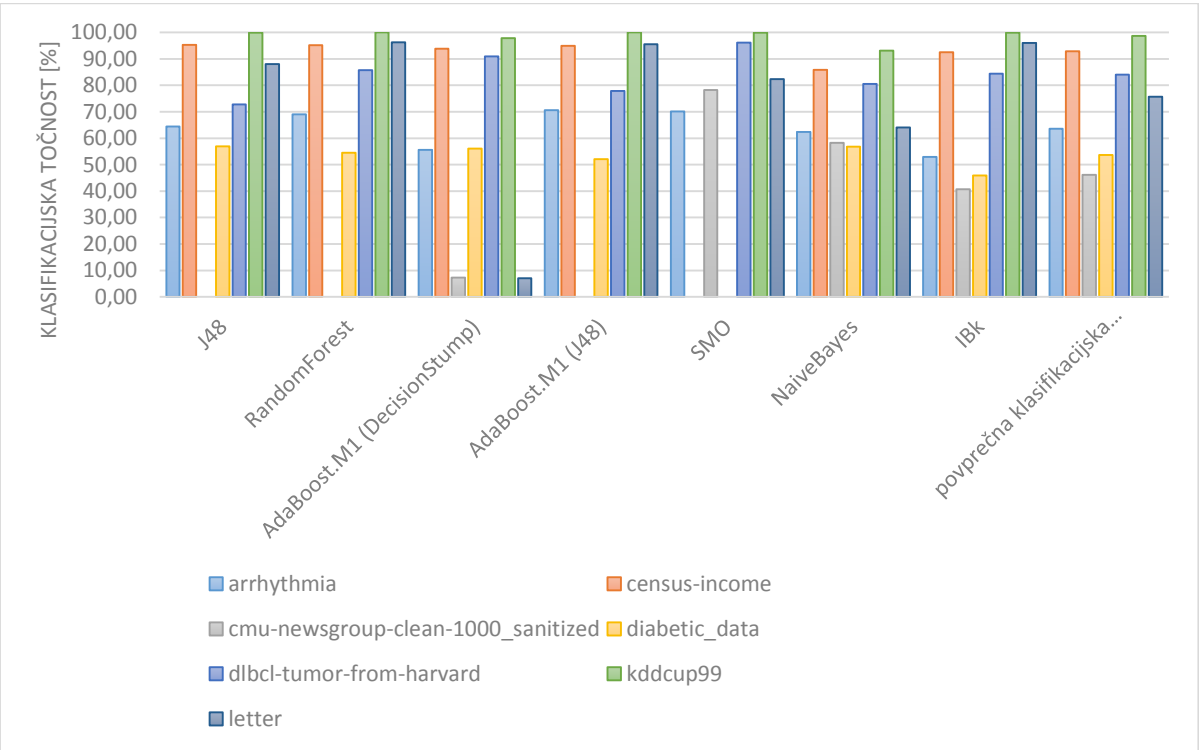
Slika 4.2: Povprečna klasifikacijska točnost v odvisnosti od števila primerkov

Na sliki 4.3 so v obliki grafa prikazani rezultati klasifikacijske točnosti na manjših podatkovnih množicah. Iz slike se jasno vidi, da sta algoritma RandomForest in AdaBoost.M1 (J48) v splošnem dosegla zelo dobre rezultate, brez velikih nihanj med točnostjo na posameznih množicah. S tega lahko sklepamo, da je njuno učenje iz učnih množic bilo najboljše oz. sta se algoritma uspela najbolje prilagoditi različnim množicam podatkov, kar pride v poštev predvsem kadar imamo opravka z velikimi podatki, kjer v naprej ne vemo kakšne podatke lahko pričakujemo. Na različne podatkovne množice se je najslabše prilagodil algoritem AdaBoost.M1 (DecisionStump), kar kažejo velika nihanja med točnostjo posameznih podatkovnih množic.



Slika 4.3: Klasifikacijska točnost na manjših podatkovnih množica

Slika 4.4 nam prikazuje rezultate klasifikacijske točnosti v obliki grafa za velike množice. Iz slike je razvidno, da so algoritmi na posameznih velikih podatkovnih množicah dosegli podobne rezultate, kar velja tudi za manjše podatkovne množice.



Slika 4.4: Klasifikacijska točnost na velikih podatkovnih množicah

Tabeli 4.11 in 4.12 nam prikazujeta uravnotežene rezultate meritve F-measure, kar pomeni, da sem v tabeli zapisal zgolj povprečne vrednosti F-measure, ki jih je algoritem imel za posamezen razred pri razvrščanju. Iz tabel 4.11 in 4.12 je razvidno, da so rezultati primerljivi z rezultati o klasifikacijski točnosti v tabelah 4.7 in 4.9.

Tabela 4.11: Rezultati F-measure na malih podatkovnih množicah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
audiology	0,75	0,76	0,31	0,83	0,81	0,68	0,76
kdd_synthetic_control	0,92	0,98	0,17	0,96	0,99	0,95	0,97
mushroom	1,00	1,00	0,96	1,00	1,00	0,96	1,00
nursery	0,97	0,99	0,55	1,00	0,93	0,85	0,98
sonar	0,71	0,85	0,72	0,78	0,76	0,67	0,87
splice	0,94	0,94	0,87	0,93	0,94	0,95	0,75
waveform-5000	0,75	0,85	0,63	0,81	0,87	0,79	0,74
Povprečje:	**0,86**	**0,91**	**0,60**	**0,90**	**0,90**	**0,83**	**0,86**

Tabela 4.12: Rezultati F-measure na velikih podatkovnih množicah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
arrhythmia	0,63	0,62	0,45	0,69	0,66	0,62	0,51
census-income	0,95	0,94	0,93	0,95	/	0,89	0,93
cmu-newsgroup-clean-1000_sanitized	/	/	0,03	/	0,78	0,59	0,44
diabetic_data	0,51	0,50	0,52	0,51	/	0,57	0,46
dlbcl-tumor-from-harvard	0,73	0,84	0,91	0,78	0,96	0,79	0,84
kddcup99	1,00	1,00	0,97	1,00	1,00	0,96	1,00
letter	0,88	0,96	0,01	0,96	0,83	0,64	0,96
Povprečje:	**0,78**	**0,81**	**0,55**	**0,81**	**0,85**	**0,72**	**0,73**

Tabeli 4.13 in 4.15 nam prikazujeta približen čas izvajanja algoritmov, se pravi, da nam tabeli prikazujeta vsoto časa, ki je bil porabljen za učenje modela iz primerkov učne množice in časa, porabljenega za razvrščanje primerkov iz testne množice v ustrezne razrede. S tem si lažje predstavljamo, koliko časa algoritmi dejansko porabijo za analizo podatkov. Ker z uporabo WEKA vmesnika nisem mogel izmeriti natančnega časa izvajanja algoritma, sem čas izvajanja izmerili na podlagi LOG zapisov o izvajanju algoritma. Iz LOG zapisov sem tako pridobil čas začetka izvajanja algoritma in čas konca izvajanja algoritma zgolj v sekundah natančno, zato se v tabelah 4.13 in 4.15 pojavijo vrednosti 0 sekund, kar pa si razlagamo kot, da je bilo izvajanje algoritma hitrejše od 1 sekunde oz. je bilo vsaj reda stotink sekunde ali pa manj.

V povprečju je na malih podatkovnih množicah najmanj časa za izvajanje porabil algoritem NaiveBayes z 0,14 sekunde. Drugi čas izvajanja je dosegel AdaBoost.M1 (DecisionStump), ki je za izvajanje v povprečju potreboval 0,86 sekunde. Največ časa sta na malih podatkovnih množicah porabila algoritma SMO, ki je potreboval 48,43 sekunde, in RandomForest, ki je potreboval 18,71 sekunde. Oba sta prav tako dosegla opazne razlike, v primerjavi z ostalimi algoritmi pri izvajanju na podatkovnih množicah nursery in splice (slika 4.6). Izstopata pa še vrednosti na podatkovni množici waveform-5000, kjer sta algoritma RandomForest in AdaBoost.M1 (J48) porabila veliko več časa, kot preostali algoritmi.

