ANALIZA MODELA PREDVIĐANJA TRENDOVA KRETANJA CIJENA VRIJEDNOSNIH PAPIRA

Kujundžić, Marko

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

Diplomski rad

ANALIZA MODELA PREDVIĐANJA TRENDOVA KRETANJA CIJENA VRIJEDNOSNIH PAPIRA

Rijeka, rujan2015

Marko Kujundžić

0069047935

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

Diplomski rad

ANALIZA MODELA PREDVIĐANJA TRENDOVA KRETANJA CIJENA VRIJEDNOSNIH PAPIRA

Mentor: doc. dr. sc. Ivan Štajduhar

Rijeka, rujan2015

Marko Kujundžić

0069047935

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Povjerenstvo za završne ispite preddiplomskog sveučilišnog studija računarstva
Diplomski sveučilišni studij računarstva
Klasa: 602-04/15-08/22
Ur. br.: 2170-15-14-15-1
Rijeka, 12.3.2015.

ZADATAK
za diplomski rad

Pristupnik: **Marko Kujundžić** JMBAG: 0069047935
Lok. mat. br.: 13040025

Naslov zadatka: **ANALIZA MODELA PREDVIĐANJA TRENDOVA KRETANJA CIJENA VRIJEDNOSNIH PAPIRA**

Thesis title: **ANALYSIS OF FINANCIAL-TRENDS FORECASTING MODELS**

Sadržaj zadatka: Zadatak je provesti sveobuhvatno testiranje raznih pristupa modeliranju trendova kretanja cijena vrijednosnih papira iz tehničkih podataka. Razmotriti učinkovitije tehnike pripremne obrade podataka, modele predstavljanja znanja i pripadajuće algoritme. Provesti temeljito testiranje i međusobnu usporedbu učinkovitosti razmatranih mehanizama. Za testiranje koristiti podatke o simbolima i indeksima nekolicine svjetskih burzi.

Zadano: 12.3.2015.

Mentor

doc.dr.sc. Ivan Štajduhar

Predsjednik Povjerenstva

izv.prof.dr.sc. Miroslav Joler

Zadatak preuzeo dana: 16.3.2015.

M. Kujundžić
(potpis pristupnika)

Dostaviti:
- Pristupnik
- Studentska služba
- Mentor
- Djelovođa Povjerenstva
- Predsjednik Povjerenstva

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

IZJAVA

Sukladno članku 11. Pravilnika o diplomskom studiju Tehničkog fakulteta Sveučilišta u Rijeci izjavljujem da sam diplomski rad izradio samostalno.

Rad sam izradio prema zadatku Povjerenstva za diplomske ispite sveučilišnog studija računarstva pod vodstvom mentora doc. dr. sc. Ivana Štajduhara uz navedenu literaturu.

Rijeka, rujan 2015.

Marko Kujundžić

0069047935

_____

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni diplomski studij računarstva

ZAHVALA

Zahvaljujem se mentoru doc. dr. sc. Ivanu Štajduharu na podršci i savjetima tijekom izrade diplomskog rada.

Također se zahvaljujem svojim prijateljima, kolegama i obitelji za podršku i korisne savjete kojima su mi pomagali za vrijeme izrade rada.

Rijeka, rujan 2015.

Marko Kujundžić

0069047935

SADRŽAJ:

1. UVOD

Predviđanje kretanja burze je privlačna ali veoma zahtijevna aktivnost. U pokušaju pronalaženja rješenja su se okušali mnogi znanstvenici, investitori i financijske institucije. Neki od razloga i faktora zašto je to tako težak problem su: nesigurnost kretanja tržišta, politički događaji, ekonomska, socijalna i psihološka situacija u zemlji i inozemstvu, očekivanja ulagača i dioničara i sl. Važno je napomenuti da pogrešno tumačenje tih faktora može rezultirati s značajnijom financijskom štetom. Također je i veoma privlačno budući da je moguće ostvariti značajnu financijsku dobit. Puno istraživanja se odvija u ovoj domeni. Ali, do sada se nije došlo do nekog zadovoljavajućeg rješenja. Još uvijek postoji debata oko toga je li moguća predikcija burzovnog tržišta na optimalan i adekvatan način.

Akademska istraživanja su pokazala da kretanja tržišta nisu nasumična, već da se kreću na nelinearan, dinamičan način. Također se pokazalo da je umjesto predviđanja točne cijene dionica ili burzovnog indeks puno učinkovitiji način predviđanja same procjene hoće li cijene pojedine dionice burze ili burzovnog indeks pasti ili rasti.

Prvi pokušaji odnosno metode koje su se koristile pri predikcijama su one bazirane na statistici: *autoregresive model* (AR) i *autoregresive moving average model* (ARMA). To su linearni modeli koji su najčešće neadekvatni prilikom predikcije kretanja burze budući da burza sadrži puno šumova i nije stacionarna[1]. Početkom 90-tih godina prošloga stoljeća u svrhu predikcije kretanja burzovnih dionica počele su se koristiti tehnike i algoritmi iz strojnog učenja. Primjeri nekih od algoritama koji se najčešće koriste su stablo odluke (engl.decision tree), stroj s potpornim vektorima (engl. support vector machines), nasumična šuma (engl. random forest) i sl. Cilj ovog rada je opisti neke od tih algoritama te utvrditi njihovu točnost predikcije koristeći nekoliko načina testiranja i nekoliko načina njihove evaluacije: f-mjera (engl. f-score, f1-score),konfuzna matrica (engl. confusion matrix) i točnost klasifikacije(engl. classification accuracy). Testiranje se provelo na uzorku od pet svjetskih burzi, s po deset dionica na svakoj burzi te s jednim do dva indeksa svake burze s maksimalnom mogućom količinom povijesnih podataka. Povijesni podaci se sastoje od zapisa koji su se spremali na dnevnoj bazi, a sami zapisi obuhvaćaju sljedeće karakteristike: najviša cijena dionice ili indeksa tokom dana (engl. High), najniža cijena (engl. Low), postojeća cijena po otvaranju (engl. Open) i zatvaranju burze (engl. Close), te brojčani opseg trgovanja toga dana (engl. Adj Close).

Umjesto da se kao ulaz u algoritme označavaju neke od vrijednosti povijesnih podaka s burze, tehnika koja se koristila u ovom radu je da se kao ulaz algoritmima za strojno učenje podaci formuliraju kao tehnički indikatori koji su proizašli iz tih podataka. Opisi tehničkih indikatora koji su se koristili će biti navedeni dalje u ovom radu. Tako formatirani podaci su se potom trenirali i testirali na tri načina: empirijska točnost (treniranje i testiranje na istom skupu podataka), podjela podataka u podatke za učenje i podatke za treniranje u omjeru 80:20, te na kraju 10-dijelna unakrsna točnost s randomiziranim podacima. Testiranje se radilo po slojevima; za svaku burzu, te za svaki simbol i indeks na burzi.

2. STROJNO UČENJE

Računalni programi koji mogu prikupljati nova znanja i vještine kroz iskustvo su sve učestaliji. Koristimo programe strojnog učenja da bi otkrili novu glazbu koja će nam se svidjeti i da bi pronašli koje nam cipele odgovaraju i kupili ih preko interneta. Strojno učenje nam omogućava glasovno diktiranje uputa našim pametnim telefonima i omogućava našim termostatima da sami reguliraju svoju temperaturu. Strojno učenje može pomoći pri dešifriranju nespretno napisnih email adresa, te može čuvati naše kreditne kartice od raznih prijevara. Strojno učenje se danas koristi kod ispitivanja novih lijekova i sustava upravljanja za autonomne uređaje. Ukratko, strojno učenje postaje važan čimbenik mnogih industrija.

Neka od definicija strojnog učenja kaže da ono predstavlja dizajn računalnih algoritama koji koriste iskustvo iz prošlosti prilikom donošenja budućih odluka; to je studija programa koji uče iz podataka. Fundamentalni cilj strojnog učenja je generalizacija, odnosno sposobnost algoritma da njegova primjena bude što točnija na novim podacima/zadacima nakon izvjesnog treniranja na prethodnim podacima. Možemo na sljedećem primjeru strojnog učenja reći nešto više o spam filtriranju, odnosno detekciji mogu li se nekeof primljenih email poruka okarakterizirati kao spam (email koji je sam po sebi nepotreban i služi za davanje nekih ne bitnih i suvišnih informacija). Prilikom observacije tisuće emailova koji su prethodno bili obilježeni kao spam, spam filteri uče kako klasificirati nove poruke.

Arthur Samuel, računalni znanstvenik koji je je bio jedan od pionira znanosti o umjetnoj inteligenciji rekao je da je strojno učenje „znanost koja računalima daje mogućnost učenja bez da su prethodno programirani“. U 50-tim i 60-tim godinama prošlog stoljeća Samuel je razvio računalne programe koji su igrali igru dame. Iako su pravila te igre jednostavna, složene strategije su potrebne da bi se pobjedilo vješte protivnike. Samuel nije nikada eksplicitno programirao te strategije, već je program sam naučio kompleksna ponašanja kroz iskustvo igranja na tisuće partija, što mu je omogućilo pobjedu nad mnogim ljudskim protivnicima[2].

Računalni znanstvenik Tom Mitchell definira strojno učenje na više formalniji način: „za program se može reći da uči iz iskustva E prilikom izvršavanja zadataka T s mjerom performanse P, ako se performansa prilikom izvršavanja zadatka T mjerena s P, poboljšava s iskustvom E“. Primjerice, možemo zamisliti da posjedujemo kolekciju slika. Svaka slika prikazuje ili psa ili mačku. Zadatak može biti sortiranje slika u odvojene kolekcije slika za

mačke i za pse. Program potom može naučiti izvršavati zadatak sortiranja prilikom observacije slika koje su već sortirane, te bi poslije mogao procijeniti svoj uspjeh time što će izračunati postotak točno klasificiranih slika.

