Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Janez Bindas

Napovedovanje vzporednih časovnih vrst s strojnim učenjem

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: prof. dr. Igor Kononenko

Ljubljana, 2018

Zahvala

Na tem mestu bi se rad zahvalil mentorju prof. dr. Igorju Kononenku za strokovnost, prijaznost in potrpežljivost med nastajanjem magistrskega dela.

Janez Bindas, 2018

Moji mami.

"/. . . / včasih potrebuješ tiho kuhinjo, skodelico kave in mamo."

— Pam Brown

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
NB	Naive Bayes	naivni bajes
KNN	K Nearest Neighbours	K najbližjih sosedov
DT	Decision Trees	odločitvena drevesa
SVM	Support Vector Machine	metoda podpornih vektorjev
NN	Neural Network	nevronska mreža
RF	Random Forest	naključni gozdovi

Povzetek

Naslov: Napovedovanje vzporednih časovnih vrst s strojnim učenjem

Magistrsko delo obravnava napovedovanje vzporednih časovnih vrst s strojnim učenjem. Vzporedne časovne vrste so časovne vrste, katerih vrednosti se spreminjajo v času ob enakih časovnih intervalih, kot je npr. ura ali dan istočasno za vse časovne vrste. Primer take časovne vrste so borzni tečaji, kjer se za vsak vrednostni papir posebej tvori ena časovna vrsta vzporedno s časovnimi vrstami ostalih vrednostnih papirjev.

Doprinos magistrske naloge je v novem kombiniranem algoritmu za napovedovanje vzporednih časovnih vrst, ki vsebuje genetski algoritem in nelinearno regresijo. Genetski algoritem je uporabljen za iskanje sita in nelinearne funkcije, ki opisujeta model. Za izračunanje neznanih koeficientov funkcije se uporablja numerična metoda nelinearne regresije.

Novo predlagani algoritem je primerljiv glede točnosti in mere dobička z obstoječimi algoritma strojnega učenja. Prednost je da za vhodne podatke ne rabi posebnih predobdelav podatkov, dokler so le ti polni. Druga prednost je tudi, da ponuja razlago, kako so podatki odvisni med sabo. Slabost algoritma pa je njegova časovna potratnost, ki smo se ji v delu delno izognili s paralelizacijo.

Ključne besede

genetski algoritmi, nelinearna regresija, predobdelava podatkov, konstrukcija atributov, vzporedne časovne vrste

Abstract

Title: Prediction of Parallel Time Series with Machine Learning

This master's thesis deals with the prediction of parallel time series with the use of machine learning. Parallel time series are time series whose values change over time at equal time intervals, such as hour or day simultaneously for all time series. An example of this type of time series are stock exchange rates, where for each security a single time series is created parallel to the time series of other securities.

The contribution of this master's thesis is a new combined algorithm for predicting parallel time series that includes a genetic algorithm and nonlinear regression. The genetic algorithm is used to find the sieve and for nonlinear functions that describe the model. The numerical method of nonlinear regression is used to calculate unknown function coefficients.

The new proposed algorithm is comparable in terms of accuracy and profit margin with existing machine learning algorithms. The advantage of the algorithm is that it does not need specific data preprocessing for input data as long as data are complete. Another advantage is that it offers an explanation on how data depend on one another. The downside is that the algorithm is time consuming, which was partly avoided with parallelism.

Keywords

genetic algorithms, nonlinear regression, data prepocessing, attribute constructions, parallel time series

Poglavje 1

Uvod

Magistrsko delo obravnava možno izboljšavo napovedovanja vzporednih časovnih vrst s strojnim učenjem. Vzporedne časovne vrste so časovne vrste, katerih vrednosti se spreminjajo v času ob enakih časovnih intervalih, kot je na primer ura ali dan, istočasno za vse časovne vrste. Primer take časovne vrste so borzni tečaji, kjer se za vsak vrednostni papir posebej tvori ena časovna vrsta vzporedno s časovnimi vrstami ostalih vrednostnih papirjev.

Borza je organizirana gospodarska družba, ki predstavlja organiziran kraj, kjer se srečujeta ponudba in povpraševanje po vrednostnih papirjih. Osnovni namen borze vrednostnih papirjev je omogočiti organizirano, transparentno, likvidno in učinkovito poslovanje z vrednostnimi papirji in drugimi finančnimi instrumenti v skladu z zakoni in predpisi.

Številne raziskave s področja napovedovanja finančnih časovnih vrst so bile narejene z različnimi algoritmi strojnega učenja. Napovedovanje takih sistemov je pogojeno s šumom, intenzivnostjo, nestacionarnostjo, negotovostjo in številnimi skritimi relacijami. Vsekakor gre za zelo izzivalno raziskovalno področje, ki so ga obravnavali številni raziskovalci [12].

Od ustanovitve borze so ljudje zaslužili velike dobičke in po drugi strani prav tako tudi velike izgube. Posledično so razni investitorji iskali orodja in tehnike napovedovanja cen vrednostnih papirjev. Izkaže se, da metode, za katere bi lahko rekli, da so optimalne, ne obstajajo. Prostor časovnih vrst je

namreč še vedno prevelik, da bi lahko uporabili vse možne kombinacije, ki bi jih uporabili za napovedovanje [12].

1.1 Cilji in namen

Magistrska naloga skuša odgovoriti na naslednja vprašanja:

1. Ali je mogoče napovedovati vzporedne časovne vrste (vrednostni papirji)?
2. Katere metode strojnega učenja so boljše po točnosti in katere po časovni potratnosti?
3. Katera metodologija primerjanja algoritmov je najboljša?
4. Kako dober je novo predlagani kombiniran algoritem, ki je sestavljen iz genetskega algoritma in nelinearne regresije?
5. Na kakšen način lahko pohitrimo kombiniran algoritem?

Glavni cilj magistrske naloge je odgovoriti na zgoraj omenjena vprašanja. Vsekakor pa je namen naloge preizkusiti dobljene rešitve na realnih podatkih in z realnimi vložki v trgovanju z vrednostnimi papirji.

1.2 Pregled strukture magistrskega dela

- V 2. poglavju je podan kratek pregled najbolj tipičnih metod strojnega učenja za napovedovanje časovnih vrst in pregled opravljenega raziskovalnega dela na področju napovedovanja časovnih vrst.
- Poglavje 3 je namenjeno razlagi novo predlaganega kombiniranega algoritma, ki je sestavljen iz genetskega algoritma in nelinearne regresije.
- V poglavju 4 je opisana metodologija za primerjanje algoritmov po točnosti in po časovni potratnosti, ki se uporablja v magistrski nalogi.

- Primerjava in analiza dobljenih podatkov sta opisani v poglavju 5.
- Sklepne ugotovitve so predstavljene v poglavju 6.

Poglavje 2

Pregled področja

Strojno učenje lahko kot del umetne inteligence definiramo kot učenje stroja iz podatkov [4]. Včasih ne moremo izluščiti informacije ali vzorca iz podatkov. V teh primerih je strojno učenje nepogrešljiva metoda, ki nam lahko pomaga pri napovedovanju ali klasifikaciji [4].

Učenju živih sistemov pravimo naravno učenje, če pa je učenec stroj – računalnik, rečemo takšnemu učenju avtomatsko ali strojno učenje [7]. Po širši definiciji strojno učenje vključuje vsak računalniški program, ki izboljša svojo izvedbo določene naloge na podlagi izkušenj [7].

Osnovni princip strojnega učenja je avtomatsko opisovanje (modeliranje) pojavov iz podatkov. Rezultati učenja iz podatkov so lahko na primer pravila, funkcije, relacije, sistemi enačb in verjetnostne porazdelitve. Naučeni modeli skušajo razlagati podatke, iz katerih so bili modeli narejeni, in lahko jih uporabijo za odločanje pri opazovanju modeliranega procesa v prihodnosti (napovedovanje) [4, 7].

Glede na vrste učenja strojno učenje razvrstimo na [7]:

- **Nadzorovano učenje (ang. *Supervised Learning*):** Algoritem uči stroj s podanimi pari vhodnih in želenih izhodnih podatkov. Pri tem želene izhode vrednosti določa učitelj oz. človek – nadzornik.
- **Nenadzorovano učenje (ang. *Unsupervised Learning*):** Algoritem razdeli dane vhodne podatke po svojih kriterijih v več kategorij, ki

imajo svojo značilnosti. Takšen pristop se imenuje rojenje (ang. *clustering*). Pri tem število kategorij in njihove značilnosti algoritem izlušči iz vhodnih podatkov, brez nadzora učitelja.

- **Delno nadzorovano učenje (ang. *Semi-Supervised Learning*):** Algoritem združi tako nadzorovano kot nenadzorovano učenje [4], pri čemer so neoznačeni podatki prisotni, za označevanje podatkov pa je proces dolgotrajen.

- **Spodbujevalno učenje (ang. *Reinforcement Learning*):** Algoritem uči z nagrajevanjem in kaznovanjem. To je taktika za maksimiranje uporabnosti agenta.

- **Večopravilno učenje (ang. *Multitask Learning*):** To je učenje, pri čemer je preprost cilj pomagati ostalim učencem do boljših rezultatov [4].

- **Ansambelsko učenje (ang. *Ensemble Learning*):** Je učenje, kjer so številni individualni učenci kombinirani v nov algoritem [4].

- **Nevronske mreže (ang. *Neural Network Learning*):** Je biološko navdihnjen algoritem, ki posnema nevrone v možganih.

- **Učenje na podlagi primera (ang. *Instance-Based Learning*):** Učitelj nauči poseben tip vzorca in ob novem podatku poskuša izračunati prilagajanje vzorcu.

V nadaljevanju so opisane nekatere metode strojnega učenja, ki jih zasledimo v literaturi za napovedovanje finančnih časovnih vrst.

2.1 Naivni bajes (ang. *Naive Bayes*)

Algoritem naivni bajes spada v skupino nadzorovanega učenja in je klasifikacijski algoritem. Temelji na verjetnostnem pristopu Bayesovega teorema, pri čemer predpostavlja, da so vsi atributi neodvisni pri danem razredu [14].