Tabela 4.13: Meritve časa izvajanja klasifikacije na majhnih podatkih v sekundah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
audiology	0	1	0	1	8	0	0
kdd_synthetic_control	1	5	0	8	3	0	0
mushroom	1	6	1	1	19	0	7
nursery	0	14	1	6	111	0	13
sonar	0	2	0	3	1	0	0
splice	1	42	1	3	193	0	6
waveform-5000	6	61	3	72	4	1	9
Povprečje:	**1,29**	**18,71**	**0,86**	**13,43**	**48,43**	**0,14**	**5,00**

V povprečju so algoritmi na malih množicah potrebovali največ časa na podatkovnih množicah splice, waveform-500 in nursery (tabela 4.14, slika 4.6).

Tabela 4.14: Povprečni časi izvajanja algoritmov na posameznih malih podatkovnih množicah

Ime podatkovne množice	**povprečni čas izvajanja na podatkovno množico**
audiology	1,43
kdd_synthetic_control	2,43
mushroom	5,00
nursery	20,71
sonar	0,86
splice	35,14
waveform-5000	22,29

Tabela 4.15 in slika 4.7 nam prikazujeta čas, ki je bil potreben za izvajanje algoritmov na velikih podatkih. Brez upoštevanja časa, kjer smo prekinili izvajanje algoritmov, kadar je čas izvajanja presegel 12 ur, je bil najhitrejši algoritem J48. Ostali algoritmi pa so za klasifikacijo potrebovali bistveno več časa. Najpočasnejši je bil v povprečju algoritem IBk, ki je potreboval 3009,9 sekunde (približno 50 minut). IBk je bistveno več časa od preostalih algoritmov potreboval na podatkovni množici z največ primerkov, na kateri je za klasifikacijo potreboval 18825 sekund (več kot 5 ur). Čas, potreben za klasifikacijo je pri algoritmu IBk hitro narasel z večjim številom primerkov.

Tabela 4.15: Meritve časa izvajanja klasifikacije na velikih podatkih v sekundah

Ime podatkovne množice/ Algoritem	**J48**	**Random Forest**	**AdaBoost.M1 (Decision Stump)**	**AdaBoost. M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
arrhythmia	4	10	1	37	6	1	0
census-income	103	472	77	1.488	več kot 12 ur	4	1.124
cmu-newsgroup-clean-1000_sanitized	več kot 12 h	več kot 12 h	630	več kot 12 h	733	1.507	225
diabetic_data	62	206	10	635	več kot 12 h	3	844
dlbcl-tumor-from-harvard	3	5	15	13	0	2	1
kddcup99	158	1.100	129	1.705	755	68	18.825
letter	34	179	2	249	74	3	50
Povprečje:	**60,7**	**328,7**	**123,4**	**687,8**	**313,6**	**226,9**	**3009,9**

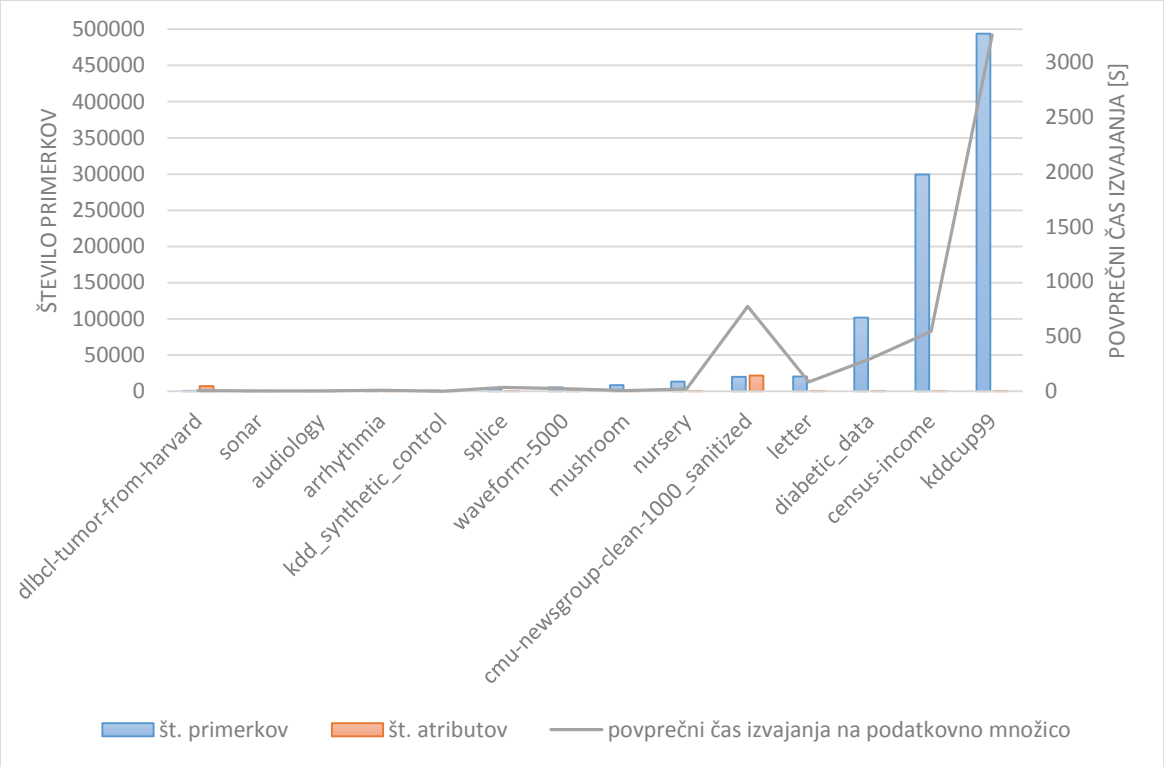
Iz tabele 4.16 in slike 4.7 vidimo, da so algoritmi v povprečju porabili na velikih množicah več časa za izvajanje, kot pa na malih. Časovna razlika med podatkovno množico, kjer so algoritmi v povprečju porabili najmanj časa (tabela 4.14) in podatkovno množico, na kateri so algoritmi v povprečju porabili največ časa (tabela 4.16), je kar 3247,71 sekunde (malo več kot 54 minut).

Tabela 4.16: Povprečni časi izvajanja algoritmov na posameznih velikih podatkovnih množicah