2.1 Učenje iz iskustva

Sustavi koji koriste strojno učenje se često opisuju kao oni koji uče iz iskustva uz ljudski nadzor ili bez njega. U nadziranom strojnom učenju (engl. supervised learning) računalni program ili algoritam predviđa izlaz tako što uči na primjeru pravilno označenih ulaza i izlaza, tj. uči iz primjera koji su obilježeni kao točni odgovori. U nenadziranom strojnom učenju (engl. unsupervised learning), računalni program ne uči iz označenog skupa podataka već sam pokušava otkriti uzorke ponašanja u podacima. Primjerice, pretpostavimo da imamo skup podataka koji opisuje visinu i težinu ljudi. Primjer nenadziranog učenja bi bio podjela podataka u grupe. Zadatak programa bi bio da producira grupe koje se dijele na muškarce i žene, ili djecu i odrasle. Sada pretpostavimo da su podaci označeni tako da prikazuju spol osobe. Primjer nadziranog učenja bi bio da računalni program pokuša pridvidjeti jeli osoba muško ili žensko koristeći podatke kao što je težina i visina.

Nadzirano učenje i nenadzirano učenje se mogu zamisliti kao dva suprotna dijela spektra. Neki tipovi problema koriste i nadzirano i nenadzirano učenje i oni su negdje na sredini tog spektra. Primjeri takvih problema u strojnom učenju se zovu podržano učenje (engl. reinforcement learning), kod kojeg računalni program dobije odgovor za odluke koje je donio, ali samiodgovor nemora biti povezan s nekom odlukom. Primjerice, računalni program koji uči igrati računalnu igru kao što je Super Mario Bros može primiti nagradu kada uspješno završi neku razinu igre ili kaznu ako izgubi život. Međutim, ta nadziranaklasa se ne asocira na specifičnu odluku koju akciju treba poduzeti u igri; trčati, izbjegavati protivnike ili skupljati cvjetove za dobivanje života.

Računalni program nadziranog učenja uči iz označenih primjera s izlaza koji bi se trebali pojaviti nakon odgovarajućeg ulaza. Postoji mnogo imena za izlaz koji daje program koji koristi strojno učenje. Termin koji ću ja nadalje koristiti je odgovor (engl. response variable). Druga imena koja se često pojavljuju u literaturi na engleskom jeziku su *dependent variables, regresant, criterion variables, measured variables, responding variables, explained*

variables, outcome variables i *output variables*. I ulazne varijable također imaju nekoliko imena. Termini na engleskom jeziku koji se još koriste su *predictors, regressors, controlled variables, manipulated variables i exposure variables*. Termin koji ću ja nadalje koristiti za ulaznu varijablu je klasa (engl. *class*).

Skup podataka koji su nastali nadziranim učenjem se naziva trening skup. Skup podataka koji nam služi na provjeru točnosti rada programa se zove test skup. Odgovor varijabla se može percipirati kao odgovor na pitanje koje je pitano od strane ulaznih varijabli. Problemi nadziranog učenja uče iz skupa odgovora na različita pitanja; odnosno, programi nadziranog učenja dobiju točne odgovore, te potom moraju naučiti kako točno odgovoriti na njima nepoznata, ali slična pitanja.

2.2 Zadaci strojnog učenja

Dva najčešća zadatka nadziranog strojnog učenja su klasifikacija i regresija. Kod klasifikacijskih zadataka računalni program mora naučiti predviđati diskretne vrijednosti za varijablu odgovora u odnosu na jednu ili više varijabli značajki odnosno klasa. Program mora predvidjeti najvjerojatniju kategoriju, klasu, ili oznaku za nove observacije. Neke od primjena klasifikacije su predviđanje vrijednosti hoće li dionica/simbol padati ili rasti, definiranje pripada li neki novinski članak sportskoj ili političkoj sekciji (analiza teksta) iutvrđivanjeidentitetapojedincaidentifikacijomotiskaprstaprilikom ulaska u zgradu .Kod regresijskih problema računalni program mora predvidjeti vrijednosti kontinuirane varijable odgovora. Neki od primjera regresijskih problema su predviđanje prodaje novih proizvoda i predviđanje plaće za posao bazirano na njegovom opisu. Slično kao i kod klasifikacije, i regresijski problemi zahtijevaju nadzirano učenje.

Česti zadatak nenadziranog učenja je otkrivanje skupa podataka koji su zvani grupe (engl.clusters) međusobno povezani unutar trening podataka. Sam postupak se često naziva clustering ili analiza *clustera*, i njegov je zadatak dodijeliti određene karakteristike grupama na način da su karakteristike slične jedinkama unutar grupe, više nego što su slične jedinkama izvan grupe. Analiza *clustera* se često koristi prilikom istraživanja velikih skupova podataka. Primjerice, na kolekciji filmskih kritika, *clustering* algoritam će pokušati otkriti skupove pozitivnih kritika i skupove negativnih kritika. Bez nadzora, algoritam neće moći procijeniti

odnosno označiti skup podataka kao pozitivan ili kao negativan. Imat će samo spoznaju o tome da su jedinke unutar grupe slične na neki način. Česta primjena analize *clustera* je u oglašavačkoj industriji gdje se upotrebom strojnog učenja otkriva koji kupac kupuje koje proizvode. Time se oglašivačima savjetuje koje aspekte svoje marketinške kampanje moraju poboljšati kako bi privukli ciljanu skupinu kupaca. *Clustering* algoritmi se također koriste kod internetskih radio stanica; na nekom skupu pjesma clustering algoritam bi mogao pokušati grupirati pjesme prema njihovom žanru i odrediti koja pjesma odgovara korisniku ovisno o njegovom trenutnom raspoloženju[3].

Smanjenje dimenzionalnosti (engl. dimensionality reduction) je još jedna česta primjena nenadziranog učenja. Neki problemi mogu u sebi sadržavati na tisuće ili čak milijune značajki koje mogu zauzimati puno memorijskog prostora. Također, moguće je da se zbog toliko velike količine podataka dođe do pojave šuma iz kojeg onda slijedi problem da program ne generalizira dobro. Rješenje tog problema je detekcija koje su značajke odgovorne za veliku promjenu, odnosno odmak od drugih podataka. Smanjenje dimenzionalnosti se također može koristiti kod vizualizacije podataka. Relativno je jednostavno vizualizirati regresijski problem kao što je predviđanje cijena nekretnina uzimajući u obzir njihovu veličinu; veličina nekretnine se može unutar grafa prikazati na x osi, dok se cijena nekretnine može prikazati na y osi. Shodno tome jednostavno je vizualizirati cijenu nekretnine kada se doda još jedna značajka. Broj soba unutar nekretnine može biti takva varijabla i ona se može dodati na z os. Problem koji ima na tisuće značajki postaje nemoguć zadatak za vizualiziranje.

2.3 Trening podaci i testni podaci

Observacije u skupu treniranih podataka su zapravo sažetak iskustva koje algoritam koristi za učenje. U problemima nadziranog učenja, svaka observacija se sastoji od odgovora i jedne ili više promatranih klasa.

Testni skup je slična skupina observacija koja se koristi radi evaluacije odnosno mjerenja performansi modela kojeg smo trenirali. Važno je napomenuti da nije najbolji pristup da trening skup i test skup dijele iste podatke. U ovom radu su se međutim trenirale i takve postavke radi usporedbe s drugim načinima treniranja i testiranja. Problem koji se može

javiti kod testiranja i treniranja na istom skupu podataka je da algoritam ne generalizira dobro, već jednostavno „zapamti“ trening skup. Program koji dobro generalizira će moći biti jednako učinkovit prilikom izvršavanja zadataka koji koriste drugi skup podataka. U suprotnome, program koji zapamti skup treniranih podataka tako da nauči prekompleksni model, će na tom skupu podataka imate odlične rezultate, ali će na nekom njemu nepoznatom skupu podataka imati loše rezultate.

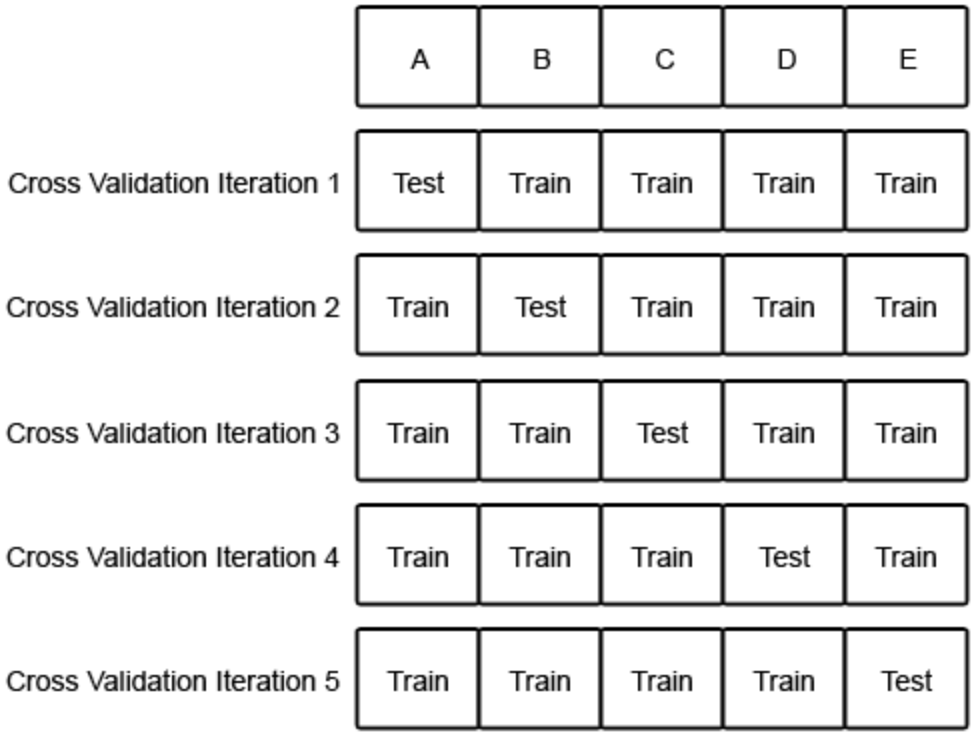
Pamćenje trening skupa se naziva pretreniranost modela (engl.overfitting). Program koji na taj način pamti observacije svoj zadatak neće izvršiti na najbolji način, budući da neke observacije mogu sadržavati šum ili upamtiti neke međuodnose između podataka koji su slučajni. Balansiranje između memorizacije i generalizacije, odnosno između prenaučenosti i podnaučenosti se kao problem pojavljuje u mnogim algoritmima strojnog učenja.