V članku [13] sta avtorja zgradila model za avtomatsko kupovanje in prodajo vrednostnih papirjev na podlagi napovedovanja z naivnim bajesom in naključnimi gozdovi. Za vhode v model sta uporabila tehnične indikatorje, kot so relativno močnostni indeks (ang. *Relative Strength Index* – RSI), premično povprečje konvergence divergence (ang. *Moving Average Convergence Divergence* – MACD), stohastični oscilator (ang. *Stochastic Oscillator*), Williams %R indikator, denarni tokovni indeks (ang. *Money Flow Index* – MFI), Bollingerjev trak (ang. *Bollinger Band*), blagovni kanalni indeks (ang. *Commodity Channel Index* – CCI), stopnja spreminjanja cen (ang. *Price Rate of Change* – PRoC) in indikator ravnovesnim obsegom (ang. *On-balance Volume* – OBV). Za podatke sta uporabila desetletno zgodovino cen vrednostnih papirjev indijske borze. Za nekaj izbranih vrednostnih papirjev sta izračunala omenjene indikatorje. Zgrajeni model je upošteval še uporabniške zahteve, kot so investicijska vsota, časovni interval investicije, minimalni dobiček in maksimalna izguba. Za primerjanje algoritmov sta uporabila mere klasifikacijsko točnost, preklic in preciznost. Iz rezultatov je razvidno, da je metoda naključni gozdovi boljša od naivnega bajesa.

2.2 K najbližjih sosedov

Metoda K najbližjih sosedov (ang. *K Nearest Neighbours*) ali na kratko k-NN je klasifikacijska metoda, ki temelji na najbližjih učnih primerkov v prostoru značilk [5]. Gre za tako imenovano leno učenje (ang. *Lazy Learning*), kjer je glavno procesiranje odloženo na klasificiranje novega primera.

Avtorji so v članku [5] za šest vrednostnih papirjev iz Jordanske borze uporabili algoritem k-NN in nelinearen regresor. Za vhodne podatke za algoritem so uporabili ceno delnice ob zaprtju borze (ang. *Closing Price*), dnevno najnižjo vrednost (ang. *Low Price*) in dnevno najvišjo vrednost (ang. *High Price*) za tisoč delovnih dni v preteklost. Pri tem so določili število sosedov na 5. Za primerjavo rezultatov so uporabili mere: srednje kvadratna napaka (ang. *Root Mean Square Error* – RMSE), vsota kvadratov (ang. *Sum of*

Squares – ESS) in povprečna pričakovana napaka (ang. *Average Estimated Error* – AEE). Avtorji navajajo, da je RMSE v povprečju 0,263, ESS 0,0378 in AEE 5,434E-9.

2.3 Odločitvena drevesa

Odločitvena drevesa (ang. *Decision Trees*) so metode, ki rekurzivno razdelijo nabor podatkov, pri čemer uporabljajo požrešen algoritem iskanja v globino. Ta postopek se ponavlja, dokler vsi podatki ne pripadajo določenemu parcialnemu razredu [17].

V članku [17] sta avtorja zgradila hibridni model, sestavljen iz metode odločitvena drevesa in metode nevronska mreža. S to kombinacijo sta dosegla 77-odstotno natančnost napovedovanja na Tajvanski borzi. Uporabila sta osnovne, tehnološke in makroekonomske indikatorje, skupaj 53 atributov. Za primerjanje algoritmov sta uporabila konfuzijsko matriko. Ugotovila sta, da je metoda odločitvena drevesa nekoliko boljša kot metoda nevronska mreža in da kombinirana metoda doseže večjo natančnost.

2.4 Metoda podpornih vektorjev

Metoda podpornih vektorjev (ang. *Support Vector Machine*) temelji na statistični teoriji in spada med algoritme nadzorovanega učenja. Glavna ideja algoritma je, da zgradi najbolj optimalno hiperravnino, ki ločuje primere med posameznimi razredi, v implicitno spremenjenem prostoru atributov s pomočjo izbrane jedrne funkcije. Pri tem algoritem maksimira razdaljo med razredi [11].

Avtor dela [11] je uporabil 34 tehnoloških vrednostnih papirjev borze S&P500 za napovedovanje z metodo podpornih vektorjev. Za podatke je uporabil cene vrednostnih papirjev med letoma 2007 in 2014, pri tem je uporabil 70 % zapisov za učenje in 30 % za testiranje. Pri napovedovanju se je odločal med kratkoročno napovedjo (napoved za en dan naprej) in dolgoročno

napovedjo (do 270 dni vnaprej). Ugotovil je, da je metoda najučinkovitejša, če napoveduje za 5, 10 in 20 dni vnaprej.

2.5 Nevronska mreža

Nevronska mreža (ang. *Neural Network*) je nastala na podlagi posnemanja naravnih nevronov v možganih. Glavni element je nevron ali vozlišče. Več nevronov, povezanih med sabo z utežmi, sestavlja plast (ang. *layer*). V nevronu se nahaja odzivna funkcija, ki poda vrednost naslednjemu nevronu ali pa izhodu [10].

Avtorji v članku [10] so uporabili nevronsko mrežo na bangladeški borzi. Za vhodne atribute so izbrali pet indikatorjev: splošni indeks (ang. *General Index* – GI), čista vrednost sredstev (ang. *Net Asset Value* – NAV), razmerje P/E (ang. *P/E Ratio*), zaslužek na delnico (ang. *Earnings per Share* – EPS) in obseg vseh delnic (ang. *Share Volume*). Nevronska mreža je bila sestavljena iz petih vhodnih nevronov, nato je sledila skrita plast petih nevronov in na koncu izhodni nevron. V nevronih so uporabili sigmoidno odzivno funkcijo. Ugotovili so, da večje število vhodov v nevronsko mrežo proizvede večjo napako.

2.6 Naključni gozdovi

Naključni gozdovi (ang. *Random Forest*) je zelo popularen in učinkovit algoritem, ki temelji na združevanju predikcij večjega števila modelov. Naključni gozdovi so sestavljeni iz množice odločitvenih dreves, ki jih zgradimo z naključno izbranimi učnimi podmnožicami [9].

V članku [9] za vhodne atribute v algoritem naključni gozdovi so avtorji uporabili enake tehnološke indikatorje kot v članku [13], ki je opisan v razdelku 2.1. Preden so podatke uporabili, so jih filtrirali z eksponentnim glajenjem. Algoritem so uporabili na samo treh vrednostnih papirjih: Apple, Amazon in Microsoft. Tako sestavljen model je imel točnost med 85 % in 95

%.

2.7 Genetski algoritem

Genetski algoritem (ang. *Genetic Algorithm*) je nastal podobno kot nevronska mreža na podlagi posnemanja bioloških procesov, kot so selekcija, križanje in mutacija. Algoritem temelji na osebkih, ki imajo genski zapis, ki predstavlja lastnosti določenega osebka. Za vsak osebek obstaja t. i. funkcija fitness, ki meri, kako dober je osebek glede na druge osebke [2].

Avtorja v članku [2] uporabita šest atributov za napovedovanje osmih vrednostnih papirjev, s katerim dosežeta 70-odsotno natančnost. Za podatke sta vzela vrednosti vrednostnih papirjev za pet let. Za funkcijo fitnes sta uporabila vsoto pravilnih napovedi.

Poglavje 3

Opis novega kombiniranega algoritma za napovedovanje vzporednih časovnih vrst

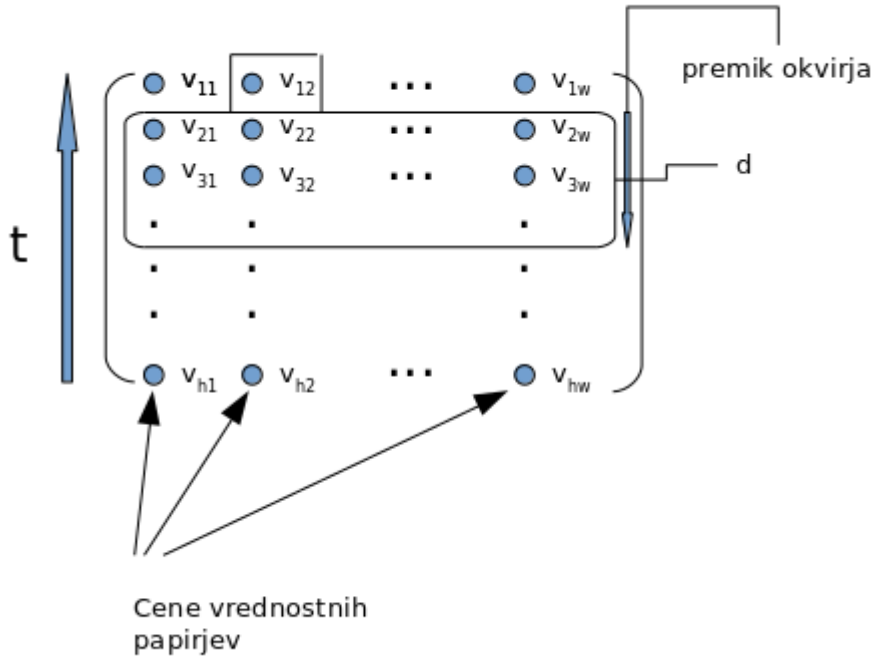
Pri pregledu literature se je pokazalo, da so v veliki meri osnovni algoritmi strojnega učenja narejeni samo za eno časovno vrsto. V poglavju 4 je opisano, kako prilagoditi algoritme, da uporabijo več časovnih vrst za napovedovanje.

Prvotna ideja, kako uporabiti več vzporednih časovnih vrst, je bila linearna regresija. Predpostavka je temeljila na tem, da so vrednosti vrednostnih papirjev odvisne med seboj. To pomeni, da obstajajo neke skrite korelacije med njimi. Torej, če en vrednostni papir začne rasti, lahko drugi vrednostni papir začne tudi rasti ali pa padati.

Primer take relacije je lahko podjetje A, ki proizvaja papir, in podjetje B, ki predeluje lesno maso. Če vrednosti vrednostnih papirjev podjetja A začnejo rasti, je velika verjetnost, da bodo vrednostni papirji podjetja B tudi začeli rasti.