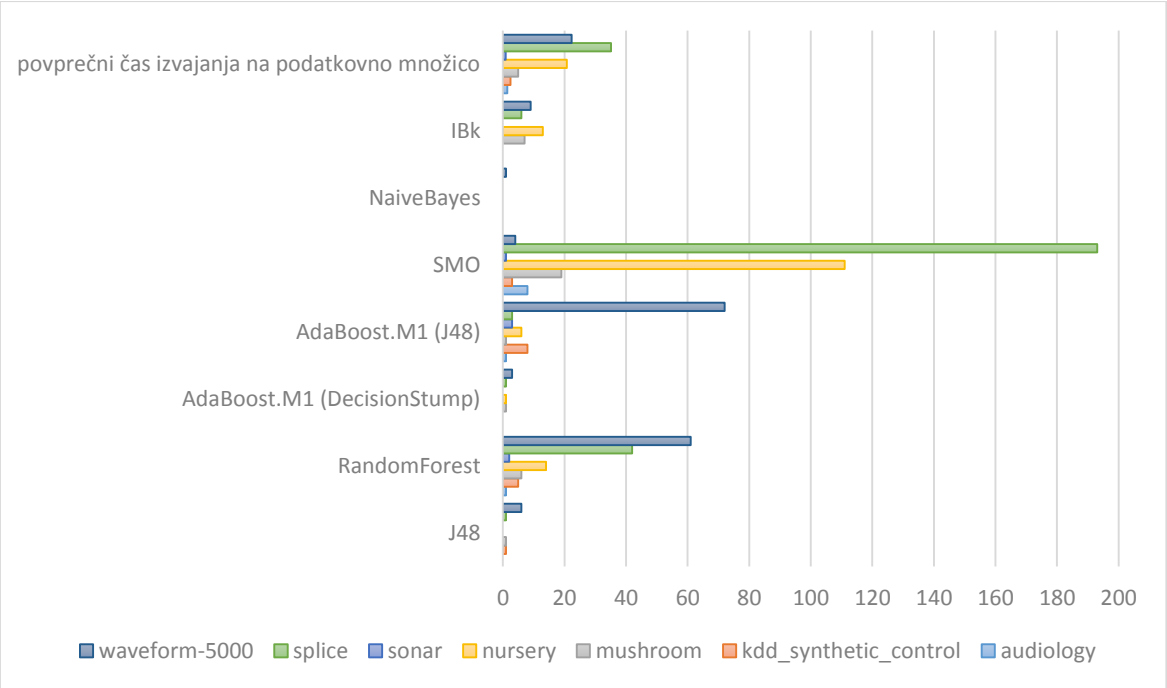
Ime podatkovne množice/ Algoritem	**povprečni čas izvajanja na podatkovno množico**
arrhythmia	8,43
census-income	544,67
cmu-newsgroup-clean-1000_sanitized	773,75
diabetic_data	293,33
dlbcl-tumor-from-harvard	5,57
kddcup99	3248,57
letter	84,43

Iz slike 4.5 je razvidno, da se je čas izvajanja začel povečevati s številom primerkov. Iz tega lahko sklepamo, da se bo čas izvajanja, potreben za analizo podatkov povečeval s številom primerkov in številom atributov, če so bili le-ti dovolj veliki in kompleksni. Ker imamo pri velikih podatkih opravka z velikim številom podatkov, moramo za analizo izbrati tak algoritem, ki nam bo dal pravo razmerje med časom izvajanja in točnostjo. Pri tem pa je izbira odvisna od namena oz. problema analize. V nekaterih primerih nam je bolj pomembno, da dobimo čim hitreje rezultate, ki lahko imajo malo manjšo točnost, v drugim pa nam je pomembna večja točnost in smo zanjo pripravljeni porabiti tudi več časa. Pri tem pa moramo upoštevati tudi, da velike podatke analiziramo tudi na večjem številu računalnikov, kar lahko bistveno vpliva na učinkovitost učnih algoritmov, saj, kot smo videli iz meritev, lahko učni algoritmi dosegajo različno učinkovitost na velikih in malih podatkovnih množicah. Pri izbiri učnih algoritmov pa moramo paziti tudi, da je algoritem zmožen analizirati podatke. Da algoritmi niso bili primerni za vse podatkovne množice, je razvidno iz meritev, kjer sem izvajanje algoritmov prekinil.

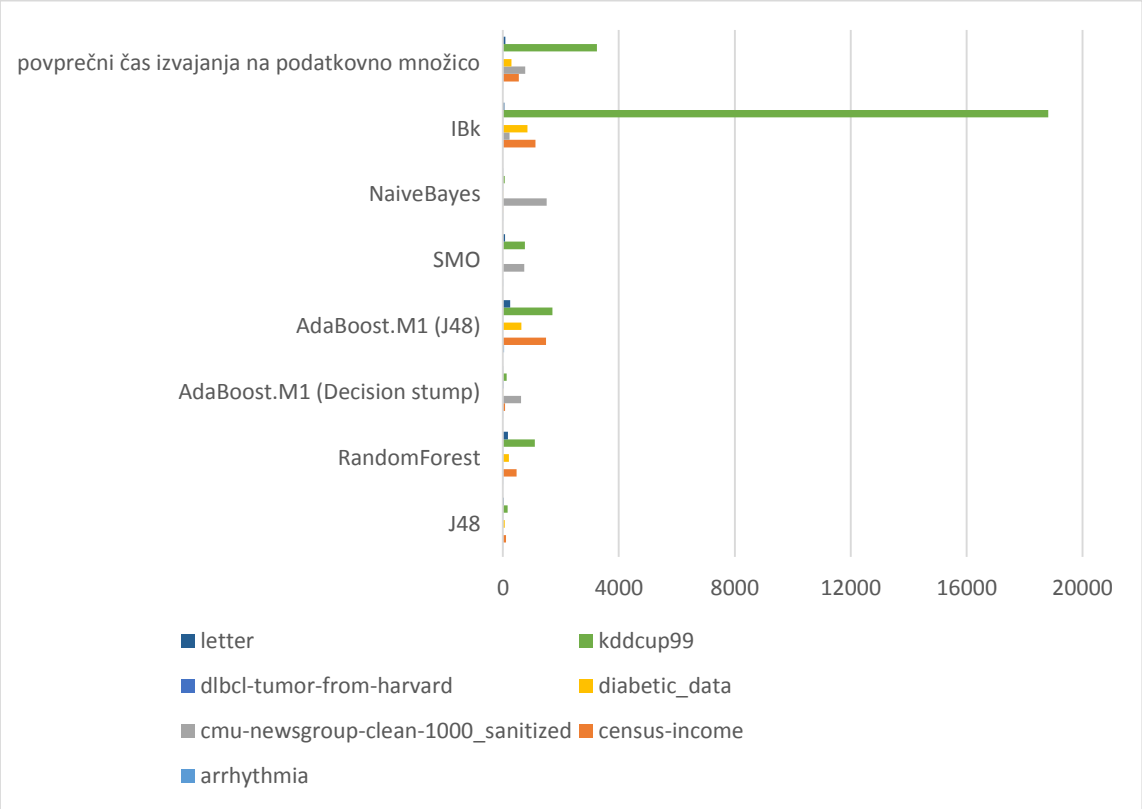
V nadaljevanju sem na podlagi meritev ocenil uspešnost učnih algoritmov na velikih podatkih.



Slika 4.5: Povprečni čas izvajanja na podatkovno množico v odvisnosti od št. primerkov in št. atributov:



Slika 4.6: Povprečni čas izvajanja na malih podatkovnih množicah



Slika 4.7: Povprečni čas izvajanja na velikih podatkovnih množicah

Kot že omenjeno v poglavju 4.1, nam tabela 4.17 prikazuje podatke, na podlagi katerih sem se odločil, da na velikih podatkovnih množicah census-income, cmu-newsgroup-clean-1000_sanitized, diabetic_data, kddcup99 nisem uporabil metode navzkrižne validacije, ampak metodo naključne razdelitve podatkov. Če primerjamo rezultate iz tabele 4.17 in rezultate za podatkovno množico kddcup99 iz tabel 4.5, 4.9, 4.12 in 4.15 je razvidno, da sem z uporabo naključne razdelitve podatkov vidno zmanjšal porabljen čas za izvajanje algoritma, pri tem pa je klasifikacijska točnost ostala enaka oz. se je spremenila za zelo malo. V primeru uporabe AdaBoost.M1 (J48) sem tako pohitril izvajanje algoritma za več kot 2 uri. Čas učenja modela pa je se je spremenil za zelo malo. V primerih algoritmov J48, SMO, NaiveBayes se je čas učenja modela nekoliko podaljšal, v primerih RandomForest, AdaBoost.M1(DecisionStump), AdaBoost.M1(J48), IBk pa se je čas za učenje modela zmanjšal.