Uz trening skup i testni skup, često se koristi i treći skup observacija koji se naziva validacijski skup (engl.hold-out skup). Validacijski skup se koristi radi finog namještanja varijable koje se nazivaju hiperparametri; oni kontroliraju kako se model uči. Program se i dalje evaluira na testnom skupu radi procjene njegovih performansi u stvarnom svijetu; validacijski skup se ne bi smio koristiti prilikom izračunavanja točnosti budući da je sam program „namješten“ po postavkama koje odgovaraju validacijskom skupu. Česti je postupak prilikom kojeg se podaci particioniraju u trening skup, validacijski skup i testni skup. Veličina samih skupova može varirati uzevši u obzir količinu dostupnih podataka. Primjer podjele podataka je 50% ili više alocirati za trening skup, 25% za testni skup, te ostatak kao validacijski skup.

Neki trening skupovi mogu sadržavati nekoliko stotina observacija, dok drugi mogu sadržavati milijune. Unazad nekoliko godina sama količina podataka se drastično povećala: prostor za pohranu podataka koristeći računarstvo u oblaku je postao jeftin, povećana je mrežna povezanost, pojavili su se pametni telefoni koji bilježe veliku količinu senzorskih podataka itd. U pravilu prediktivna moć algoritama strojnog učenja se povećava kako se povećava količina trening skupa. Međutim, nekoliko algoritama strojnog učenja radi po principu „*garbage in, garbage out*“ (smeće unutra, smeće vani). Kao obrazloženje te tvrdnje uzmimo situaciju iz svakodnevnog studentskog života. Student koji uči za ispit čitajući kompliciranu i veliku knjigu koja sadrži dosta ne bitnih informacija će vrlo vjerojatno imati slabiji rezultat od studenta koji čita kratku ali efikasnu skriptu. Slično tome, algoritam koji je treniran na velikom skupu podataka može biti pun šumova i sadržavati ne bitne ili krivo

označene podatke. Algoritam koji je treniran na malom skupu podataka koji su bolje prezentacija problema će u većini slučajeva biti bolji izbor. Mnogo skupova podataka za nenadzirano strojno učenje se priprema ručno, kao što je bio slučaj i za ovaj rad.

Najtransparentnija metoda treniranja i testiranja podataka je unakrsna provjera . U unakrsnoj provjeri trening skup se particionira. U praksi to znači da se trenira na svim skupovima podataka osim na jednom koji služi za testiranje. Particije se potom rotiraju dok se na testiranje ne izvrši na svim skupovima podataka. Na slici 1 opisana je 5-dijelna unakrsna validacija.



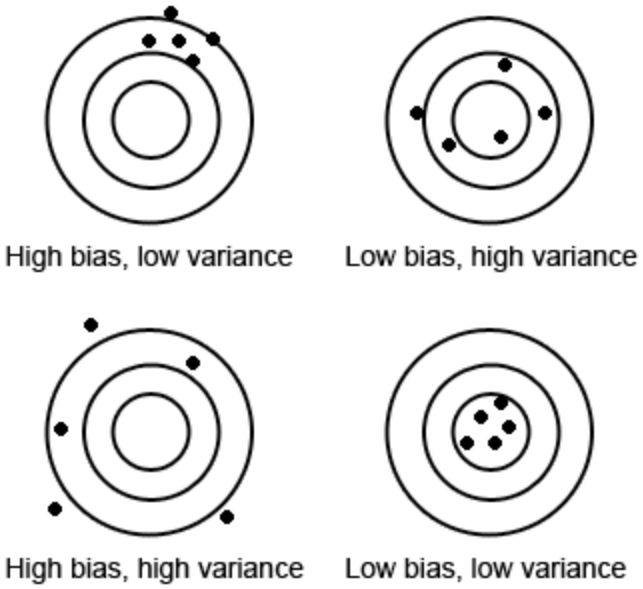
Slika 1. Unakrsna validacija(izvor [4])

Slika opisuje skup podataka koji je particioniran u 5 dijelova jednake veličine. Skupovi su označeni slovima A,B,C,D i E. Inicijalno model se trenira na particijama od B do E, a testira na particiji A. U sljedećoj iteraciji model se trenira na particijama A,C,D i E a testira se na particjii B. Particije se potom rotiraju sve dok modeli nisu trenirani i testirani na svim

particijama. Unakrsna provjera pruža procjenu koja je puno točnija od modela čija se procjena bazira na testiranju jedne particije podataka.

3. MJERENJE PERFORMANSI, PRISTRANOST I VARIJANCA

Mnogo metrika se može koristiti radi određivanja je li neki program uspješan u učenju u svrhu izvršenja svog zadatka ili nije. Za probleme nadziranog učenja, postoje mnoge metrike koje za svoju mjeru performanse uzimaju koliko se dogodilo prediktivnih pogrešaka. Dva su glavna razloga odgovorna za pogrešku predikcije, pristranost modela (engl.bias) i varijanca. Zamislimo da imamo trening skups podacima koji su svi jedinstveni, ali jednako dobro opisuju populaciju. Model s visokom pristranošću će javljati slične greške za ulazne podatke neovisno o trening skupu; model sam pokušava predvidjeti odnosno zaključiti koja je veza između podatka u trening skupu. Model s visokom varijancom će javljati različite greške za ulazne podatke ovisno o trening skupu na kojem je model bio treniran. Model s visokom pristranošću nije dovoljno fleksibilan, dok je model s visokom varijancom previše fleksibilan čime također modelira šum u trening skupu. Drugim riječima, to znači da model s visokom varijancom prenauči trening podatke dok model s visokom pristranošću podnauči trening podatke. Primjer koji slikovito opisuje pristranost i varijancu je igra pikada. Svaku pikado strelicu možemo promatrati kao predikciju različitog skupa podatka. Model s visokom pristranosti ali niskom varijancom će bacati strelice koje su daleko od središta mete, ali veoma blizu jedan drugom. Model s visokom pristranosti i visokom varijancom će bacati strelice preko cijele mete; strelice će također biti daleko od središta mete i daleko jedna od druge. Model s niskom pristranošću i visokom varijancom će bacati strelice koje su bliže središtu mete, ali nisu dovoljno dobro grupirane. Na kraju, model s niskom pristranosti i niskom varijancom će bacati pikado strelice koje su grupirane veoma blizu i veoma su blizu središta mete. Opis navedenih stanja je prikazan na slici 2.



Slika 2. Odnos pristranost/varijanca(izvor [4])

U idealnom slučaju model će imati i nisku varijancu i nisku pristranost, ali njihov odnos je najčešće obrnuto proporcionalan tako da će smanjenje jednog rezultirati rastom drugog. To stanje se u literaturi na engleskom jeziku još naziva *bias-variance trade-off.*

Algoritmi nenadziranog učenja ne javljaju grešku signala koji se odnosi na mjeru. Umjesto toga njihova točnost se mjeri pravilnim određivanjem nekih atributa koji su otkriveni u toj strukturi podataka.

Neke mjere evaluacije se mogu izračunati samo za specifični tip zadatka. Sustavi koji koriste strojno učenje bi se trebali evaluirati s mjerama koje najbolje opisuju cijenu koja je povezana s cijenom grešaka u stvarnom svijetu. Sljedeći primjer opisuje evaluaciju koja vrijedi za jedan zadatak, ali se nemože primjeniti specifično na neki problem. Zamislimo klasifikacijski problem čiji je zadatak observacija tumora i predviđanje koji je tumor iz skupa benigan, a koji je maligan. Točnost (engl. accuracy) je intuitivna mjera koja govori koliko je program točan. Iako točnost mjeri performanse programa, ona ne mjeri odnosno ne razlikuje slučajeve u kojima su maligni tumori bili pogrešno klasificirani kao benigni ili beningni tumori koji su pogrešno klasificirani kao maligni. U nekim aplikacijama, cijena takve pogreška može biti zanemariva, dok u ovom konkretnom slučaju je to ozbiljan propust;

neuspjeh identifikacije malignih tumora je mnogo ozbiljniji problem od pogrešnog klasificiranja benignog tumora kao malignog.

Možemo mjeriti svaki od mogućih izlaza koji daje klasifikacijski algoritam što nam omogućuje različite poglede na njegove performanse. Kada sustav točno klasificira tumor kao benigni ili maligni tada je predikcija točno pozitivna (engl. true positive). Ako sustav netočno klasificira benigni tumor kao maligni ili maligni tumor kao benigni, predikcija se označava kao pogrešno pozitivna (engl. false positive). Slično tome, pogrešno negativna (engl. false negative) je također netočna predikcija da je tumor benigan, i točno negativna (engl. true negative) je točna predikcija da je tumor benigan. Ta četiri ishoda se mogu koristiti prilikom izračuna nekoliko mjera kao što su točnost, preciznost (engl. precision) i opoziv (engl. recall).