Zato je izpeljana predpostavka, da so vrednosti vrednostnih papirjev odvisne od vrednostih drugih vrednostnih papirjev. Cena je torej lahko utežena vsota vrednostnih papirjev.

Iz Slike 3.1 lahko izpeljemo enačbo (3.1), ki ima $w \cdot d$ neznanih spremen-



Slika 3.1: Matrika cen vrednostnih papirjev in okvir.

ljivk. Da bi poiskali rešitev za tak sistem, potrebujemo več takih enačb. To pa dobimo tako, da okvir premaknemo za eno vrstico po matriki bolj dol, in dobimo enačbo (3.2).

$$a_{11}v_{21} + a_{12}v_{22} + ... + a_{1w}v_{2w} + a_{21}v_{31} + a_{22}v_{32} + ... + a_{2w}v_{3w} + ... = v_{12} \tag{3.1}$$

$$a_{11}v_{31} + a_{12}v_{32} + ... + a_{1w}v_{3w} + a_{21}v_{41} + a_{22}v_{42} + ... + a_{2w}v_{4w} + ... = v_{22} \tag{3.2}$$

...

Ta postopek ponavljamo, dokler imamo podatke. Poudariti moramo, da bo sistem enačb napovedoval časovno vrsto številka 2.

V bolj kompaktni obliki lahko zapišemo enačbe kot:

$$v_{i2} = \sum_{k=1}^{w} \sum_{h=1}^{d} a_{k,h} v_{k+i+o,h} \tag{3.3}$$

V enačbi (3.3) predstavlja w število časovnih vrst, spremenljivka d predstavlja globino okvirja in spremenljivka o število dni, kolikor jih napovedujemo vnaprej. Neznane spremenljivke so uteži $a_{k,h}$, ki jih lahko izračunamo z metodo najmanjših kvadratov [3].

3.1 Metoda najmanjših kvadratov

Metoda najmanjših kvadratov je metoda, ki rešuje sisteme linearnih enačb. Enačbe so oblike:

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2 + ... + a_m z_m + e \tag{3.4}$$

Enačbo (3.4) lahko zapišemo v matrični obliki:

$$\{Y\} = [Z]\{A\} + \{E\} \tag{3.5}$$

V enačbi (3.5) nastopa matrika $[Z]$, kjer so $z_{i,j}$ vrednosti vrednostnih papirjev. Tako dobimo matriko $[Z]$:

$$[Z] = \begin{bmatrix} z_{01} & z_{11} & \dots & z_{m1} \\ z_{02} & z_{12} & \dots & z_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{0n} & x_{1n} & \dots & z_{mn} \end{bmatrix} \tag{3.6}$$

Rešitev dobljenega sistema dobimo z naslednjo izpeljavo:

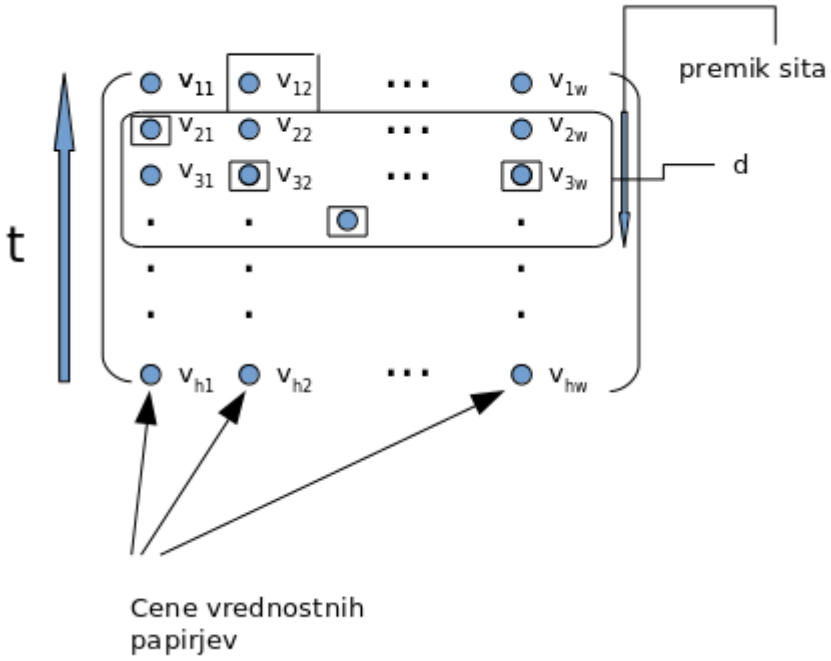
$$[[Z]^\top [Z]]\{A\} = \{[Z]^\top \{Y\}\} \tag{3.7}$$

$$A = [[Z]^\top [Z]]^{-1}\{[Z]^\top \{Y\}\} \tag{3.8}$$

Z enačbo (3.8) lahko izračunamo uteži A.

Tak sistem enačb je zelo velik in za uporabo skoraj neuporaben zaradi porabe prostora v pomnilniku in časovne potratnosti. Tako je nastala ideja, da ga nekoliko zmanjšamo na način, da ima algoritem manj neznank. Zato smo namesto okvirja uporabili sito, gl. Sliko 3.2. Označeni kvadrati označujejo, katere spremenljivke bomo uporabili v sistemu linearnih enačb. Vendar so nastale naslednje težave, in sicer kako definirati sito, koliko spremenljivk uporabiti in na kateri poziciji v situ naj nastopajo.

Rešitev problema se je našla v genetskem algoritmu, ki je opisan v razdelku 2.7. Genetski algoritem je bil modificiran tako, da išče najbolj optimalno sito. Funkcija fitnes je zgrajena tako, da opazuje naklon med dvema zaporednima napovedima; če je bil naklon navzgor, je pomenilo, da je priporočljivo kupiti vrednostni papir, in obratno, če je bil naklon negativen. Na začetku je algoritem imel določeno vsoto denarja, ki jo je uporabil za nakup vrednostnih papirjev za sto dni v preteklosti. Geni so predstavljali konkretno sito, ki je na začetku bilo naključno generirano. Nato je za vsak gen/sito algoritem izračunal funkcijo fitnes in gen/sito razvrstil po najbolj ugodni funkciji fitnes. Ko je algoritem izračunal vse funkcije fitnes, je nato prvo tretjino genov/sit prestavil v drugo iteracijo. Drugo tretjino genov/sit



Slika 3.2: Matrika cen vrednostnih papirjev in sito.

je nato križal med seboj, in na koncu še nad tretjo tretjino izvršil mutacijo. Nato je šel na naslednjo iteracijo in ponovil postopek.

Dobljena rešitev je predpostavila, da so relacije med vrstami linearne; da se je izboljšala napoved, je bilo potrebno poiskati rešitve z nelinearnimi sistemi.

3.2 Nelinearna regresija

Iz numerične matematike lahko za reševanje nelinearnih sistemov uporabimo nelinearno regresijo. Ta metoda rešuje naslednje sisteme enačb:

$$y_i = f(x_i; a_0, a_1, ..., a_m) + e_i \tag{3.9}$$

Pri tem je funkcija f nelinearna funkcija neodvisne spremenljivke x_i in odvisnih parametrov $a_0, a_1, ..., a_m$. Enačbo (3.9) lahko zapišemo krajše:

$$y_i = f(x_i) + e_i \tag{3.10}$$

Dobljeni sistem lahko s Taylorjevo vrsto razvijemo okoli točke $\boldsymbol{x}$ in dobimo:

$$y(x_i)_{j+1} = f(x_i)_j + \frac{\partial f(x_i)_j}{\partial a_0}\Delta a_0 + \frac{\partial f(x_i)_j}{\partial a_1}\Delta a_1 + ... + \frac{\partial f(x_i)_j}{\partial a_m}\Delta a_m \tag{3.11}$$

V enačbi (3.11) je $f(x_i)_j$ začetna vrednost funkcije s parametri $a_0, a_1, ..., a_m$ in $y(x_i)_{j+1}$ je predvidena nova vrednost funkcije. Enačbo (3.10) lahko vstavimo v enačbo (3.11) in dobimo:

$$y_i - f(x_i)_j = \frac{\partial f(x_i)_j}{\partial a_0}\Delta a_0 + \frac{\partial f(x_i)_j}{\partial a_1}\Delta a_1 + ... + \frac{\partial f(x_i)_j}{\partial a_m}\Delta a_m + e_i \tag{3.12}$$

V kompaktnejši matrični obliki dobimo:

$$\{D\} = [Z_j]\{\Delta A\} + \{E\}, \tag{3.13}$$

pri čemer je matrika $[Z_j]$ Jakobijeva matrika in ima naslednjo obliko:

$$[Z_j] = \begin{bmatrix} \frac{\partial f_1}{\partial a_0} & \frac{\partial f_1}{\partial a_1} & \cdots & \frac{\partial f_1}{\partial a_m} \\ \frac{\partial f_2}{\partial a_0} & \frac{\partial f_2}{\partial a_1} & \cdots & \frac{\partial f_2}{\partial a_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial a_0} & \frac{\partial f_n}{\partial a_1} & \cdots & \frac{\partial f_n}{\partial a_m} \end{bmatrix} \tag{3.14}$$

Pri tem je n število podatkov, m število parametrov in $\frac{\partial f_i}{\partial a_k}$ je parcialni diferencial funkcije v k-tem parametru. Vektor $\{D\}$ vsebuje diference med pravo vrednostjo in izračunano vrednostjo funkcije:

$$\{D\} = \begin{Bmatrix} y_1 - f(x_1) \\ y_2 - f(x_2) \\ . \\ . \\ . \\ y_n - f(x_n) \end{Bmatrix}$$

Vektor $\{\Delta A\}$ je sledeč:

$$\{\Delta A\} = \begin{Bmatrix} \Delta a_0 \\ \Delta a_1 \\ . \\ . \\ . \\ \Delta a_m \end{Bmatrix}$$

Tak sistem lahko rešimo z naslednjo enačbo:

$$[[Z_j]^\top [Z_j]]\{\Delta A\} = \{[Z_j]^\top \{D\}\} \tag{3.15}$$

Sedaj lahko nastavimo vrednosti atributov $\{A\}$:

$$\begin{aligned} a_{0,j+1} &= a_{0,j} + \Delta a_0 \\ a_{1,j+1} &= a_{1,j} + \Delta a_1 \\ &... \end{aligned} \tag{3.16}$$

Pri dobljenem sistemu je torej prvotno potrebno nastaviti vrednosti parametrov $\boldsymbol{A}$. V našem primeru smo nastavili začetne vrednosti parametrov na 1. Omenjena metoda je iterativna, kar pomeni, da algoritem večkrat poženemo, in po vsaki iteraciji se spremenijo vrednosti parametrov $\boldsymbol{A}$.