Tabela 4.17: Meritve pridobljene z uporabo navzkrižne validacije na podatkovni množici kddcup99

	J48	**Random Forest**	**AdaBoost.M1 (DecisionStump)**	**AdaBoost.M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
Čas učenja modela [s]	83,21	727,62	81,11	1.094,86	525,61	3,86	/
Klasifikacijska točnost [%]	99,96	99,98	97,86	99,98	99,93	92,80	/
F-measure	1,00	1,00	0,97	1,00	1,00	0,95	/
Čas izvajanja algoritma	853	6.873	930	10.927	4.550	270	več kot 12 ur

4.3 Ocena učinkovitosti

V tem poglavju sem na podlagi meritev izbral učne algoritme, ki so po mojem mnenju najučinkovitejši za uporabo na velikih podatkih. Učne algoritme sem izbral na podlagi problemov, ki se lahko pojavijo pri analizi velikih podatkov. Prvi problem je, da potrebujemo

čim hitrejšo analizo/klasifikacijo podatkov, ki nam bo dala še dovolj dobro klasifikacijsko točnost. Drugi problem je potreba po doseganju najboljše klasifikacijske točnosti, ki pa bo lahko potrebovala malo dlje časa. Tretji problem je hitra in dovolj natančna analiza, ki predstavlja najboljši kompromis med hitrostjo analize in klasifikacijsko točnostjo.

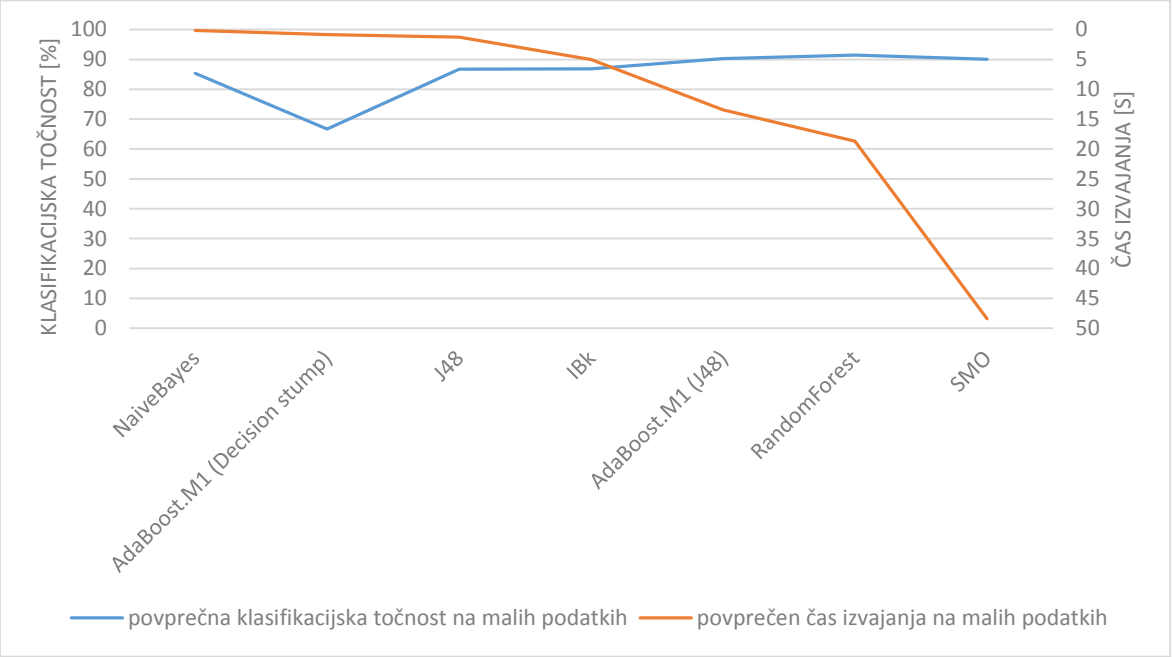
V tabeli 4.18 so povzeti rezultati povprečnih vrednosti meritev, ki jih bom uporabil za ocenjevanje učinkovitosti algoritmov na velikih podatkih.

Tabela 4.18: Meritve uporabljene za ocenjevanje učinkovitosti algoritmov

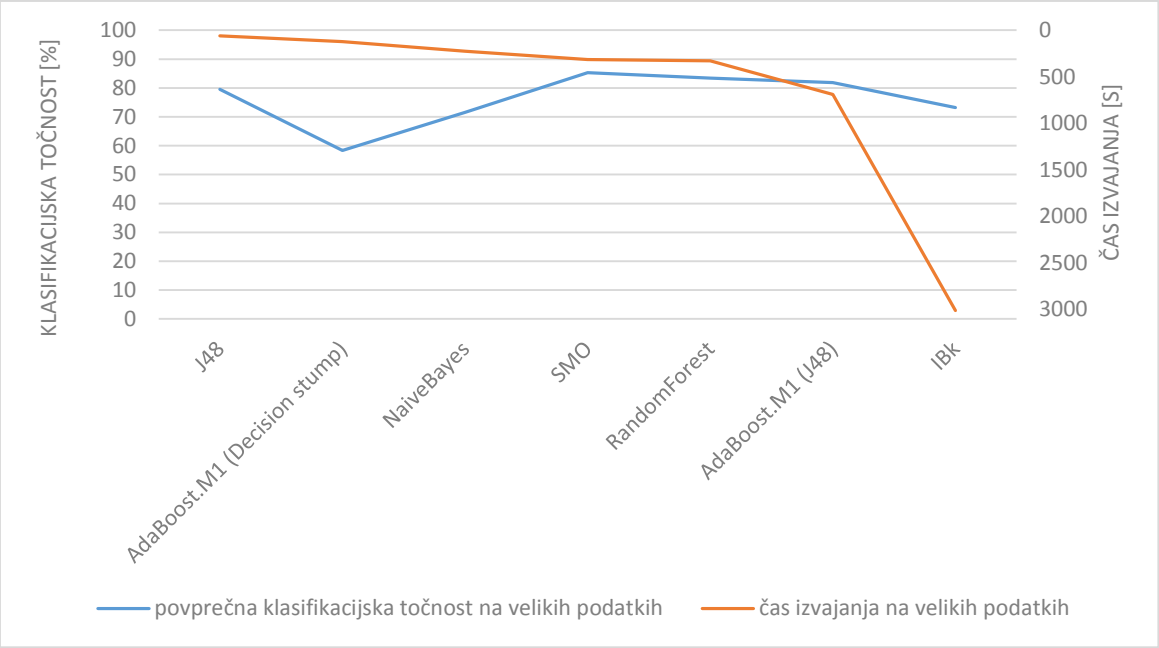
	J48	**Random Forest**	**AdaBoost.M1 (Decision stump)**	**AdaBoost. M1 (J48)**	**SMO**	**Naive Bayes**	**IBk**
Povprečna klasifikacijska točnost na malih podatkih	86,7	91,46	66,75	90,24	90,03	85,34	86,8
Povprečna klasifikacijska točnost na velikih podatkih	79,54	83,45	58,38	81,84	85,34	71,58	73,2
Povprečje klasifikacijske točnosti skupaj:	**83,12**	**87,46**	**62,57**	**86,04**	**87,69**	**78,46**	**80,00**
Čas izvajanja na malih podatkih	1,29	18,71	0,86	13,43	48,43	0,14	5
Čas izvajanja na velikih podatkih	60,7	328,7	123,4	687,8	313,6	226,9	3009,9
Povprečje časa izvajanja skupaj:	**31,0**	**173,7**	**62,1**	**350,6**	**181,0**	**113,5**	**1507,5**

Sliki 4.8 in 4.9 nam prikazujeta hitrosti izvajanja algoritmov in njihovo klasifikacijsko točnost, pri čem so algoritmi urejeni v vrstnem redu po času izvajanja, in sicer od najhitrejšega do najpočasnejšega. Iz slike 4.8 je razvidno, da je najhitrejši algoritem na malih podatkovnih množicah NaiveBayes, ki nam kljub svoji hitrosti izvajanja da dober rezultat točnosti. V