Točnost se izračunava prema sljedećom izrazu:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

gdje je:

TP - broj točno pozitivnih rezultata
TN - broj točno negativnih rezultata
FP - broj pogrešno pozitivnih vrijednosti
FN - broj pogrešno negativnih vrijednosti

Preciznost predstavlja frakciju tumora koji su označeni kao maligni i njihova stvarna vrijednost je da su maligni. Preciznost se računa prema sljedećem izrazu:

$$P = \frac{TP}{TP + FP}$$

gdje je:

TP - broj točno pozitivnih rezultata
FP - broj pogrešno pozitivnih vrijednosti

Opoziv je je skup malignih tumora koje je sustav identificirao. Opoziv se računa po sljedećem izrazu:

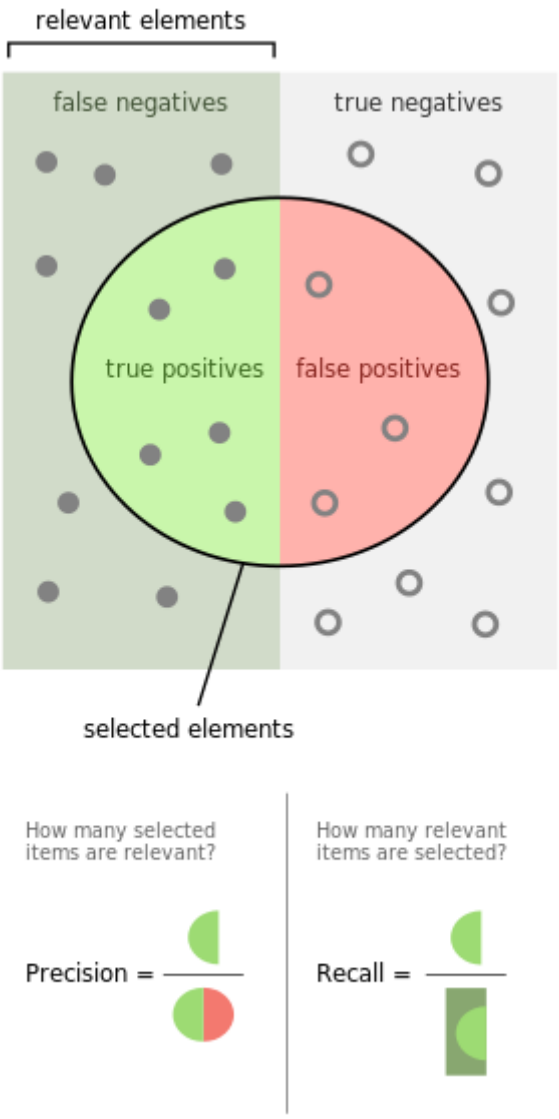
$$R = \frac{TP}{TP + FN}$$

gdje je:

TP - broj točno pozitivnih rezultata
FN - broj pogrešno negativnih vrijednosti

U ovom primjeru preciznost mjeri frakcije tumora za koje je program točno predvidio da su maligni. Opoziv mjeri skup tumora koje je program točno klasificirao kao maligne.

Preciznost i opoziv mogu otkriti da klasifikator s impresivnom razinom točnosti zapravo griješi u detekciji većine malignih tumora. Ako su tumori benigni, čak i klasifikator koji predviđa malignost može imati visoku točnost. Slika 3 prikazuje odnos preciznosti i odziva.



Slika 3. Preciznost i odziv (izvor[5])

Prilikom evaluacije mjerenja korištene su uz točnost s još dvije metode: F1-score i confusion matrix. F1-score je mjera kojom se testira točnost testa. F1-score koristi preciznost i odziv. Može se interpretirati kao težinski prosjek preciznosti i odziva, gdje se 1 interpretira kao njegova najbolja vrijednost, a 0 kao najlošija.

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

Confusion matrix, još znana pod imenom kontigentna tablica (engl.*contingency table*) ili matrica grešaka (engl. *error matrix*), je tablični prikaz koji omogućuje vizualizaciju

performansi algoritma. Koristi se kod algoritama nadziranog učenja. Kod nenadziranog učenja se koristi poklapajuća matrica (engl. matching matrix). Matrica grešaka interpretira se na način da svaki stupac matrice predstavlja instance klase koju pokušavamo predvidjeti, dok svaki red predstavlja instancu stvarne klase. Matrica je zvana „*confusing*“ (engl. zbunjena) zato što je korisniku jednostavno vidjeti je li sustav zamijenio, odnosno pogriješio između dvije klase. Slika 4 prikazuje matricu grešaka.

	p’ (Predicted)	n’ (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Slika 4. Matrica grešaka (izvor [6])

4. SCIKIT-LEARN

Od svog službenog izdanja 2007. godine scikit-learn je postao jedna od najpopularnijih knjižnica otvorenog koda za strojno učenje u programskom jeziku Python. Scikit-learn uključuje algoritme za probleme strojnog učenja kao što su klasifikacija, regresija i grupiranje podataka. Također ima nekoliko modula koji služe za izdvajanje nekih atributa, procesiranje podatka i evaluaciju modela.

Scikit-learn u sebi kombinira nekoliko mogućnosti popularnih Python knjižnica SciPy, NumPy i matplotlib kao što su provođenje efektivnih operacija na velikom skupu polja i matrica i korištenje vizualizacijskih alata.

Scikit-learn je popularan unutar akademske zajednice iz nekoliko razloga: jako je dobro dokumentiran, lagan je za korištenje, i sadrži raznolik API. Računalni programeri mogu koristiti scikit-learn prilikom eksperimentiranja s različitima algoritmima uz sitne modifikacije na svega par linija koda. Scikit-learn također ima u sebi ugrađene neke unaprijed određeneskupove podataka koji omogućavaju programerima da se fokusiraju na algoritme, umjesto da gube vrijeme na prikupljanje i čišćenje podataka.

Licenca kojom se koristi je BSD i može se koristiti u komercijalnim aplikacijama bez ikakvih restrikcija.

5. PRIKUPLJANJE PODATAKA

Prvo što je bilo potrebno učiniti je prikupiti podatke s pet svjetskih burzi. S svake burze bilo je potrebno odabrati deset dionica te jedan do dva burzovna indeksa. Podatke je bilo potrebno prikupiti u što većem povijesnom opsegu, odnosno maksimalnom mogućem. Podacisu prikupljeni iz jednog izvora:s web stranice http://finance.yahoo.com. Ona se pokazala adekvatnom zato što nudi povijesne podatke za većinu svjetskih burzi. Burze koje su odabrane za prikupljanje podataka su: New York Stock Exchange, NASDAQ, London Stock Exchange, Toronto Stock Exchange te Hong Kong Stock Exchange. Popis svih dionica i indeks čiji su podaci prikupljeni se nalaze u tablicama 1,2,3,4 i 5.

New York Stock Exchange	
1.	General Motors Company (GM)
2.	International Business Machine (IBM)
3.	Bank of America Corporation (BAC)
4.	Ford Motor Co. (F)
5.	McDonald's Corp (MCD)
6.	Hewlett-Packard Company (HPQ)
7.	The Coca-Cola Company (KO)
8.	Oracle Corporation (ORCL)
9.	Wal-Mart Stores (WMT)
10.	The Walt Disney Company (DIS)
INDEX: Dow Jones Industrial Average, S&P 500	

Tablica 1. Popisuvrštenihvrijednosnihpapira i korištenihburzovnihindeksa s NYSE-a

NASDAQ	
1.	Microsoft Corporation (MSFT)
2.	Apple Inc. (AAPL)
3.	Cisco Systems,Inc (CSCO)
4.	Intel Corporation (INTC)
5.	Applied Materials, Inc (AMAT)
6.	Mylan N.V. (MYL)
7.	Sirius XM Holdings Inc. (SIRI)
8.	Qualcomm Incorporated (QCOM)
9.	Yahoo! Inc. (YHOO)
10.	Micron Technology, Inc.(Mo)
INDEX: NASDAQ Composite	

Tablica 2. Popisuvrštenihvrijednosnihpapira i korištenihburzovnihindeksa s NASDAQ-a

London Stock Exchange	
1.	Vodafone Group Public Limited Company (VOD.L)
2.	Lloyds Banking Group Plc (LLOY.L)
3.	Clarkson Plc (CKN.L)
4.	Old Mutual Plc (OML.L)
5.	JD Sports Fashion Plc (JD.L)
6.	Premier Oil Plc (PMO.L)
7.	Barclays Plc (BARC.L)
8.	Royal Dutch Shell Plc (RDS.L)
9.	Tesco Plc (TSCO.L)
10.	Aviva Plc (AV.L)
INDEX: FTSE 100	

Tablica 3. Popisuvrštenihvrijednosnihpapira i korištenihburzovnihindeksa s LSE-a

Toronto Stock Exchange	
1.	Royal Bank Of Canada (RY.TO)
2.	Canadian National Railway Company (CNR.TO)
3.	Toronto-Dominion Bank (TD.TO)
4.	Enbridge Inc. (ENB.TO)
5.	The Bank Of Nova Scotia (BNS.TO)
6.	Transcanada Corp (TRP.TO)
7.	Thomson Reuters Corporation (TRI.TO)
8.	Bank Of Montreal (BMO.TO)
9.	Imperial Oil Ltd (IMO.TO)
10.	Loblaw Companies Limited (L.TO)
INDEX:S&P/TSX Composite index	

Tablica 4. Popisuvrštenihvrijednosnihpapira i korištenihburzovnihindeksa s TSE-a

Hong Kong Stock Exchange	
1.	Petrochina (0.857.HK)
2.	Industrial And Commercial Bank Of China (1398.HK)
3.	Bank Of China (3988.HK)
4.	China Mobile (0.941.HK)
5.	Sinopec Corp (0386.HK)
6.	China Shenhua (1088.HK)
7.	Ping An (2318.HK)
8.	China Merchants (0133.HK)
9.	China Telecom (0728.HK)
10.	Sun Hung Kai Co (0086.HK)
INDEX: Hang Seng Index	

Tablica 5. Popisuvrštenihvrijednosnihpapira i korištenihburzovnihindeksa s HKSE-a

5.1 Tehnički indikatori

Podaci nakon što se skinu su u .csv formatu sadrže vrijednosti *open, high, low, close, volume* i *ajd close*. Međutim, ti podaci se nisu koristili kao ulaz za pojedine algoritme koji će se kasnije trenirati, već se se izračunali tehnički indikatori. Popis indikatora i njihove formule se nalaze u tablici 6.