Algoritem lahko ustavimo na več načinov; prvi je, da omejimo število iteracij, drugi pa, da pogledamo, kako se vrednosti parametrov spreminjajo:

$$|\epsilon_a|_k = \left| \frac{a_{k,j+1} - a_{k,j}}{a_{k,j+1}} \right| 100\% \tag{3.17}$$

Če določimo neko vrednost, do katere se bodo vrednosti parametrov $\boldsymbol{A}$ spreminjali, lahko ustavimo izvajanje algoritma, ko doseže to vrednost.

3.3 Uporaba predlaganega kombiniranega algoritma

Sedaj ko imamo razložene osnovne gradnike novo predlaganega algoritma, je potrebno razložiti, kako so ti gradniki povezani med seboj. Začne se z

matriko (3.6), kjer so z_{ij} vrednosti vrednostnih papirjev, in večji kot je j, starejši so podatki; i predstavlja, za kateri vrednostni papir gre. Prav tako tudi predpostavimo, da so z_{ij} večji od nič, torej nenegativna števila. Število m prestavlja število vseh vključenih vrednostnih papirjev, n pa pove, koliko podatkov imamo v zgodovini. V našem primeru se število podatkov giblje okoli 4000 delovnih dni v preteklost (to je približno 16 let) za 1500 vrednostnih papirjev.

Podatke smo dobili na spletni strani `www.quandl.com`. Na tem mestu je treba poudariti, kaj storiti z neveljavnimi oz. nepopolnimi podatki, ki jih internetna baza na žalost vsebuje. Najbolj preprosta rešitev je bila, da se uporabijo samo tisti vrednostni papirji, ki imajo vse veljavne podatke.

Nato se je izbral nek vrednostni papir s številko, ki se je skupaj z matriko $[Z]$ poslal v algoritem. Naslednji korak algoritma je, da zgradi genome. Genomi so v tem primeru sito, ki jih zgradi naključno; prav tako zgradi naključno pripadajoče nelinearne funkcije. Funkcije so naslednje oblike:

$$f(x_t) = a_1 x_{t+o} + a_2 \tag{3.18}$$

Funkcija (3.18) je linearna funkcija, ki ima za parameter eno časovno vrsto (vrednostni papirji) x_t, o predstavlja odmik v situ te časovne vrste. Atributa a_1 in a_2 sta neodvisna parametra, ki se pri nelinearni regresiji izračunata.

Naslednja funkcija je:

$$f(x_t) = \sin(a_3 x_{t+o} + a_4) + a_5 \tag{3.19}$$

Funkcija (3.19) je nelinearna in ima neodvisne parametre a_3,a_4 in a_5, ki se prav tako kot (3.18) izračuna pri nelinearni regresiji.

Naslednja funkcija je:

$$f(x_t, x_{t1}) = \sin(a_6 x_{t+o} + a_7) \sin(a_8 x_{t1+o1} + a_9) + a_{10} \tag{3.20}$$

Funkcija (3.20) je nelinearna funkcija, ki je odvisna od dveh različnih vrednostnih papirjev, in ima 5 neodvisnih parametrov. Na enak način smo uporabili tudi preostale funkcije (sinh, cosh itd.).

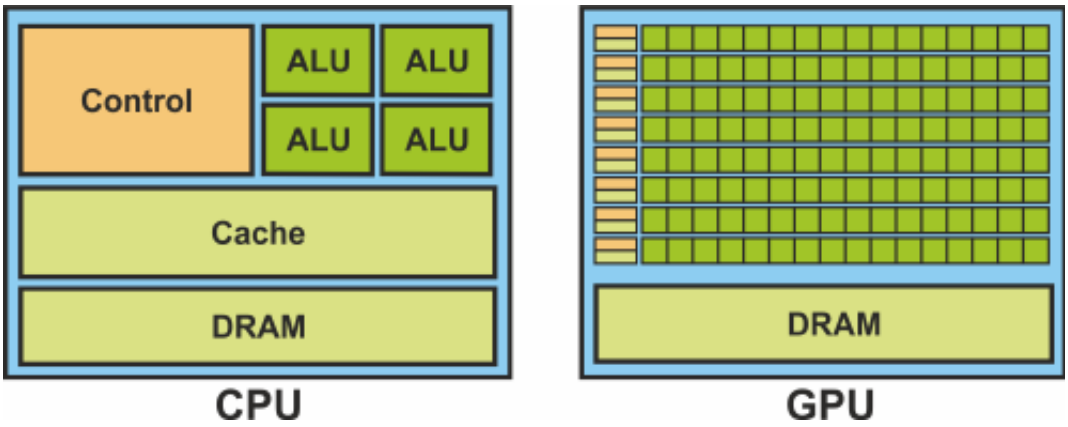
Algoritem iz predlaganih funkcij sestavi eno "dolgo" funkcijo, naključno namreč izbira med tremi omenjenimi funkcijami in jih sešteje, istočasno pa še naključno določi, za katere časovne vrste gre in njihove odmike v situ. Tu smo omejili dolžino funkcije na sedem (sedem naključno izbranih omenjenih funkcij). Tako dobimo en set nelinearnih funkcij, nato pa se sito premakne za eno vrstico nižje, in dobimo drug set nelinearnih funkcij. To počnemo, dokler imamo podatke. Medtem ko premikamo sito navzdol, moramo sestaviti še Jakobijevo matriko (3.14) iz "dolge" funkcije. To pa storimo tako, da poiščemo odvod funkcije po vseh a-jih in izračunamo po pripadajočih vrednostih časovne vrste. Tu omenimo, da se da algoritem zelo pospešiti z uporabo vmesnika OpenGL.

Ko izračunamo Jakobijevo matriko, moramo nato nastaviti začetne vrednosti za vse a-je. V našem primeru smo jih nastavili na začetno vrednost 1. Nato lahko z enačbo (3.15) izračunamo prvih nekaj približkov a-ja. Tu smo omejili število iteracij na deset ali če je vrednost funkcije (3.17) padla pod 5 %.

Ko imamo izračunane a-je, lahko dejansko iz modela izračunamo napoved, tako da v "dolgi" funkciji vnesemo a-je in pripadajoče vrednosti časovnih vrst. Na ta način smo, odvisno od spremenljivke o v enačbi (3.3), napovedali vrednosti za o dni naprej. Dobljene napovedi niso natančne in odstopajo od realnih podatkov. Da lahko uporabimo omenjene napovedi, uporabimo naklone grafov napovedi. To pomeni, da moramo izračunati napoved za dva dni v prihodnost in iz naklona razbrati, ali bo časovna vrsta rasla ali padala.

3.4 Pohitritev predlaganega algoritma z vmesnikom OpenGl

V prejšnjem razdelku je bilo omenjeno, da lahko algoritem pohitrimo z uporabo programskega vmesnika OpenGL. Da lahko razložimo, na kakšen način to izvedemo, je potrebno razložiti, kako deluje OpenGL.



Slika 3.3: Primerjava CPU in GPU arhitekture.

3.4.1 Primerjava GPU in CPU arhitekture

Na Sliki 3.3 je prikazana razlika med CPU in GPU arhitekturo. CPU arhitektura je sestavljena iz nekaj velikih jeder aritmetičnih enot (ALU), ki so namenjena za obdelavo splošnih operacij z velikim začasnim pomnilnikom in enim velikim krmilnim modulom. CPU je optimiziran za serijske operacije, medtem ko ima GPU veliko malih ALU-jev, majhnih kontrolnih modulov in majhen predpolnilnik, zato je optimiziran za vzporedne operacije.

3.4.2 Predstavitev vmesnika OpenGL

OpenGL je aplikacijski programski vmesnik (ang. *Application Programming Interface* – API), ki lahko prikazuje grafične primitivne elemente, transformacije matrik, sledenje svetlobnim žarkom itd. OpenGL je nastal leta 1992 ter se od takrat večkrat spremenil in posodobil. Trenutna verzija vmesnika OpenGL je 4.5.

GLSL ali ang. *OpenGL Shading Language* je programski jezik, ki omogoča pisanje lastnih algoritmov za senčenje:

- *vertex shaders*,
- *fragment shaders*,

- *geometry shaders* in
- *teselation shaders*.

GLSL je podoben programskemu jeziku C z zelo omejenim naborom tipov in funkcij, a omogoča hitrejše izrisovanje, saj rešuje problem ozkega grla pri komunikaciji med CPU in GPU. V našem primeru se bomo podrobneje posvetili algoritmu *vertex shader*.

V vertex shader ponavadi podamo 3D- ali 4D-vektorje (vertexe) in na njih lahko izvajamo različne operacije. Predlagan shader je prilagojen, da dobi samo zaporedno številko, ki predstavlja, za kateri element v SSBO (ang. *Shader Storage Buffer Object*) mora računati. Seveda se računanje dogaja paralelno, zato je treba paziti, da v SSBO ne pišemo hkrati v isto lokacijo iz drugih paralelnih procesov. Hkrati obstajajo (uniformni) atributi v shaderju, ki so enaki za vse paralelne procese. S temi atributi je možno opisati, kaj mora vsak program delati. Ti atributi lahko kažejo tudi na funkcije in so podobni kazalcem na funkcije v programskem jeziku C.