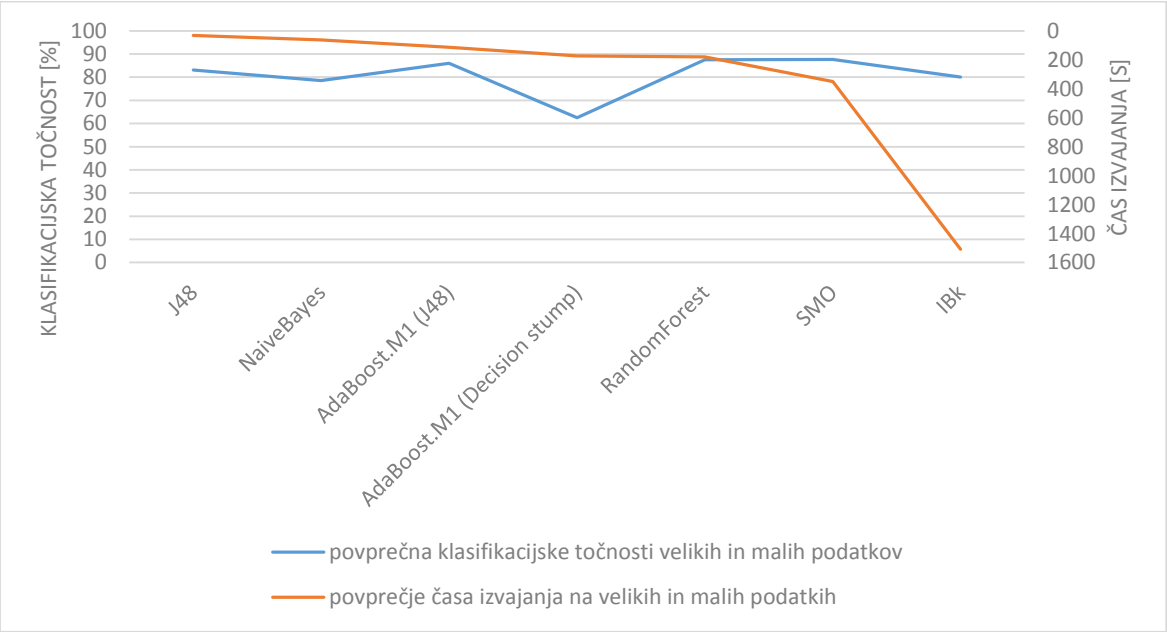
primerjavi z J48, ki nam da nekoliko boljšo točnost, je za več kot 1 sekundo hitrejši, kar lahko pripelje še do večje razlike pri velikih podatkih. Za problem, kjer se potrebuje čim hitrejšo analizo in imamo opravka z malimi podatkovnimi množicami, bi tako izbrali algoritem NaiveBayes. V primeru, da je število podatkov zelo veliko oz. so podatki bolj kompleksi pa bi izbrali algoritem J48 (slika 4.9). Algoritem J48 bi bil prav tako najučinkovitejši v primeru hitre analize, ko ne vemo, ali bodo podatki veliki ali majhni.



Slika 4.8: Prikaz klasifikacijske točnosti in hitrosti izvajanja od najhitrejšega do najpočasnejšega na malih podatkovnih množicah



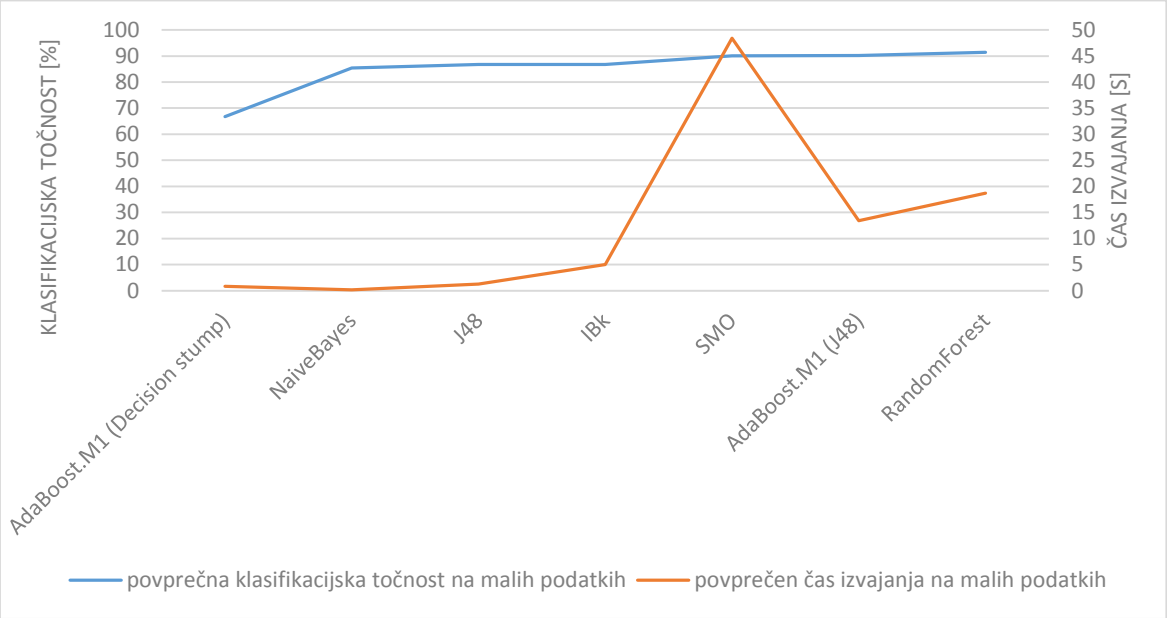
Slika 4.9: Prikaz klasifikacijske točnosti in hitrosti izvajanja od najhitrejšega do najpočasnejšega na velikih podatkovnih množicah



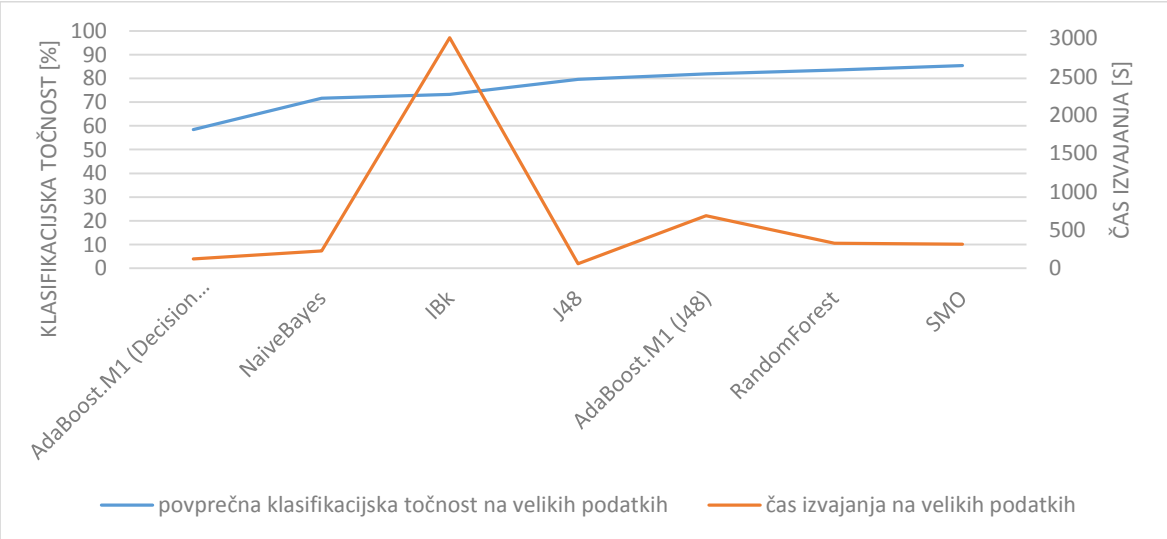
Slika 4.10: Prikaz povprečne klasifikacijske točnosti in hitrosti izvajanja od najhitrejšega do najpočasnejšega na med velikimi in malimi podatkovnimi množicami