Imena	Formule	Parameteri
5-days and 10-days moving average ($SMA(5)_t, SMA(10)_t$)	$\frac{C_t + C_{t-1} + \cdots + C_{t-n+1}}{n}$	$n = 5, n = 10$
5-days and 10-days weighted moving average ($WMA(5)_t, WMA(10)_t$)	$\frac{(n)C_t + (n-1)C_{t-1} + \cdots + C_{t-n+1}}{(n + (n-1) + \cdots 1)}$	$n = 5, n = 10$
Stochastic %K (K)	$\frac{C_t - LL_n}{HH_n - LL_n} 100$	$n = 14$
Stochastic %D (%D)	$\frac{\sum_{i=0}^{n-1}(K_{t-i})}{n}$	$n = 3$
Moving Average ConvergenceDivergence ($MACD$)	$EMA(fast)_t - EMA(slow)_t$	$fast = 12,$ $slow = 26$
Commodity channel index (CCI)	$\frac{M_t - SM_t}{0.015Dt}$	$n = 20$
5-days and 10-days disparity	$\frac{C_t}{SMA(n)_t} 100$	n=5, n=10
Price oscilator ($OSCP$)	$\frac{SMA(fast)_t - SMA(slow)_t}{SMA(fast)_t} 100$	$fast = 5,$ $slow = 10$
10-days Rate of change	$\frac{C_t}{C_{t-n}} 100$	$n = 10$

Imena	Formule	Parameteri
(ROC)		
10-days momentum	$C_t - C_n$	$n = 10$
Relative strength index (RSI)	$100 - \frac{100}{1 + \left(\frac{\sum_{i=0}^{n-1} Up_{t-i}}{n}\right) / \left(\frac{\sum_{i=0}^{n-1} Dw_{t-i}}{n}\right)}$	$n = 14$
5-days standard deviation	$\sqrt{\frac{\sum_{i=0}^{n-1}(c_{t-i} - SMA(n)_t)^2}{n}}$	$n = 5$

Tablica 6. Tehnički indikatori i njihove formule

C_t je završna cijena (engl. closing price) dionice ili indeksa prilikom zatvaranja burze. H_t predstavlja najvišu cijenu koja je postignuta u danu *t*. L_t predstavlja najnižu cijenu koja je postignuta u danu *t*. HH_n označava najvišu cijenu u periodu od *n* dana, a LL_n predstavlja najnižu cijenu u periodu od *n* dana. $EMA(n)_t = EMA(n)_{t-1} + \alpha$ predstavlja polirajući faktor *2/(n+1),* gdje je *n* broj koji se koristi prilikom izračuna eksponencijalnog kretajućeg prosjeka. M_t je tipična cijena, $Mt = \frac{Ct+Lt+Ht}{3}$, SM*t* je jednostavan kretajuću prosjek tipičnih cijena u rasponu od *n* dana, a Up_t i Dw_t su promjene cijena tokom dana prema gore ili prema dolje[7].

5.1.1 Opis indikatora

SMA je jednostavni kretajući prosjek (engl. Simple moving average). Izračunava se na uzorku od 5 i 10 dana. Koristi se za poliranje kretanja vrijednosti signala pojedine dionice, čineći ga lakšim za čitanje investitoru prilikom identifikacije trendova (rast, pad).

%K i **% D** su stohastički oscilatori. %K se koristi prilikom usporedbe završne cijene u trenutku t s visoko-niskim rasponom tokom perioda od n dana. %D je SMA od %K u periodu od n dana. Vrijednosti stohastičkih oscilatora variraju od 0 do 100. Ako je njihova vrijednost veća od 80, smatra se da su se dionice previše kupovale. Ako je vrijednost manja od 20, smatra se da su se dionice previše prodavale.

MACD (engl. Moving average convergance divergance) se izračunava tako što se oduzme brzi eksponencijalni kretajući prosjek od sporog. Često se koristi u kombinaciji s prethodno navedenim tehničkim indikatorima prilikom procjene treba li neke dionice kupiti ili prodati.

CCI (engl. Commodity chanel index) se koristi za detekciju dali je neka dionica previše puta kupljena ili previše puta prodana. Izračunava se tako da se oduzme tipična cijena unutar n dana od SMA, te ta vrijednost podijelis prosječnom vrijednosti apsolutne devijacije tipične cijene. Ako je CCI vrijednost veća od 100, smatra se da je dionica previše puta kupljena. Ako je CCI vrijednost manja od -100 , smatra se da je dionica previše prodana.

RSI (engl. Realtive strenght index) se također koristi za detekciju je li neka dionica previše puta kupljena ili previše puta prodana. Njegove vrijednosti variraju između 0 i 100. Ako je vrijednost RSI veća od 70, smatra se da je dionica previše puta kupljena, a ako je RSI vrijednost ispod 30 smatra se da je dionica previše puta prodana.

Raspršenost (engl. Disparity) i ROC (engl. Rate of change) su slični indikatori. Raspršenost je omjer između cijene na zatvaranju u vremenu t i SMA u periodu od n dana. ROC je je omjer završne cijene u trenutku t i završne cijene u trenutku t-n. Ako su vrijednosti oba indikatora iznad 0, to predstavlja signal investitorima da kupe odabranu dionicu. No međutim ako su vrijednosti indikatora ispod 0, to je signal investitorima da je dionicu bolje što prije prodati.

Price oscillatorje tehnički indikator koji prikazuje razliku između brzog i sporog kretajućeg prosjeka, podijeljenog s brzim kretajućim prosjekom. Generira signal za kupnju ako je vrijednost brzog kretajućeg prosjeka veća od vrijednosti sporog kretajućeg prosjeka. U suprotnome generira signal za prodaju.

Momentum se izračunava kao razlika između cijene dionica u trenutku t i cijene dionica u trenutku t-n. Negativna vrijednost momentuma se često koristi od strane investitora prilikom generiranja signala za kupnju, a pozitivna vrijednost momentuma prilikom generiranja signala za prodaju.

Standardna devijacija se koristi prilikom mjerenja nekonzistentnosti u periodu od n dana.

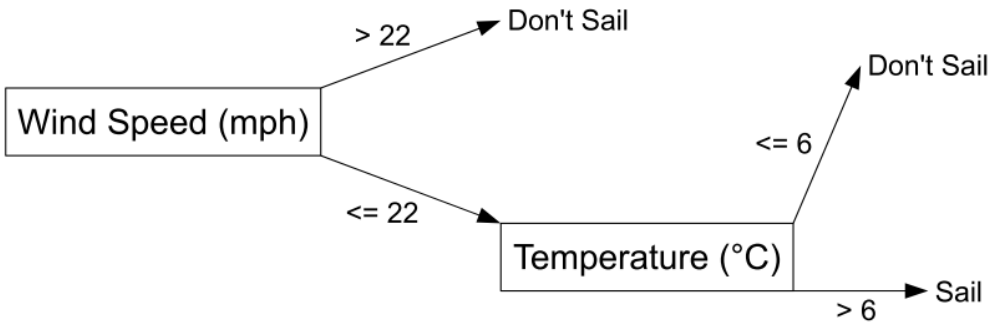
Prilikom izračunavanja smjera dionice za period unaprijed 1, 5, i 10 dana je izračunat na sljedeći način. Primjerice za period unaprijed 5 dana, ako je closing cijena u trenutku t+5 veća nego što je bila u trenutku t, onda se ta vrijednost označava s 1, u suprotnome se značava s 0.

Tehnički indikatori su se izračunali koristeći Python programski paket pandas. Pandas je softverska knjižnica napisna za programski jezik Python. Služi za analizu i manipulaciju velikim skupovima podataka.

6. TEORIJSKI OPIS ALGORITAMA KOJI SU SE IMPLEMENTIRALI

6.1 Stablo odluke

Stablo odluke (engl. Decision tree) je algoritam za učenje koji stvara strukturirani klasifikacijski model tijekom inicijalnog treniranja na kojem se baziraju predikcije. Takve modele se smatra hitrima budući da se svi izračuni koji se odnose na buduće predikcije izvršavaju u periodu treniranja. Rekurzivna metoda od vrha prema dnu se najčešće koristi prilikom konstrukcije stabla. To je znano kao podijeli-pa-vladaj pristup (engl. divine-and-conquer) i može se riješiti rekurzivno. Korijen (engl. root) se grana prema atributima koji najbolje razdvaja podatke u homogene podskupove s istom klasom. Razne metode postoje prilikom određivanja tog atributa; međutim, najpoznatiji algoritmi koriste ili entropiju ili Gini index bazirana mjerenja. Na slici 5je primjer stabla odluke.



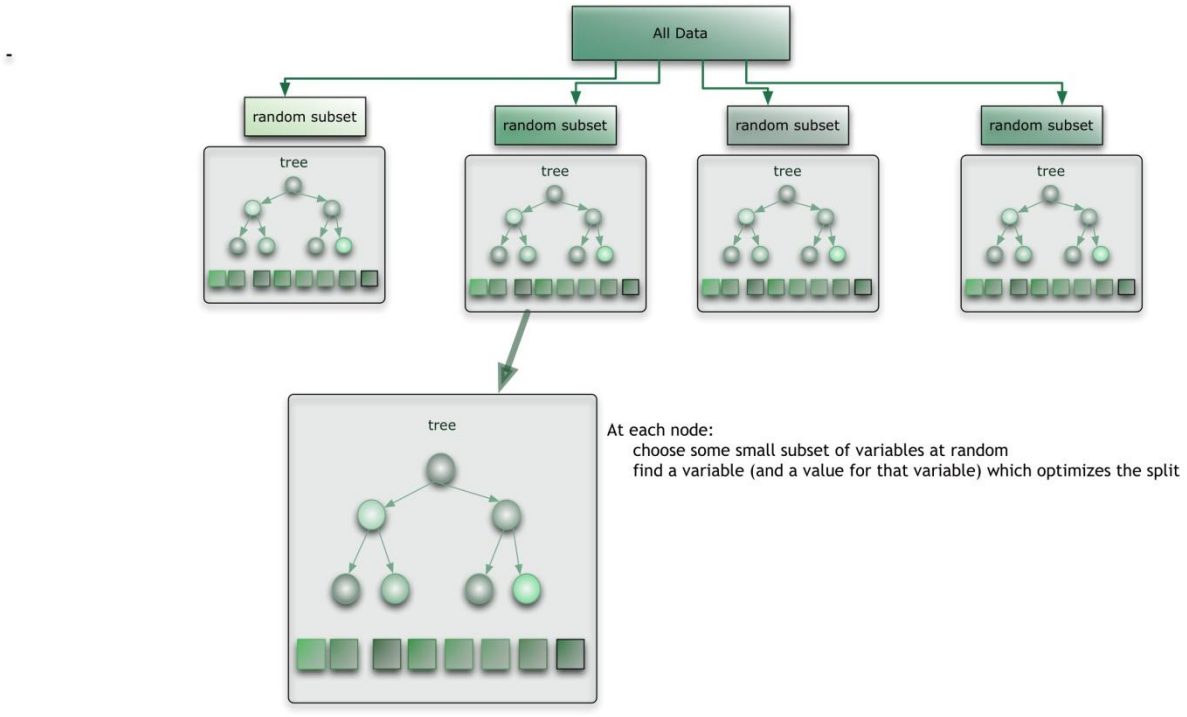
Slika 5. Jednostavan opis rada stabla odluke(izvor [8])