3.4.3 Prilagoditve vertex shaderja

Pri pregledu delovanja vmesnika OpenGl je logična rešitev za pohitritev predlaganega algoritma računanje Jakobijeve matrike s sitom za vsako vrstico paralelno. Elementi Jakobijeve matrike so neodvisni drug od drugega in se jih lahko preprosto izračuna.

Glavni problem tega pristopa je bil, da je OpenGL GS zelo omejen, in posledično je bilo treba implementirati preproste linearne algebraične operacije, kot so množenje in seštevanje med vektorji in matrikami.

V zaključku poglavja je treba omeniti, da je predlagani algoritem še vedno časovno dokaj potraten. V naslednjem poglavju je opisano, kako primerjati različne algoritme med seboj, čemur sledi opis prilagoditev obstoječih algoritmov za napovedovanje časovnih vrst.

Poglavje 4

Izdelava metodologije za primerjanje algoritmov in njihove prilagoditve

V poglavjih 2 in 3 so opisani algoritmi, ki so primerni za napovedovanje časovnih vrst. Za uporabo vzporednih časovnih vrst je potrebno te algoritme spremeniti oziroma pripraviti podatke v ustrezno obliko. Vsi omenjeni algoritmi spadajo v skupino nadzorovanega učenja, kar pomeni, da imajo vsi podatki definiran razred, v katerega spadajo.

Kako določiti razrede in kako definirati atribute?

Osnovna ideja za definiranje atributov je, da poleg izbrane napovedne časovne vrste izberemo še nekaj vzporednih časovnih vrst in jih poravnamo po času. Razred določimo tako, da iz napovedne časovne vrste vzamemo prvi (najnovejši) element in ga definiramo kot razred, tako da pogledamo, če raste ali pada. Ostalim izbranim časovnim vrstam pa odrežemo prve elemente. Tako dobimo par razred in množico atributov. Atributov je v tem primeru več, in zato je potrebno razširiti osnovne algoritme, da uporabijo takšno število atributov.

Za učenje je potrebno več podatkov. Dobimo jih tako, da se po podatkih premikamo navzdol in vsakič iz napovedane časovne vrste vzamemo prvi

element, ki je razred, ostalim vzporednim časovnim vrstam pa odrežemo prve podatke. To je podobno kot pri premikanju okvirja (gl. Sliko 3.1).

Podatki, ki jih imamo, so nenegativna števila, in žal jih ne moremo uporabiti direktno v omenjenih osnovnih algoritmih. Prednost novega predlaganega algoritma je tudi to, da deluje na nespremenjenih podatkih, dokler je matrika podatkov polna. Da bomo lahko primerjali algoritme med seboj, bo potrebno podatke transformirati.

Prva transformacija je normiranje časovnih vrst, ki preslika vrednosti časovnih vrst med 0 in 1. To stori tako, da poišče minimalno in maksimalno vrednost časovne vrste in preslika vse vrednosti po naslednji formuli (4.1):

$$v_{nova} = \frac{v_{stara} - min}{max - min} \tag{4.1}$$

S to preslikavo smo ohranili predvsem obliko časovne vrste.

Druga transformacija preslika časovno vrsto v ničle in enke; ničla pomeni, da časovna vrsta pada, enka pa da časovna vrsta narašča. Ta transformacija je primerna za vse omenjene algoritme, to pomeni, da algoritmi konvergirajo, seveda pa se pri tem postopku izgubljajo informacije.

Tretja transformacija je diskretizacija, ki določi meje med intervali diskretnih vrednosti. Primer intervala je [-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0]. Če razlika pade med -0.8 in -0.6, je to 2. interval, če je razlika 0.1, je to 6. interval itn. Ta transformacija zmanjša prostor zaloge vrednosti atributov.

4.1 Metodologija primerjanja algoritmov

Algoritme strojnega učenja in predlagan algoritem smo testirali na istih podatkih z istimi transformacijami. Tako dobimo primerljive rezultate, ki smo jih v poglavju 5 analizirali.

Še preden se lotimo analize, je potrebno podatke razdeliti na učne in testne množice. Za učne množice smo izbrali okoli 1500 časovnih vrst, ki imajo 4000 delovnih dni zgodovine. Matrika podatkov je velika 1500x4000

(6.000.000 podatkov). Učni množici smo odstranili prvih 30 zapisov, ki smo si jih shranili za primerjavo. Nato smo pognali izbrani algoritem in napovedali, ali bo časovna vrsta naraščala ali ne, in s shranjenimi zapisi smo lahko primerjali, če je algoritem pravilno napovedal. Ko smo dobili primerjavo, smo matriki podatkov dodali najstarejšo shranjeno vrstico (da smo ohranili zaporedje). Nato pa smo pognali novo iteracijo na novih podatkih (shranjena vrstica + stari podatki). Tako dobljeni podatki za 30 delovnih dni so bili primerni za analizo.

4.2 Mere za ocenjevanje učenja

Za primerjavo algoritmov so v literaturi [6] opisane najpogostejše mere. V magistrski nalogi so uporabljene naslednje mere:

4.2.1 Klasifikacijska točnost (ang. *Classification Accuracy*)

Klasifikacijsko točnost ocenjujemo na neodvisni testni množici, to je na 30 delovnih dni, kot je omenjeno v začetnem odstavku tega poglavja.

4.2.2 Senzitivnost in specifičnost

Ker imamo dvorazredni problem, saj nas zanima, ali bo vrednost vrednostnega papirja rasla ali padala, sta primerni dve meri: senzitivnost (ang. *sensitivity*) in specifičnost (ang. *specificity*). Ti meri sta izpeljani iz štirih osnovnih količin, razvidnih iz tabele [6].

pravi razred	napovedani razred		vsota
	P	N	
P	TP	FN	POS=TP+FN
N	FP	TN	NEG=FP+TN
vsota	PP=TP+FP	PN=FN+TN	n = TP+FP+FN+TN

Tabela 4.1: Števila napačnih klasifikacij za dvorazredni problem.

Pri tem so pomeni oznak naslednji:

P – Pozitivni razred.

N – Negativni razred.

n – Število vseh primerov.

TP (ang. *True Positives*) – Število pravilno klasificiranih pozitivnih primerov.

FP (ang. *False Positives*) – Število lažnih pozitivnih primerov, torej število negativnih primerov, ki jih je klasifikator zmotno uvrstil med pozitivne.

TN (ang. *True Negatives*) – Število pravilno klasificiranih negativnih primerov.

FN (ang. *False Negatives*) – Število lažnih negativnih primerov, torej število pozitivnih primerov, ki jih je klasifikator zmotno uvrstil med negativne.

POS – Število pozitivnih primerov.

NEG – Število negativnih primerov.

PP (ang. *Predicted Positives*) – Število primerov, klasificiranih kot pozitivnih.

PN (ang. *Predicted Negatives*) – Število primerov, klasificiranih kot negativnih.

Senzitivnost ali občutljivost je verjetnost, da klasifikator zazna pozitivni primer, torej ocenjuje odstotek pravilno klasificiranih pozitivnih primerov:

$$Senz = \frac{TP}{TP + FN} = \frac{TP}{POS} \tag{4.2}$$

Specifičnost ocenjuje odstotek pravilno klasificiranih negativnih primerov:

$$Spec = \frac{TN}{TN + FP} = \frac{TN}{NEG} \tag{4.3}$$

Klasifikacijska točnost je:

$$T = \frac{TP + TN}{TN + FP + FN + TN} = \frac{TP + TN}{n} \tag{4.4}$$

4.2.3 Mera "dobička"

Mera "dobička"je mera, ki kaže, koliko je klasifikator pridobil dobička glede na začetni vložek. V našem primeru smo določili začetni vložek na 10.000. Dobiček smo računali na podlagi naslednje strategije trgovanja: če je klasifikator napovedal, da bo vrednostni papir rasel, smo za vso vsoto, ki nam je bila na razpolago, kupili delnice, in obratno, če je napovedal, da bo padel, smo prodali vse delnice.

Poglavje 5

Primerjava in analiza algoritmov za napovedovanje vzporednih časovnih vrst

Primerjava in analiza algoritmov je potekala v dveh fazah. Prva faza je vsebovala analizo in primerjavo na umetno generiranih podatkih, druga faza je vsebovala realne podatke iz vrednostnih papirjev. Izračun je potekal na procesorju i7-6700HQ CPU @ 2.60GHz s 16GB rama in z grafičnim procesorjem GeForce GTX 960M s 4GB rama.

Za analizo obstoječih algoritmov se je uporabilo obstoječe implementacije v R-u, ki so imele sledeče parametre:

- Naivni bajes ni potreboval nobenih dodatnih nastavitev parametrov.
- Metoda najbližjih sosedov je za k imela nastavljeno vrednost 20, kar pomeni, da je iskala večinski razred 20-ih najbolj podobnih vzorcev.
- Metoda odločitvena drevesa ni potrebovala dodatnih nastavitev parametrov.
- Metoda podpornih vektorjev je za nastavitev klasifikacijske napovedi imela nastavljeno vrednost parametra "C-classification".

- Metoda naključni gozdovi ni potrebovala dodatnih nastavitev parametrov.

- Pri metodi nevronska mreža se je uporabilo tri skrite plasti s po 10 nevroni na plast in z enim izhodnim nevronom. Pri tej metodi je bilo potrebno še nastaviti maksimalno število iteracij na 10e8 in odzivno funkcijo v nevronu na tanh.

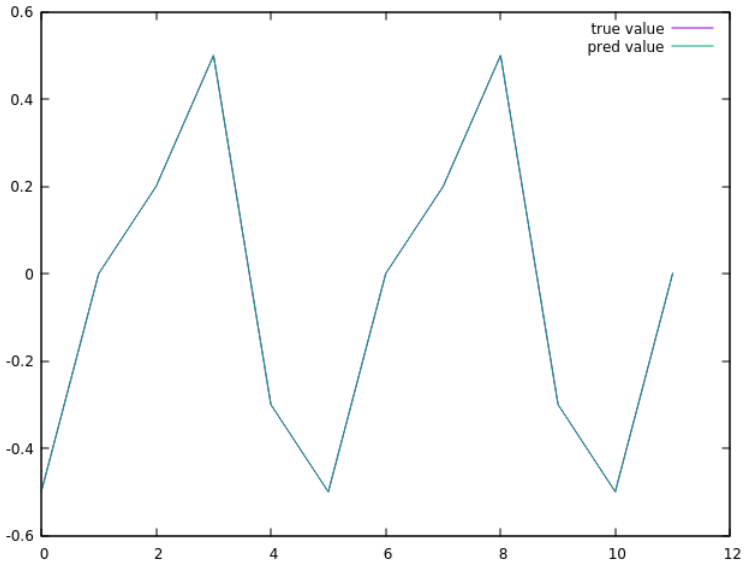
Pri kombiniranem algoritmu se je nastavilo globino sita na 100 in število generacij na 8.