S slikami 4.11, 4.12 in 4.13 si lahko pomagamo pri izbiri najučinkovitejših algoritmov za drugi problem, kjer me zanima največja kvalifikacijska točnost, čas izvajanja pa mi pri tem

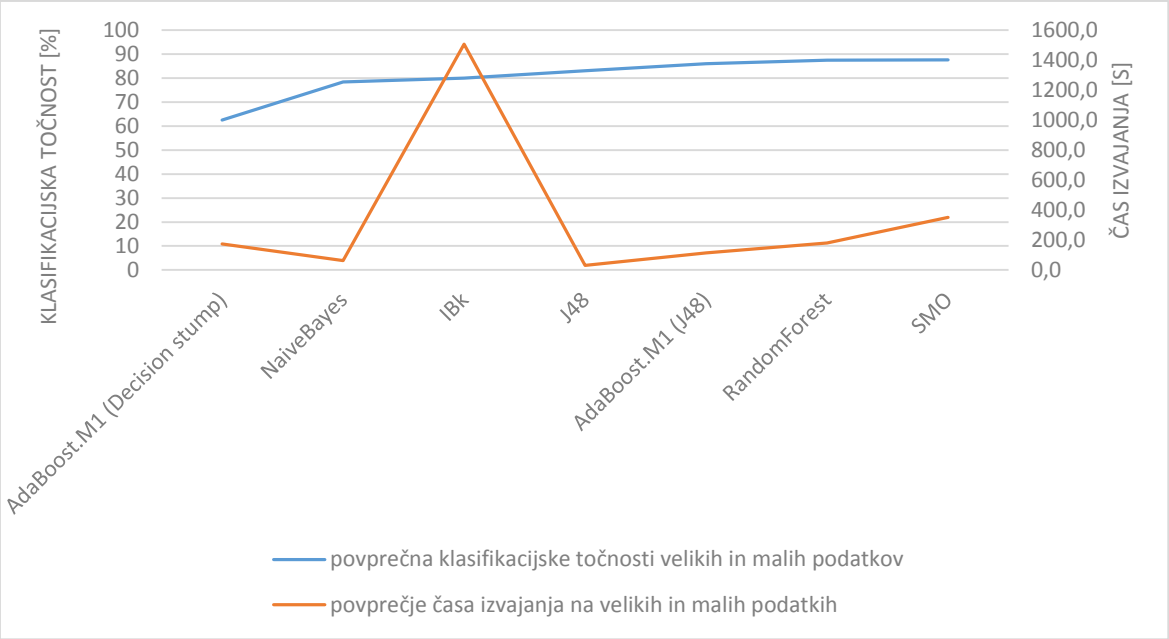
ni tako pomemben. V tem primeru bi bil najučinkovitejši algoritem za male podatkovne množice RandomForest, ki nam da najboljšo točnost, čas izvajanja pa kljub temu ni najslabši. Za velike množice in množice, pri katerih je pomembna skalabilnost učnih algoritmov na velike in male podatkovne množice, pa bi izbrali algoritem SMO (slika 4.12) ali pa algoritem RandomForest, saj so bili njuni rezultati zelo primerljivi.



Slika 4.11: Prikaz klasifikacijske točnosti in hitrosti izvajanja od najmanj točnega do najbolj točnega na malih podatkih

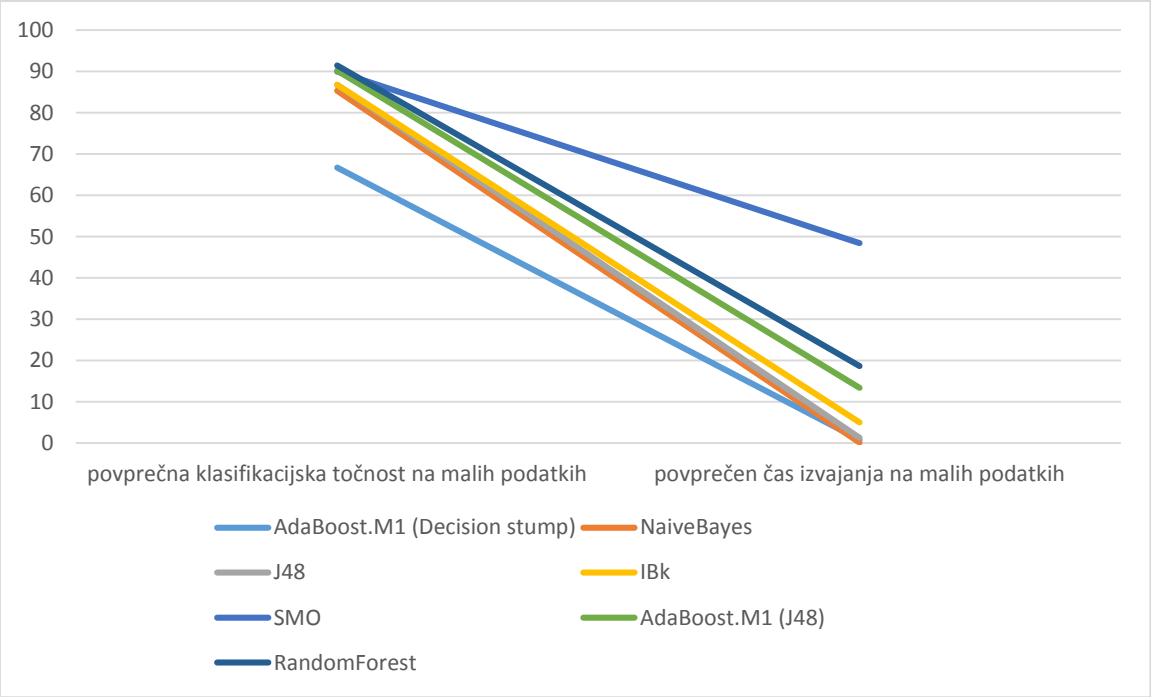


Slika 4.12: Prikaz klasifikacijske točnosti in hitrosti izvajanja od najmanj točnega do najbolj točnega na velikih podatkih