Slika prikazuje primjer mogućega stabla odluke kojeg bi pomorci mogli koristiti prilikom odluke je li je sigurno ploviti ili nije pod određenim uvjetima. Na slici se može vidjeti da je brzina vjetra (engl. wind speed) atribut na čvorištu, a da je njegov testni uvjet to je li veći ili

manji od 22 (vrijednost brzine vjetra izražena u miljama na sat). Ako je, predikcija je da se ne plovi (engl. Don't Sil). Ako nije, testni uvjet na sljedećem čvoru se evaluira. Taj proces se ponavlja sve dok se ne postigne predikcija. Sukladno tome, svaka predikcija se može gledati kao spoj jednog ili više testnih uvjeta.

6.2 Nasumična šuma

Ideja nasumične šume (engl. random forest) je da generira nekoliko stabala odluke iz nasumično odabranih podskupova podataka. Na svakom čvoru se definira određen broj varijabli zaduženih za predviđanje. Predikcijska varijabla koja postigne najbolju podjelu podataka, mjereno od strane nekih objektivnih funkcija, se koristi prilikom binarne podjele na tom čvoru. Na sljedećem čvoru se odabere drugi broj predikcijskih varijabli i ponovi se postupak. Na slici 6. je prikazan primjer rada nasumične šume.

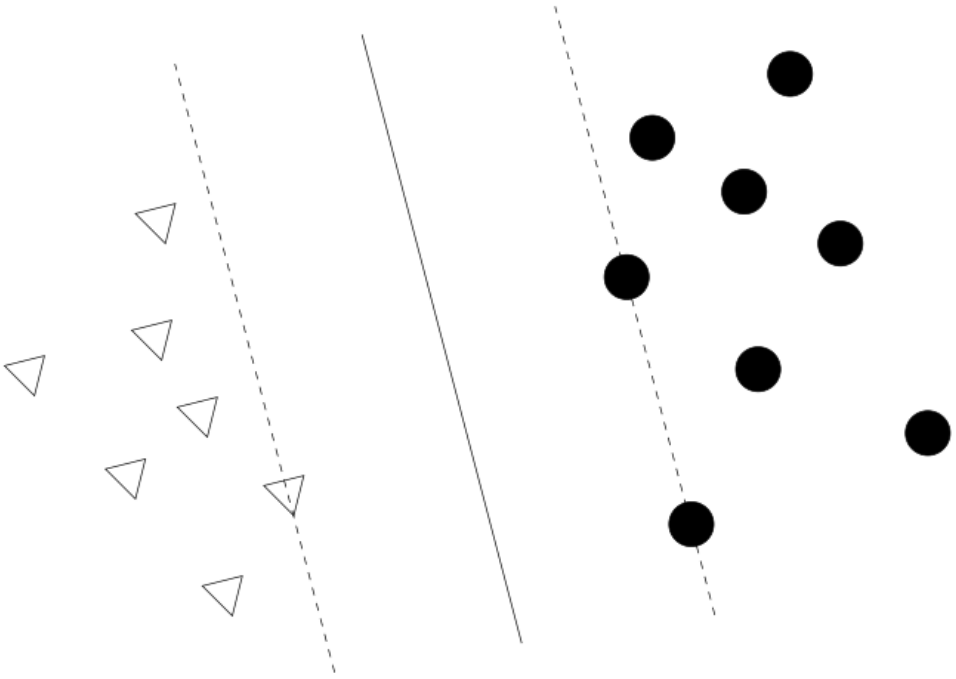


Slika 6. Opis rada nasumične šume(izvor [9])

6. 3 Stroj s potpornim vektorima

Stroj s potpornim vektorima (engl. Support vector machines - SVM) su skup blisko povezanih i složenih algoritama za strojno učenje koji se mogu koristiti i kod klasifikacije i kod regresije. SVM se koristi u mnogim klasifikacijskim domenama kao što su kategorizacija teksta, predikcija spajanja DNA baza, prepoznavanje lica. Na primjeru klasifikacijskog trening skupa, u kojem je svaka vrijednost mapirana kao pripadnik jedne ili druge klase, SVM algoritam kreira model koji dodjeljuje nove primjere u jednu ili drugu kategoriju. Sam model se reprezentira kao točke unutar ravnine, mapirane na način da su primjeri suprotnih kategorija odvojeni što većom prazninom. Novi primjeri se potom mapiraju u isti prostor u kojem su određeni procjenom.

Slika 7 prikazuje klasifikacijski problem u kojem je puna linija separator. Razlika između isprekidanih linija se naziva margina. Tri točke koje su smještene na isprekidanoj liniji se definiraju kao potporni vektori. Te točke zapravo definiraju separatora, ostale nisu zabilježene u modelu.



Slika 7. Ilustracija rada SVM-a(izvor [8])

Prednosti SVM-a su: efektivni u prostorima velikih dimenzija, dobri su u slučajevima kada je broj dimenzija veći nego broj uzoraka, štede memoriju zbog podskupa trening podataka u funkciji odlučivanja (potporni vektori), mogu se primjeniti na mnoge probleme, nude nekoliko dobrih opcija kernel funkcija.

Nedostaci SVM-a su: rezultati će biti lošiji ako je broj značajki puno veći od broja uzoraka, ne pružaju dobre procjene točnosti, preporuča se upotreba u kombinaciji s primjenom unakrsne točnosti.

7. IMPLEMENTACIJA UNUTAR KODA

7.1 Učitavanje podataka

Prvo što je potrebno napraviti je unos podataka u scikit-learn. Tu se i dalje nastavio koristiti pandas library koji je prvotno bio upotrebljen prilikom izračuna tehničkih indikatora. Metoda rada bila je da se prvo učitaju svi podaci iz csv datoteke, a potom da se izvrši podjelu na značajke i na klase. Učitavanje svih podataka iz csv datoteke opisuje sljedeća linija koda:

```
data = pd.read_csv('D:\diplomski_kod\AAPLtestni.csv')
```

7.2 Podjela na značajke i klase

Prilikom podjele na značajke i klase pratila se službena konvencija imenovanja unutar scikit-learna. Značajke se označavaju s 'X' , dok se klasa označava s 'y'. Kao značajke su odabrani svih tehnički indikatori; taj odabir značajki se vrši dva puta. Jednom su odabrane nediskretizirane vrijednosti dok su drugi puta odabrane vrijednosti koje su prethodno diskretizirane. Također, i klasaje odabrana tri puta. Za Direction1 koji predstavlja smjer pada ili rasta u trenutku „t + 1“, Direction5 koji predstavlja smjer pada ili rasta u trenutku „ t + 5“ i za Direction10 koji predstavlja smjerpada ili rasta „t + 10“ . Sljedeće linije koda to opisuju:

```
X = data[['RSI', 'CCI', '5SMA', '10SMA', 'ROC', 'Momentum', '%K', '%D', 'Standard Deviation', 'Disparity5', 'Disparity10',
'OSCP', '5WMA', '10WMA']].astype(np.float32)
```

```
y = data['Direction1'].astype(np.float32)
```

Nakon što smo učitali podatke u format koji nam odgovara te nakon što smo napravili podjelu na značajke i klase, daljnji postupak se sastoji u nekoliko koraka koji će se prikazati na konkretnom primjeru korištenja.

Korak 1: Import klase odnosno tipa algoritma koji se koristiti

```
from sklearn.ensemble import RandomForestClassifier
```

Korak 2: Instanciranje „estimatora“ (scikit-izraz za model)

```
rf = RandomForestClassifier(n_estimators=100)
```

U ovoj liniji također je moguće namještanje parametara. Ako se to ne učini, parametri su namješteni na svoje pretpostavljene vrijednosti. Trenutno stanje parametara se može jednostavno vidjeti s sljedećom komandom:

```
print rf
```

Koja kao output daje:

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',

max_depth=None, max_features='auto', max_leaf_nodes=None,

min_smples_leaf=1, min_smples_split=2,

min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,

oob_score=False, random_state=None, verbose=0,

warm_start=False)

7.3 Odabir načina testiranja, treniranja i evaluacije

Na sljedećem koraku se vrši podjela na train-test podjele. Ovo je samo opis jednog od tri načina testiranja i treniranja kojasu bila provedena radi prikaza na ovom primjeru. Druga dva su bili testiranje i treniranje na istom skupu podatka i 10-dijelna unakrsna točnost. Ovo je trenutno opis train-test podjele u omjeru 80:20. Sljedeća linija koda opisuje tu podjelu:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

Korak 3: Treniranje modela (punjenje modela s podacima). U ovom koraku model uči vezu između X i y odnosno značajki i klasa. Ova se operacija odvija „u mjestu“ tako da se ne mora dodijeljivati drugom objektu:

```
rf.fit(X_train,y_train)
```

Korak 4: Vršenje predikcija za nove observacije.U sljedećoj liniji zapravo se testira što je model naučio u prethodnom koraku, koristeći njemu nepoznate podatke.

```
output = rf.predict(X_test)
```

„Output“ predstavlja polje prediktivnih vrijednosti. Točnost pretpostavke se izračunava na sljedeći način:

```
print metrics.accuracy_score(output, y_test)
```

7.4 Metodologija pokusa

Sve do sada navedeno je bio primjer koraka koje je potrebno poduzeti da bi se neki podaci testirali i trenirali s određenim algoritmima i validirali na određene načine. Zadatak ovog rada je bio nešto opsežniji. Bilo je potrebno koristiti tri različita algoritma. Koristili su se stablo odluke, nasumična šuma te stroj s potpornim vektorima. Nakon što se izvršilo treniranje i testiranje algoritama bilo je potrebno izmijeriti nekoliko metrika. Neke od metrika koje je bilo potrebno implementirati su: točnost klasifikacije, F1-score i matrica grešaka.