5.1 Umetno generirani podatki

Umetno generirane podatke se je v prvi fazi pridobilo na več načinov. Prva podatkovna zbirka je bila sestavljena iz petih časovnih vrst.

Prva časovna vrsta, ki je bila izbrana za napovedovanje, je bila alternirajoča časovna vrsta, v kateri sta se izmenjavali vrednosti 0,01 in 1,0. Če bi se izbralo vrednost nič, bi bila mera dobička vedno nič. To bi pomenilo, da kupujemo delnice, ki imajo vrednost nič. Vzporedne štiri časovne vrste so bile generirane z naključnimi vrednostmi med 0,01 in 1,0. Vse časovne vrste so bile dolge 4.000 zapisov. Namen tega testa je, da se preizkusijo algoritmi na preprostem problemu, kjer tudi ni odvisnosti med vzporednimi časovnimi vrstami. Tako dobljene časovne vrste se je razdelilo na učno in testno množico, tako da se je vzelo prvih d zapisov za testne podatke. V tem primeru so vsi algoritmi, vključno z kombiniranim algoritmom, napovedali s klasifikacijsko točnostjo 100 %, in mera dobička je bila $2/(0.01)^d$. Pri tem se je uporabil vložek vrednosti dveh denarnih enot. Podobno se je algoritem obnašal ob spremembi alternirajoče funkcije, kot je prikazano na grafu 5.1. Tudi v tem preprostem problemu ni odvisnosti med vzporednimi časovnimi vrstami. V tem primeru se je uporabila mera dobička tako, da se je vsem vrednostim časovne vrste prištelo 0,6, in zato ni bilo negativnih števil.

Druga podatkovna zbirka je vsebovala pet časovnih vrst, sestavljenih iz

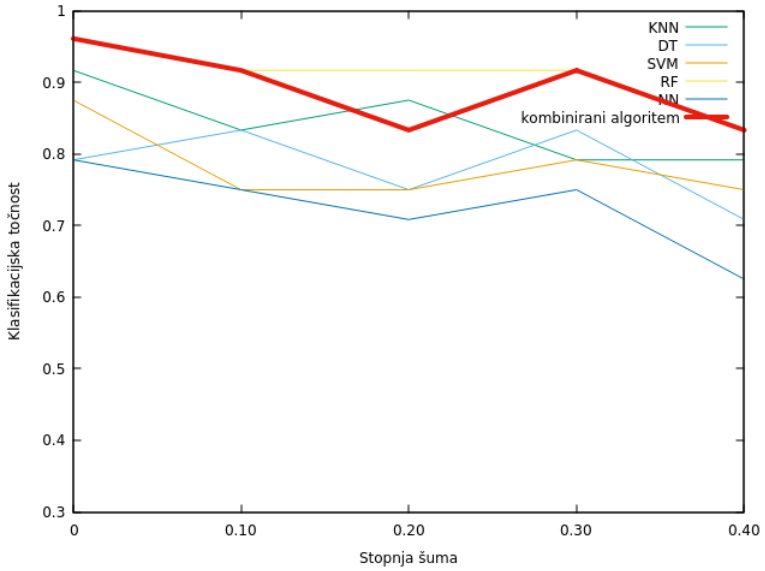


Slika 5.1: Graf spremenjene alternirajoče funkcije in predikcije za $d = 12$.

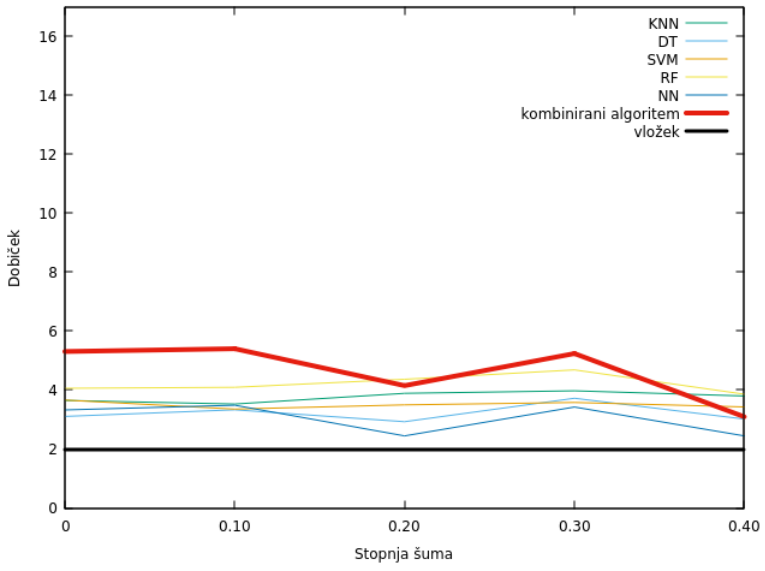
sinusnih funkcij naslednje oblike:

$$TS(t) = \sin(\omega t + d) + rand()st_{sum} - st_{sum}/2 + 0.6 \tag{5.1}$$

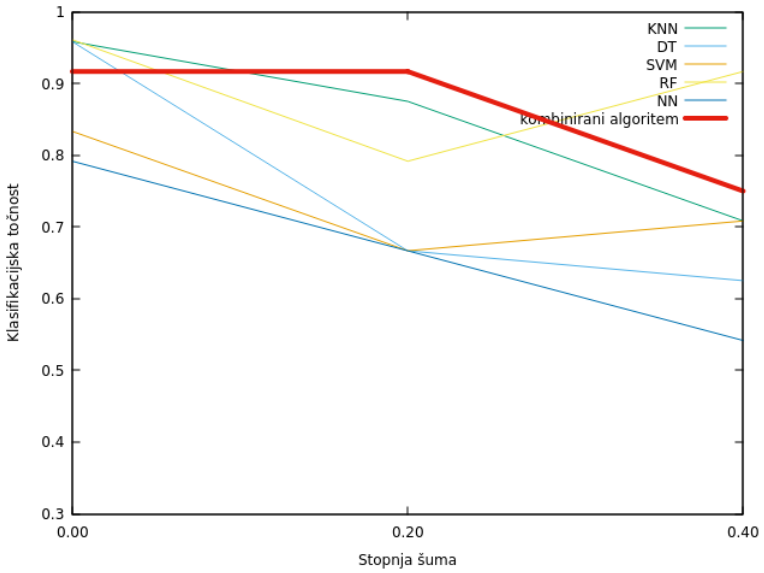
st_{sum} predstavlja stopnjo šuma, ki se je spreminjal od 0 do 0,40 s korakom 0,10. ω je bil nastavljen v prvi časovni vrsti na 0,25, v drugi na 0,15, v tretji na 0,5, v četrti na 0,7 in v peti časovni vrsti na 1,2. d je bil nastavljen v prvi časovni vrsti na 0,0, v drugi na 0,5, v tretji na 0,6, v četrti na 1,5 in v peti časovni vrsti na 0,3. Dobljene časovne vrste se je seštelo v novo vrsto, ki jo je bilo potrebno napovedati. Odvisnost ciljne časovne vrste od vzporednih časovnih vrst je torej podana z vsoto teh časovnih vrst. Tako dobljene časovne vrste se je razdelilo na testne in učne primere, kakor v prejšnjem razdelku, samo da je je bil d nastavljen na 25. To je pomenilo, da se je napovedovalo za 25 dni. Iz Slike 5.2, kjer je prikazana klasifikacijska točnost glede na stopnjo šuma, je razvidno, da je kombinirani algoritem primerljiv z obstoječimi algoritmi. Podobno lahko sklepamo iz Slike 5.3, kjer je prikazana mera dobička glede na stopnjo šuma. V tem primeru lahko rečemo, da je kombinirani algoritem ob nižji stopnji šuma boljši od obstoječih algoritmov.



Slika 5.2: Graf klasifikacijske točnosti glede na stopnjo šuma za vsoto petih časovnih vrst.



Slika 5.3: Graf dobička glede na stopnjo šuma za vsoto petih časovnih vrst.



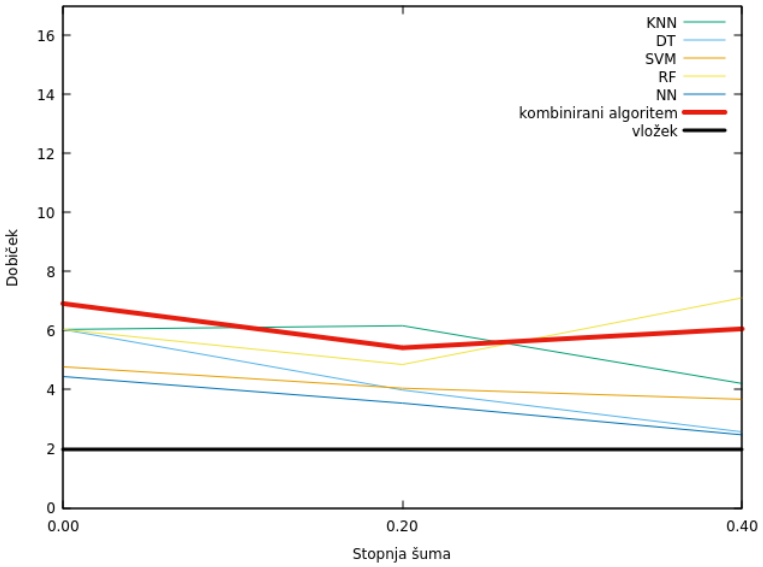
Slika 5.4: Graf klasifikacijske točnosti glede na stopnjo šuma časovne vrste s sinusno funkcijo, 5 časovnih vrst.