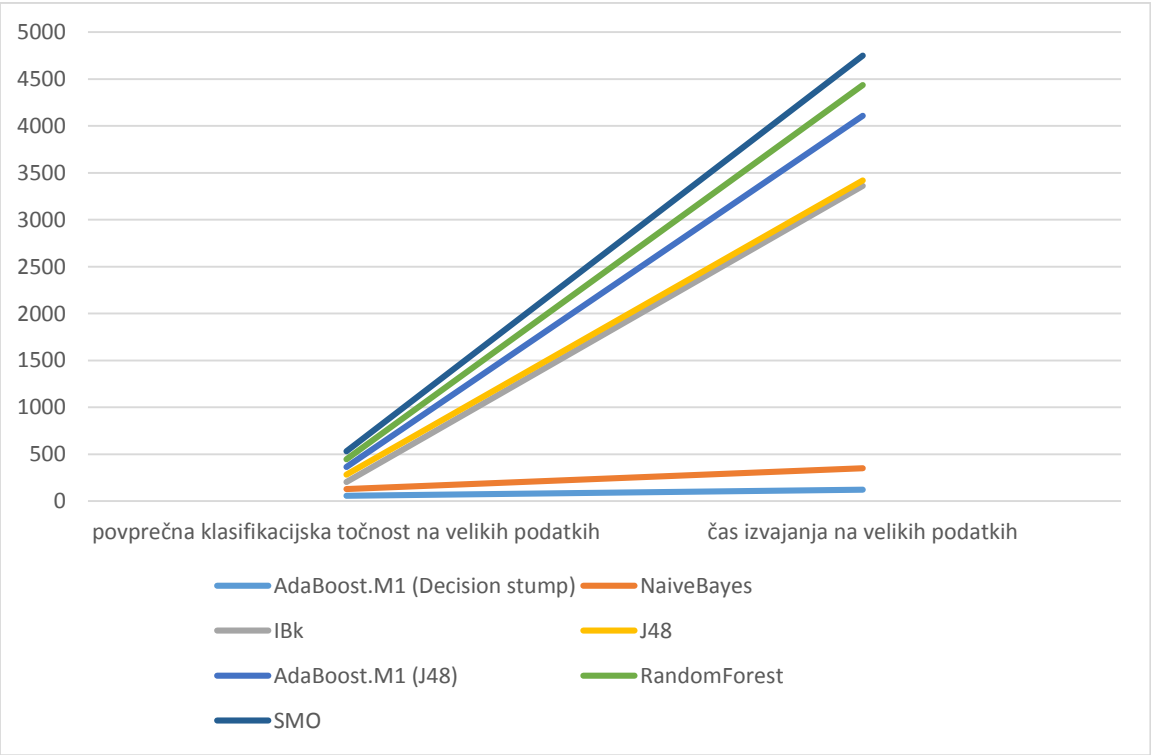


Slika 4.13: Prikaz povprečne klasifikacijske točnosti in hitrosti izvajanja od najmanj točnega do najbolj točnega med velikimi in malimi podatkovnimi množicami

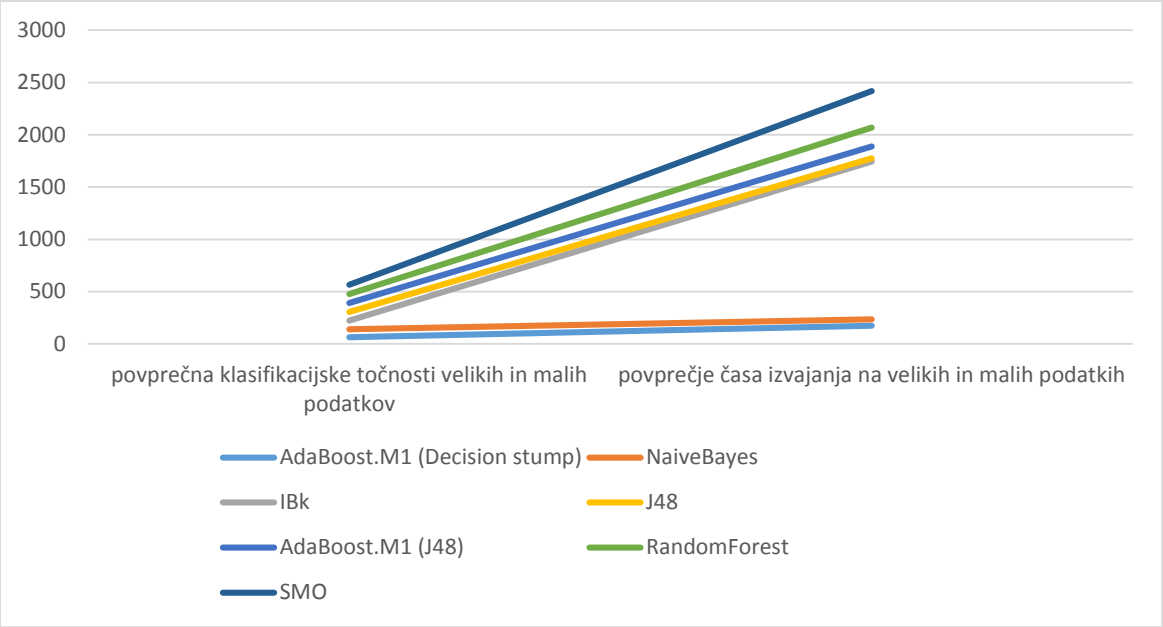
Iz slik 4.14, 4.15, 4.16 je razvidno, da je algoritem J48 konstantno dosegel dobro rezultate ali pri hitrosti izvajanja ali pa klasifikacijski točnosti. Zato bi algoritem J48 izbral za tretji problem, kot najučinkovitejši algoritem, kjer potrebujemo hitro in kljub temu natančno analizo. Iz slike 4.4 je razvidno, da se je algoritem uspešno učil nad različnimi podatkovnimi množicami.



Slika 4.14: Primerjava klasifikacijske točnosti in hitrosti izvajanja algoritmov na malih podatkih



Slika 4.15: Primerjava klasifikacijske točnosti in hitrosti izvajanja algoritmov na velikih podatkih



Slika 4.16: Primerjava povprečne klasifikacijske točnosti in hitrosti med velikimi in malimi podatki izvajanja algoritmov

5 SKLEP

Doba velikih podatkov, kot lahko poimenujemo dobo, v kateri se trenutno nahajamo, ima v današnjem času vse večji pomen in vpliv na vsakdanje življenje. Podjetja, organizacije ustvarjajo čedalje več podatkov, ki jih želijo čim bolj uspešno izkoristiti za reševanje problemov, doseganja ciljev in ustvarjanje vrednosti. Za pridobitev vrednosti iz teh podatkov pa jih morajo najprej analizirati. V ta namen uporabljajo različne učne algoritme strojnega učenja, ki pa nimajo vedno iste učinkovitosti.

V diplomski nalogi sem v ta namen izvedel eksperiment, s katerim sem skušal ugotoviti, kateri izmed izbranih učnih algoritmov je za velike podatke najučinkovitejši. Pri izvajanju eksperimenta sem meril hitrost učenja modela, klasifikacijsko točnost, F-measure in hitrost izvajanja celotne analize podatkov. Na podlagi meritev sem izbral algoritme, ki po mojem mnenju najučinkoviteje rešujejo zadane probleme, kot so potreba po hitri analizi, veliki točnosti ali pa pravem sorazmerju med hitrostjo in klasifikacijsko točnostjo, kjer pa upoštevamo, da se učinkovitost algoritmov lahko razlikuje med izvajanjem na malih in velikih podatkovnih množicah.

S pomočjo eksperimenta tako lahko sklepam, da je za učinkovito analizo »velikih podatkov« potrebna dobra izbira učnega algoritma, ki bo iz podatkov pridobil največjo vrednost. Pri izbiri pa moramo upoštevati tudi, nad kakšnimi podatki bomo analizo izvedli, ter kaj z njo želimo doseči, bodisi hitro analizo, veliko točnost ali pa kompromis med hitrostjo in točnostjo. Paziti pa moramo tudi na to, da nekateri algoritmi s privzetimi nastavitvami niso primerni za analizo velikih podatkov. To se vidi tudi iz rezultatov meritev, kjer nismo mogli pridobiti meritev v manj kot 12 urah, ali pa smo za boljšo analizo morali spremeniti določene nastavitve. Sprememba nastavitev algoritma je bila opazna pri algoritmu AdaBoost.M1, kjer lahko s samo spremembo šibkega algoritma dobimo boljšo učinkovitost. Na podatkovni množici diabetic_data, pa sem moral omejit število dreves pri algoritmu RandomForest.

VIRI

[1] McKinsey & Company, “Big data: The next frontier for innovation, competition, and productivity,” *McKinsey Glob. Inst.*, no. June, p. 156, 2011.

[2] J. Dean, “Big Data, Data Mining and Machine Learning.”

[3] I. Kononenko, *Strojno učenje*, 2. popravl. Ljubljana: Založba FE in FRI, 2005.

[4] “UCI Machine Learning Repository.” [Online]. Available: http://archive.ics.uci.edu/ml/. [Accessed: 04-Oct-2015].

[5] “mldata.” [Online]. Available: http://mldata.org/. [Accessed: 04-Oct-2015].

[6] “Weka 3 - Data Mining with Open Source Machine Learning Software in Java.” [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 04-Oct-2015].