Testiranje se vršilo na dva skupa podatka. Tehnički indikatori su uzeti kao značajke u oba slučaja, međutim u jednom su bili s diskretiziranim vrijednostima, a u drugom s nediskretiziranim vrijednostima. Vektori klasa su bili Direction1, Direction5 i Direction10 .Na tih šest permutacija skupa podataka za svaki su se izvršili navedeni algoritmi i tražene metrike. Načini testiranja su se također razlikovali. Oni su se razlikovali u određivanju koji podaci će služiti za treniranje a koji za testiranje. Podjela se vršila na sljedeći način:

a) Empirijska točnost (testiranje na istim podacima na kojima se model i učio)
b) Podijela podataka na train/test u omjeru 80:20

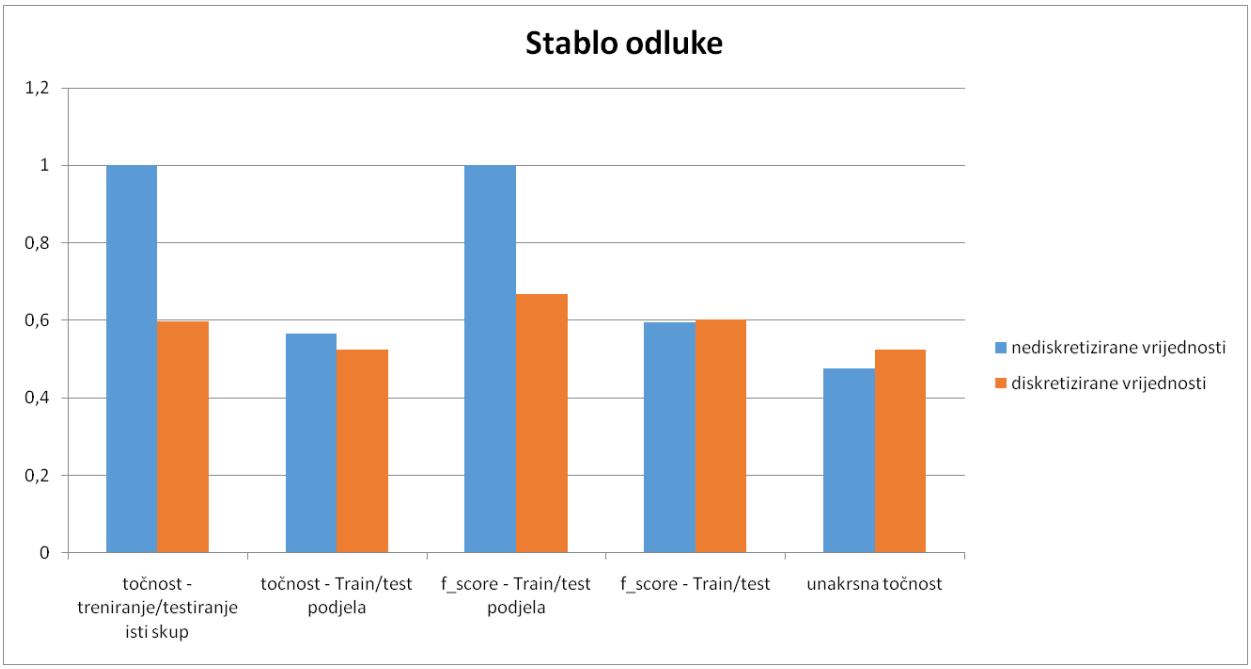
c) 10-dijelna unakrsna točnost (s randomiziranim podacima)

Cjelokupni proces se odvijao na nekoliko slojeva. Za svaku burzu, a unutar nje za svaki indeks odnosno simbol. Izračunala se aritmetička sredina svih dobivenih metrika za simbol/indeks. Na razini pojedine burze i na razini svih burzi se je također izračunala aritmetička sredina i standardna devijacija koje su se potom zapisivale u zasebne datoteke za svaku burzu te u jednu datoteku za sve burze, odnosno za sve podatke koji su se istestirali.

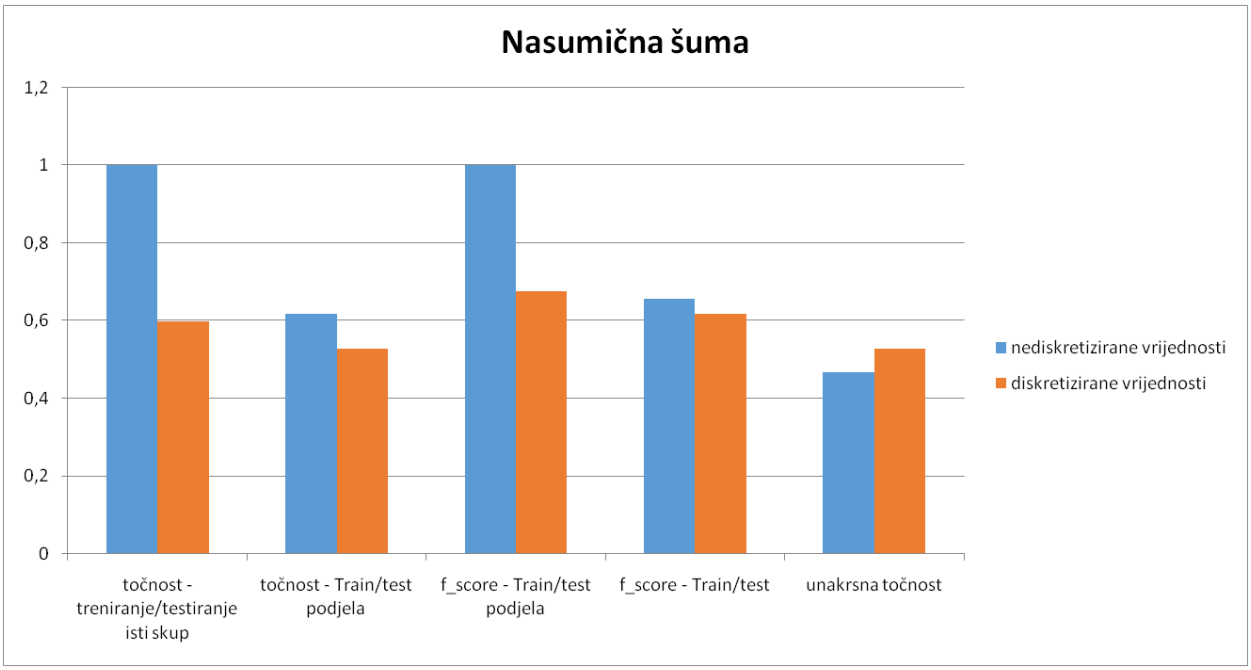
Svi dobiveni rezultati su se zapisivali u .txt datoteke. Tako primjerice imamo za svaki simbol/indeks svake burze zasebnu tekstualnu datoteku koja sadrži podatke o svemu navedenom.

8. DOBIVENI REZULTATI

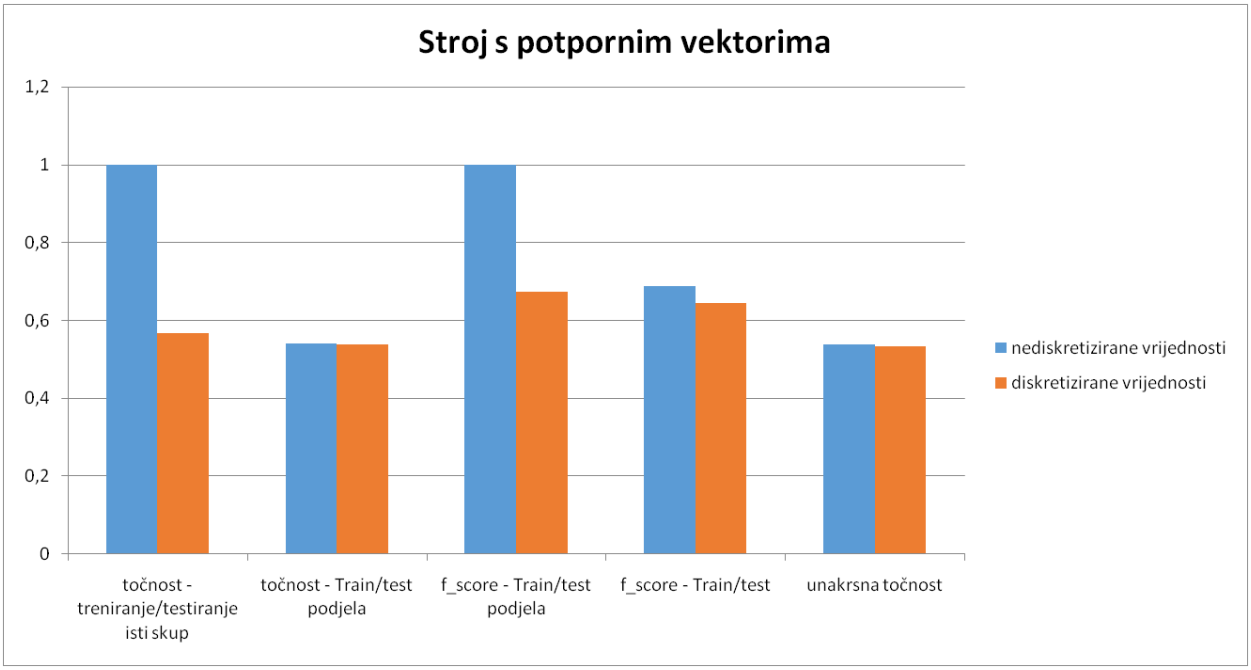
Slike 8,9 i 10 prikazuju dobivene metrike pojedinih algoritama, na skupu diskretiziranih i nediskretiziranih podataka. Metrike obuhvaćaju podatke svih burzi i svih dionica i indeksa.