Naslednja podatkovna zbirka je sestavljena podobno kot prejšnja, le da je odvisnost ciljne časovne vrste od štirih vzporednih časovnih vrst bolj zapletena in je podana z:

$$TS_0(t) = \sin(TS_1(t)) \sin(TS_2(t)) + \sin(TS_3(t)) \sin(TS_4(t))$$

Indeksi ob časovni vrsti predstavljajo, za katero časovno vrsto gre. Iz Slike 5.4 in Slike 5.5 je razvidno, da je kombinirani algoritem primerljiv z obstoječimi algoritmi. Zanimiv je potek grafa dobiček (gl. Sliko 5.3), ki kljub padanju klasifikacijske točnosti narašča ob šumu stopnje 0,40. Pri tem je potrebno omeniti, da mera dobička in klasifikacijska točnost nista nujno kolerirani. To pomeni, da imamo lahko ob višji klasifikacijski točnosti nižji dobiček ali obratno. Razlaga je v tem, da je algoritem pravilneje napovedal za višje skoke časovne vrste, kar pomeni, da je dobiček zaradi tega pozitiven.

Naslednja umetna podatkovna zbirka je sestavljena iz 100 vzporednih časovnih vrst oblik iz funkcije (5.1). Pri tem se je za ciljno časovno vrsto

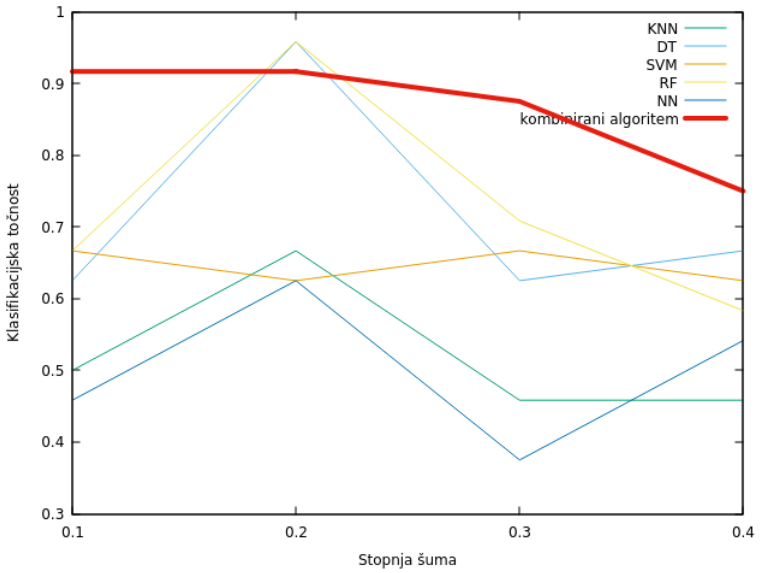


Slika 5.5: Graf dobička glede na stopnjo šuma časovne vrste s sinusno funkcijo, 5 časovnih vrst.

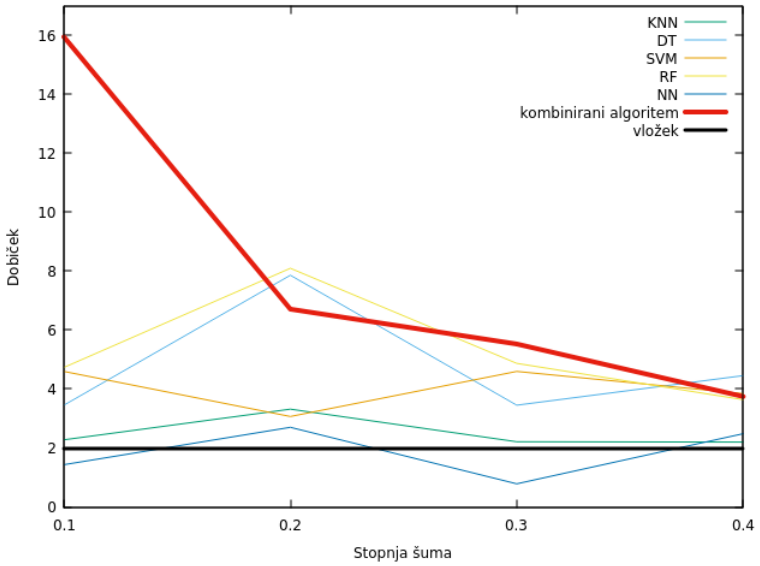
TS_0 vzelo seštevek 10., 20., 30., 40. in 50. časovne vrste.

$$TS_0(t) = TS_{10}(t) + TS_{20}(t) + TS_{30}(t) + TS_{40}(t) + TS_{50}(t)$$

Rezultate algoritmov na tej podatkovni zbirki prikazujeta grafa 5.6 in 5.7. Pri tem smo analizirali tudi algoritme na binarno transformiranih podatkih – gl. Sliko 5.8 in Sliko 5.9, kjer je razvidno, da je kombinirani algoritem slab, saj je klasifikacijska točnost okoli točnosti večinskega razreda. Prav tako je dobiček skoraj povsod pod ostalimi algoritmi.

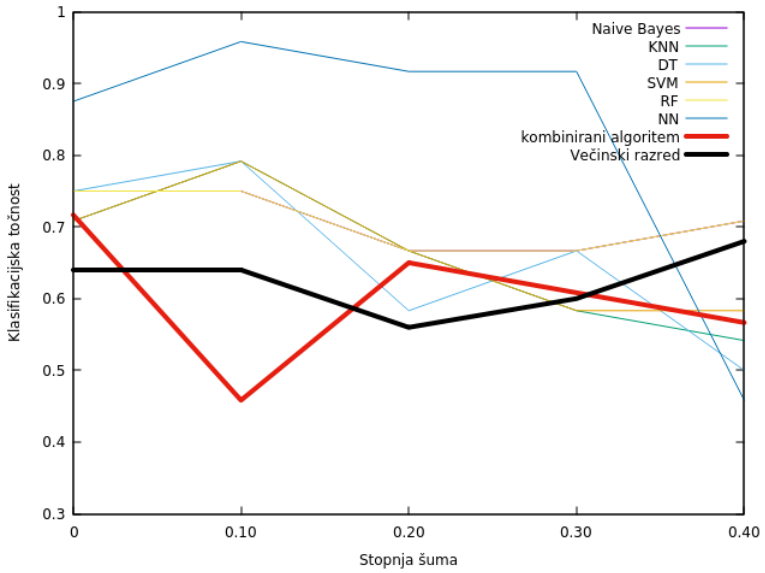


Slika 5.6: Graf klasifikacijske točnosti glede na stopnjo šuma časovne vrste s sinusno funkcijo, 100 časovnih vrst.

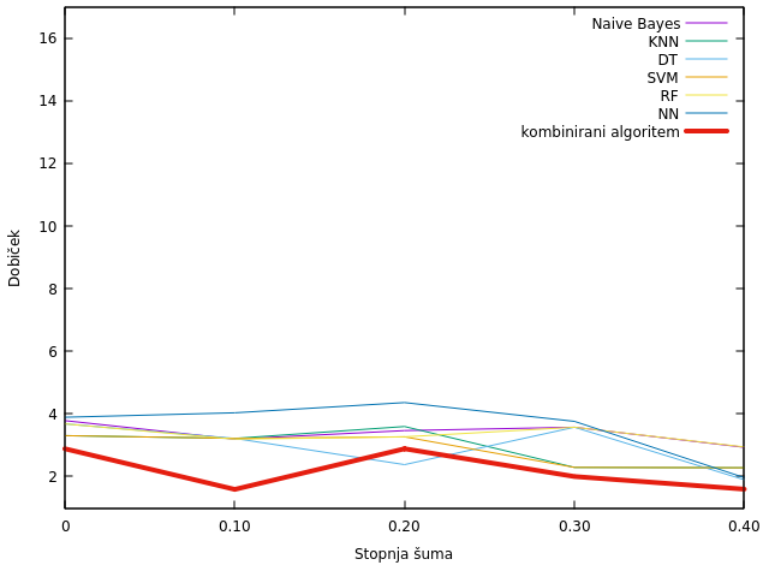


Slika 5.7: Graf dobička glede na stopnjo šuma časovne vrste s sinusno funkcijo, 100 časovnih vrst.

Rezultati na umetno generiranih podatkih kažejo, da je kombiniran algo-



Slika 5.8: Graf klasifikacijske točnosti glede na stopnjo šuma časovne vrste s sinusno funkcijo, 100 časovnih vrst, binarna transformacija.



Slika 5.9: Graf dobička glede na stopnjo šuma časovne vrste s sinusno funkcijo, 100 časovnih vrst, binarna transformacija.

ritem dober za zvezne in ne preveč zašumljene podatke, saj v teh primerih napoveduje bolje kot obstoječi algoritmi. Omeniti je potrebno, da algoritem

ni preiskal vseh možnih kombinacij funkcij in sit. To je tudi logično, saj so bili umetno generirani podatki na tej podatkovni zbirki oblike sinus in je linearna regresija hitro našla prave koeficiente funkcij.