[7] J. Gantz and D. Reinsel, “Extracting Value from Chaos,” *IDC IVIEW*, 2011. [Online]. Available: http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf. [Accessed: 03-Sep-2015].

[8] D. Feinleib, *Big Data Bootcamp*. Apress, 2014.

[9] B. J. Gantz, D. Reinsel, and B. D. Shadows, “THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East,” *Study report, IDC*, vol. 2007, no. December 2012, pp. 1–16, 2012.

[10] D. Laney, “3D Data Managment: Controlling Data Volume, Velocity, and Variety.,” *Tech. report, META Group, Inc (now Gartner, Inc.)*, no. February 2001, p. 4, 2001.

[11] E. Dumbill, “What is big data? An introduction to the big data landscape.” [Online]. Available: https://beta.oreilly.com/ideas/what-is-big-data. [Accessed: 12-Aug-2015].

[12] T. Kraska, “Finding the Needle in the Big Data Systems Haystack,” *IEEE INTERNET COMPUTING*, 2013. [Online]. Available: http://database.cs.brown.edu/papers/t-needlehaystack.pdf. [Accessed: 12-Aug-2015].

[13] A. Jacobs, “The Pathologies of Big Data,” *ACM Queue*. [Online]. Available: http://delivery.acm.org/10.1145/1540000/1536632/p36-jacobs.pdf?ip=89.143.103.65&id=1536632&acc=OPEN&key=4D4702B0C3E38B35.4D4702B0C3E38B35.4D4702B0C3E38B35.6D218144511F3437&CFID=535395668&CFTOKEN=29750845&__acm__=1439368579_3813277fb812cf8bef71395a84. [Accessed: 12-Aug-2015].

[14] O’Reilly Radar Team, *Big Data Now*. 2011.

[15] M. A. U. D. Khan, M. F. Uddin, and N. Gupta, “Seven V’s of Big Data understanding Big Data to extract value,” *Proc. 2014 Zo. 1 Conf. Am. Soc. Eng. Educ. - “Engineering Educ. Ind. Involv. Interdiscip. Trends”, ASEE Zo. 1 2014*, 2014.

[16] “Welcome to Apache™ Hadoop®!” [Online]. Available: https://hadoop.apache.org/. [Accessed: 03-Sep-2015].

[17] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” 2004. [Online]. Available: http://static.googleusercontent.com/media/research.google.com/sl//archive/mapreduce-osdi04.pdf. [Accessed: 03-Sep-2015].

[18] "A Modern Data Architecture with Apache™ Hadoop®: The Journey to a Data Lake," *A Hortonworks White Paper*, 2014. [Online]. Available: http://info.hortonworks.com/rs/h2source/images/Hadoop-Data-Lake-white-paper.pdf. [Accessed: 03-Sep-2015].

[19] Hortonworks Inc, "Business Value of Hadoop … as seen through data," 2013. [Online]. Available: http://hortonworks.com/wp-content/uploads/2014/05/Hortonworks.BusinessValueofHadoop.v1.0.pdf. [Accessed: 03-Sep-2015].

[20] "The 7 pillars of Big Data," *Petroleum Review*, 2015. [Online]. Available: http://www.arria.com/media-files/Pet Rev Jan - Arria NLG.pdf. [Accessed: 12-Aug-2015].

[21] "Digital Disease Detection: Using Social Media To Predict Flu Trends | HealthMap." [Online]. Available: http://www.healthmap.org/site/diseasedaily/article/digital-disease-detection-using-social-media-predict-flu-trends-31114. [Accessed: 07-Sep-2015].

[22] "Using Twitter and Social Media to Predict Disease: Identifying Risk and Impacting Change." [Online]. Available: http://thedoctorweighsin.com/using-twitter-social-media-predict-disease-identifying-risk-impacting-change/. [Accessed: 07-Sep-2015].

[23] S. Oberlin, "Machine Learning, Cognition, and Big Data." [Online]. Available: http://www.ca.com/us/~/media/files/articles/ca-technology-exchange/machine-learning-cognition-and-big-data-oberlin.aspx. [Accessed: 07-Sep-2015].

[24] D. Wang, X. Liu, and M. Wang, "A DT-SVM Strategy for Stock Futures Prediction with Big Data," *Comput. Sci. Eng. (CSE), 2013 IEEE 16th Int. Conf.*, 2013.

[25] S. K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," 1997. [Online]. Available:

http://citeseerx.ist.psu.edu/viewdoc/similar?doi=10.1.1.26.9663&type=ab. [Accessed: 01-Sep-2015].

[26] J. R. Quinlan and M. Kaufmann, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA, 1993.

[27] L. Breiman, “RANDOM FORESTS,” *University of California*, 2001. [Online]. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf. [Accessed: 07-Sep-2015].

[28] Y. Freund and R. E. Schapire, “Experiments with a New Boosting Algorithm,” *AT&T Research*, 1996. [Online]. Available: http://web.eecs.utk.edu/~leparker/Courses/CS425-528-fall10/Handouts/AdaBoost.M1.pdf. [Accessed: 07-Sep-2015].

[29] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*,” 1996. [Online]. Available: http://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf. [Accessed: 07-Sep-2015].

[30] “10701 Machine Learning.” [Online]. Available: http://www.cs.cmu.edu/~./10701/lecture/boosting.pdf. [Accessed: 07-Sep-2015].

[31] V. Vapnik, *The nature of statistical learning theory*, 2nd editio. Springer Verlag, 1995.

[32] “Naive Bayesian.” [Online]. Available: http://www.saedsayad.com/naive_bayesian.htm. [Accessed: 07-Sep-2015].

[33] O. Sutton, “Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction,” 2012. [Online]. Available: http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf. [Accessed: 07-Sep-2015].

[34] E.M. Mirkes, “KNN and Potential Energy,” *University of Leicester*, 2011. [Online]. Available: http://www.math.le.ac.uk/people/ag153/homepage/KNN/KNN3.html#L2. [Accessed: 07-Sep-2015].

[35] L. Kozma, “k Nearest Neighbors algorithm (kNN),” *Helsinki University of Technology*, 2008. [Online]. Available: http://www.lkozma.net/knn2.pdf. [Accessed: 07-Sep-2015].

Univerza v Mariboru

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

FERI

IZJAVA O AVTORSTVU

Spodaj podpisani/-a JOŽE GOBAR

z vpisno številko EM0647M

sem avtor/-ica diplomskega dela z naslovom:

UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

(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)

  ASSIST. 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 6.10.2015

Podpis avtorja/-ice:

Jože Gobar

Univerza v Mariboru

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

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: JOŽE GOBAR

Vpisna številka: E10647111

Na programu: INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA

izdelal zaključno delo z naslovom:

UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

(naslov zaključnega dela v slovenskem in angleškem jeziku)

EFFICIENCY OF LEARNING ALGORITHMS ON BIG DATA

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:
6.10.2015, MARIBOR

Podpis mentorja:

Datum in kraj:
6.10.2015, MARIBOR

Podpis somentorja (če obstaja):

Priloga:

- Poročilo o preverjanju podobnosti z drugimi deli.«

UNIVERZA V MARIBORU

Fakulteta za elektrotehniko, računalništvo in informatiko

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

Ime in priimek diplomanta-tke: Jože Gobar

Vpisna številka: E10647/11

Študijski program: INFORMATIKA IN TEHNOLOGIJE KOMUNICIRANJA

Naslov diplomskega dela: UČINKOVITOST UČNIH ALGORITMOV NA VELIKIH PODATKIH

Mentor: Vili Podgorelec

Somentor: Sašo Karakatič

Podpisani-a Jože Gobar **izjavljam, da sem za potrebe arhiviranja oddal elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Diplomsko 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 diplomskega dela je istovetna 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 diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah UM.

Datum in kraj: Maribor, 06.10.2015

Podpis diplomanta-tke: Jože Gobar

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)