Slika 8. Prikaz dobivenih rezultata korištenjem stabla odluke



Slika 9. Prikaz dobivenih rezultata korištenjem nasumične šume



Slika 10. Prikaz dobivenih rezultata korištenjem stroja s potpornim vektorima

Iz dobivenih rezultata možda se činikako su najbolji rezultati dobiveni kada se model trenirao i testirao na istim podacima. Međutim, to je jedan od poznatih problema strojnog učenja u kojem se model „pretrenira“. Cilj kojem bi svaki model trebao težiti je postizanje dobre generalizacije [10].Analizirajući rezultate preostala dva načina evaluacije: podjele podataka na skup za treniranje i skup za testiranje te unakrsne točnosti možemo zaključiti kako rezultati variraju. Promatrajući rezultate u kojima se boljom pokazala metoda podjele podataka na skup za treniranje i skup za testiranje, možemo zaključiti da uzrok tome vjerojatno leži u činjenici da postoji određena razlika između skupa podataka na kojima smo model trenirali i skupa podataka na kojima smo model testirali. Također, na osnovu toga možemo zaključiti da je ipak najbolja metoda unakrsne točnosti budući da ona otklanja problem u kojem jedan dio podatka vrijednostima značajno odskače od drugog. Metoda diskretizacije podataka se također pokazala kao dobrim sredstvom izjednačavanja razlika između podataka i samim time su rezultati dobiveni trening test podjelom i unakrsnom točnosti doveli do veoma sličnih vrijednosti.

9. ZAKLJUČAK

Strojno učenje je zasigurno jedna od znanstvenih disciplina koja će u budućnosti najviše napredovati. Od internetskih tražilica do autonomnih vozila strojno učenje je svuda oko nas i njegov utjecaj će biti sve veći i veći. U ovom radu razmatralo se o mogućnostima predikcije pada ili rasta burzovnih dionica gdje strojno učenje također uzima sve veći zamah. Provela se implementacija i testiranje pojednih algoritama i njihove učinkovitosti na uzroku od deset dionica i indeksa sa pet svjetskih burzi. Za programiranje same aplikacije odnosno potrebnih skripti se koristio programski jezik Python. Skripte su logički razdvojene u 3 dijela. Kada interpretiramo rezultate koje smo dobili nakon izvršavanja cijelokupnog testiranja možemo zaključiti da su se pojavile određene zakonitosti koje su već dobro istražene i dokumentirane u svijetu strojnog učenja. Jedna od njih je pretreniranost modela koja se pojavila u situaciji kada se treniranje i testiranje modela odvijalo na istom skupu podatka. Model se previše prilagodio trenutnim podacima te se pojavila klasična greška, a to je da model ne generalizira dobro. Što se tiče ostalih načina testiranja, rezultati su u pokazali da se točnost može mjeriti sa postotkom od otprilike 60%. Ako se uzmu u obzir rezultati dobiveni s treniranjem i testiranjem na istom skupu podatka postotak točnosti tada značajno skače. Na razini pojedinih burzi i na razini cijelokupnog testiranja također su se provela mjerenja aritmetičke sredine točnosti svih metrike i izmjerena je standardna devijacija. Rezultati svih mjerenja su dobro dokumentirani i zapisani u nekoliko različitih datoteka. Za svaku dionicu su zabilježene sve vrijednosti dobivene mjerenjem, zabilježeno je testiranje na diskretiziranim i nediskretiziranim vrijednostima podatka kao i pripadajuće metrike. Podaci su se zapisivali i u .txt datoteku radi preglednosti čitanja i u .csv datoteku u svrhu statističkog mjerenja i prikaza razultata kasnije. Cjelokupno testiranje cijelog sustava vremenski ovisi o brojnim faktorima, među kojima su: mogućnosti računala na kojem se simulacija izvršava i na količini podatka koja se obrađuje. Uz sve navedeno, čak i na računalima sa boljim konfiguracijama trajanje cijelokupne simulacije može trajati do nekoliko sati. To se situacija djelomično mogla izbjeći da se simulacija programirala na način da se može izvršavati na način koji omogućuje paralelno računanje. U istraživanju rezultata dobivenih sličnim mjerenjima iz postojećih znanstvenih članaka može se zaključiti da još uvijek nije došlo do neke metode koja omogućuje stabilnu predikciju rasta ili pada burze. Razlog tome je vjerojatno to što je burza veoma dinamičan prostor sa brojnim neočekivanim usponima i padovima koji ovisi o mnogo čimbenika koji veoma često nisu predvidljivi.

10. LITERATURA

[1] Chi-Yuan Yeh, Chi-Wei Huang, Shie-Jue Lee, „A multiple-kernel support vector regression approach for stock market price forecasting“

[2] Chinook: http://webdocs.cs.ualberta.ca/~chinook/project/legacy.html

[3] Dang Trung Thanh, Kiyoaki Shirai, „Machine Learning Approaches for Mood Classication of Songs toward Music Search Engine“

[4] Gavin Hackeling, „Mastering Machine Learning with scikit-learn“

[5] Wikipedia, https://en.wikipedia.org/wiki/Precision_and_recall

[6] Tools for Machine Learnign Performance Evaluation
http://aimotion.blogspot.hr/2010/08/tools-for-machine-learning-performance.html

[7] Teo Manojlović, Ivan Štajduhar, „Predicting Stock Market Trends Using Random Forest: A Sample of the Zagreb Stock Exchange“

[8] David Gray, „Software Defect Prediction Using Static Code Metrics : Formulating a Methodology“

[9]A Gentle Introduction to Random Forest: https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

[10] Pedro Domingos, „A Few Useful Thing to Know about Machine Learning“

SAŽETAK

Napravila se sveobuhvatna analiza pet svjetskih burzi sa po deset dionica i sa nekoliko indeksa na svakoj od tih burzi. Podaci su se prikupili sa stranice Yahoo Finance koja sadrži sve povijesne podatke najvećih svjetskih burzi. Nakon što su podaci skinuti u csv formatu izvršio se izračun tehničkih indikatora. Programski jezik koji se koristio je Python 2.7. Nakon izračuna tehničkih indikatora provela se diskretizacija podatka. Algoritmi su kasnije koristili i diskretizirane i nediskretizirane vrijednosti. Nakon izračuna tehiničkih indikatora vrši se treniranje i testiranje algoritama. Algoritmi koji su se obuhvatili su stablo odluke, nasumična šuma te stroj s potpornim vektorima. Metrike koje su potom bile izvršene su *F-score*, *confusion matrix* te unakrsna validacija. Podaci su spremljeni zasebno za svaku burzu u .txt obliku koji obuhvaća tekstualni zapis svih metrika, svih načina treniranja i testiranja i zapis svih rezultata. Također su se podaci spremali i u csv formatu čime se olakšala kasnija statistička obrada. Cijeli kod se izvršava po slojevima: za svaku burzu te za svaki indeks.

Ključne riječi: strojno učenje, burza, dionica, indeks, Python, predikcija, algoritmi, metrike

DODATAK A

U ovom dijelu je opisan proces instalacije svih alata koji su se koristili prilikom pisnja koda. Navest će se procedura za operativni sustav Windows 7.

Prvo što je potrebno je skinuti i instalirati sam Python. Najlakše je to učiniti s službene stranice www.Python.org . Ona nudi 2 verzije Pythona; Python 3.4.3 i verziju 2.7.10. U kodu je korištena Python verzija 2.7.10. Nakon što se Python instalira potrebno ga je unutar Windows sustava dodati kao environment varijablu. To se može učiniti na sljedeći način: desni klik na ikonu Computer -> Properties -> Advanced system skuptings i odabrati Environment Variables. Među sistemskom varijablom „Path“ potrebno je dodati putanju do mjesta gdje smo prethodno instalirati Python. Ako su prilikom instalacije ostavljenje defaultne vrijednosti Python bi se trebao nalaziti na sljedećoj ili sličnoj putanji C:\Python27. Radi provjere uspješnosti instalacije dovoljno je otići u Command Prompt i upisti „Python“. Sme skripte se mogu pokrenuti na taj način. Iz Command Prompta se navigira u direktorij gdje je skripa i pokrene se komanda „Python ime_skipte“.

Nakon uspješne instalacije Pythona potrebno je instalirati pakete s kojima se radilo prilikom pisnja programskog koda. Prije svega je potrebno instalirati package manager pip ukoliko je instalirana verzija Pythona manja od verzije 2.7.9. Nakon nje pip dolazi predinstaliran. Za pakete su skinute binarne datoteke s web stranice http://www.lfd.uci.edu/~gohlke/Pythonlibs/ . Tu su instalirani paketi **numpy-1.9.2**, **pandas-0.16.2**, **scikit_learn-0.16.1,TA_Lib-0.4.9** te **scipy-0.16.0**. Prilikom odabira paketa važno je obratiti pažnju na to je li operativni sustav 32-bitni ili 64-bitni. Nakon što se svi paketi skinu, postupak konfiguracije nalaže da se pokrene Command Prompt i pozicionira na lokaciju C:\Python27\Scripts. Potom se pokrene pip package manager s komandom „pip install put_do_paketa“.

Nakon što su svi instalati uspješno nemješteni kod se može pokrenuti. Tokom razvijanja koda se kao razvojna okolina koristio PyCharm. No međutim, on nije potreban prilikom izvođenja skripte već je služila kao alat i razvojna okolina radi jednostavnijeg kodiranja. PyCharm se može besplatno skinuti s lokacije https://www.jetbrains.com/pycharm/.