5.2 Realni podatki

Realne podatke se je pridobilo na internetni strani `www.quandl.com`, kjer so podatki o vrednostnih papirjih. Podatke se je zajelo od leta 1980 do leta 2017 za približno 3.600 papirjev. Iz podatkovne zbirke se je izbrisalo tiste papirje, ki niso imeli popolnih podatkov za 4.000 delovnih dni v preteklost. Slabost kombiniranega algoritma je namreč v tem, da potrebuje popolne podatke v tabeli. Na ta način se je pridobilo 1.600 vrednostnih papirjev, ki jih je bilo potrebno razvrstiti po datumu. Nato se je izbralo devet vrednostnih papirjev, za katere se je napovedovalo za en dan naprej, kako se bodo gibale njihove vrednosti. Za d se je vzelo 30 dni. Podobno kot pri umetnih podatkovnih zbirkah se je izračunala točnost in mera dobička glede na vložek 10.000 denarnih enot. Vrednostni papirji, ki se jih je uporabilo za napovedovanje, so bili naslednji:

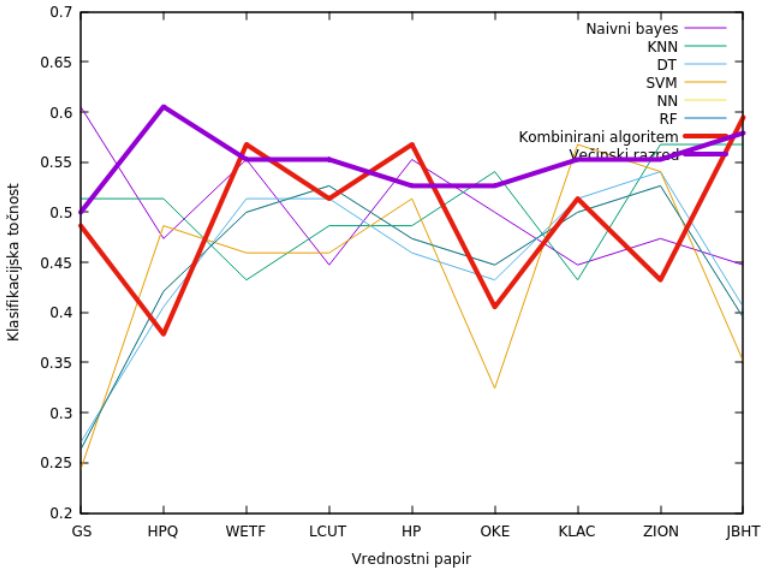
- **GS** – "Goldman Sachs Group Inc."
- **HPQ** – "HP Inc."
- **WETF** – "Wisdom Tree Investments Inc."
- **LCUT** – "Lifetime Brands Inc."
- **HP** – "Helmerich & Payne, Inc."
- **OKE** – "ONEOK, Inc."
- **KLAC** – "KLA-Tencor Corp."
- **ZION** – "Zions Bancorp"
- **JBHT** – "J B Hunt Transport Services Inc."

Vrednostni papir	Večinski razred	Klasifikacijska točnost
GS	0/1	0.5
HPQ	0	0.61
WETF	1	0.55
LCUT	0	0.55
HP	1	0.53
OKE	0	0.52
KLAC	0	0.55
ZION	1	0.55
JBHT	1	0.58
Povprečje		0.55
Standardna deviacija		0.031
Standardna deviacija v %		5.59 %

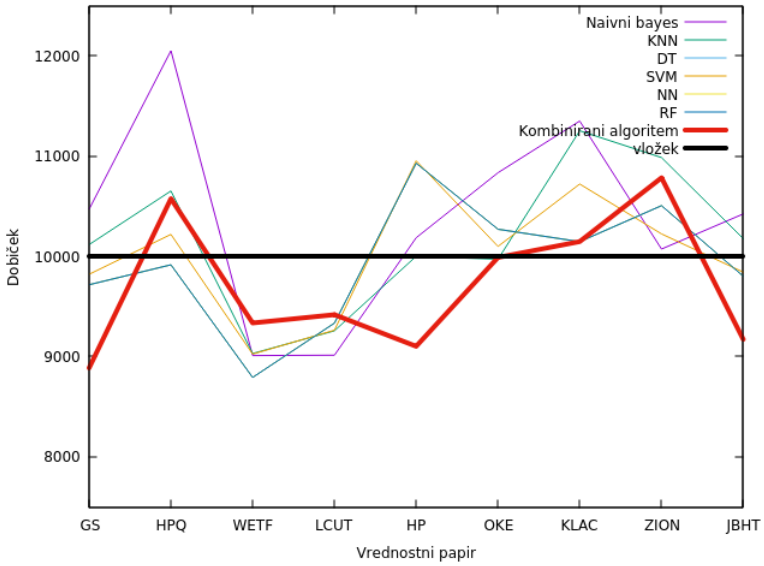
Tabela 5.1: Klasifikacijska točnost večinskega razreda

Večinski razred za te vrednostne papirje prikazuje tabela 5.1. Algoritme smo poganjali na dveh različicah podatkov: na originalnih in na binarnih, kjer sta podani samo vrednosti -1 v primeru, da vrednost delnice pada, oziroma 1 v primeru, da vrednost delnice raste. Rezultate napovedi si lahko ogledamo na grafih 5.10, 5.11, 5.12 in 5.13. Iz grafov lahko sklepamo, da so rezultati slabi. Vsi algoritmi imajo namreč klasifikacijske točnosti okoli klasifikacijske točnosti večinskega razreda na binarnih in originalnih podatkih. Kombinirani algoritem je tudi v tem primeru primerljiv z obstoječimi algoritmi.

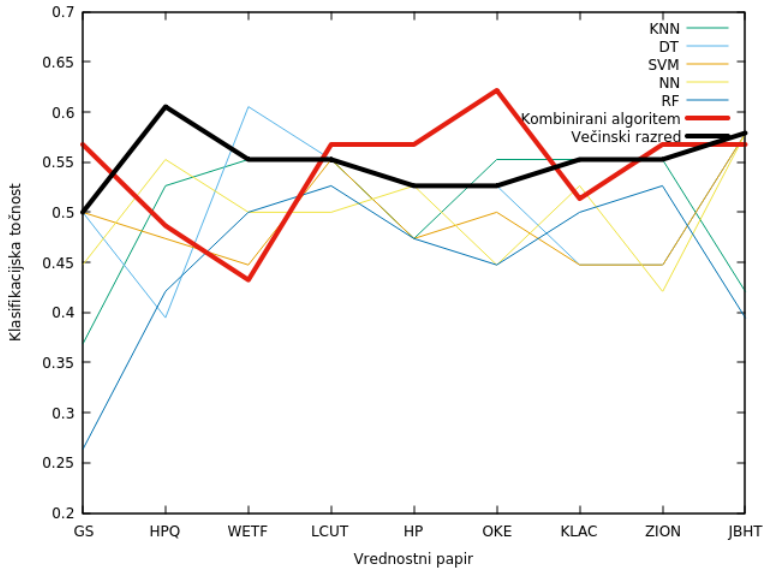
V fazi primerjanja algoritmov na realnih podatkih smo za primerjavo uporabili tudi preprost algoritem, ki je napovedoval na podlagi prejšnjega gibanja vrednosti vrednostnih papirjev; če je prejšnji dan papir rastel, je napovedal, da bo v prihodnosti rastel, in obratno, če je prejšnji dan padal, je napovedal, da bo v prihodnosti padal. Rezultate v primerjavi z kombiniranim algoritmov prikazujeta grafa 5.14 in 5.15, kjer je razvidno, da je preprost algoritem po klasifikacijski točnosti slabši od kombiniranega algoritma. Pri primerjavi po meri dobička preprosti algoritem ne presega črte vložka, kar



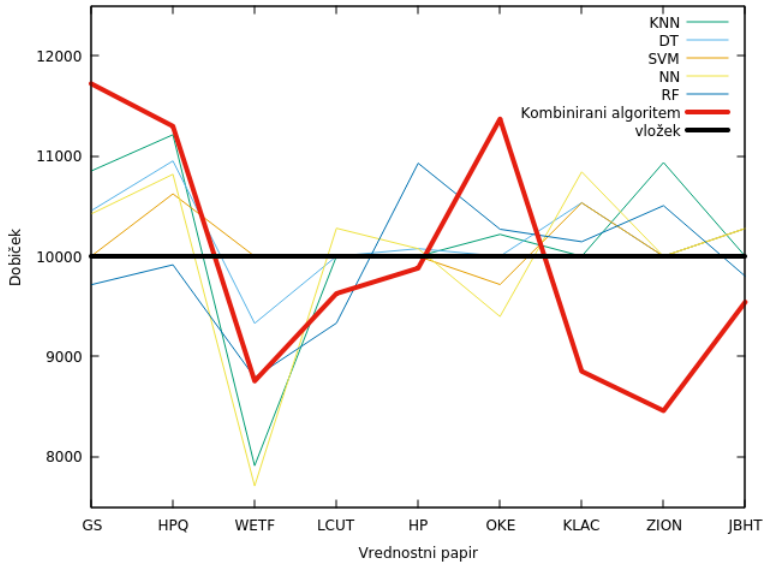
Slika 5.10: Graf klasifikacijske točnosti napovedovanja vrednostnih papirjev, binarna transformacija.



Slika 5.11: Graf mere dobička napovedovanja vrednostnih papirjev, binarna transformacija.

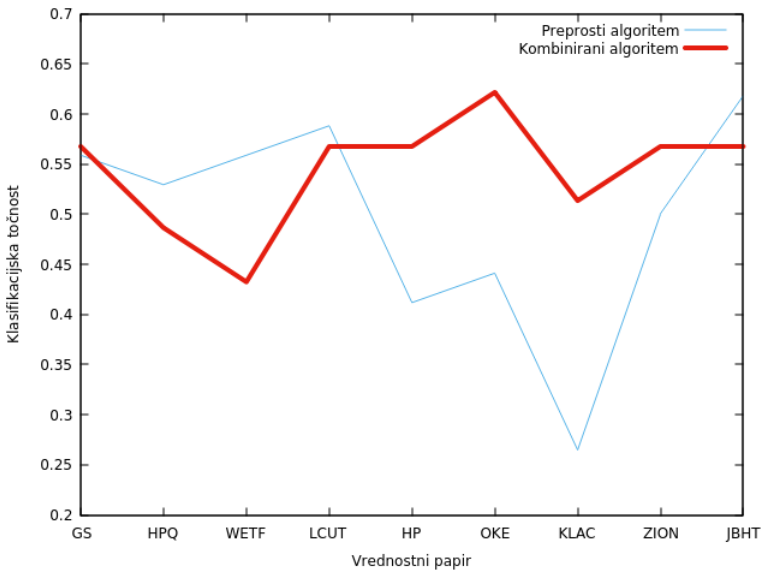


Slika 5.12: Graf klasifikacijske točnosti napovedovanja vrednostnih papirjev, originalni podatki.

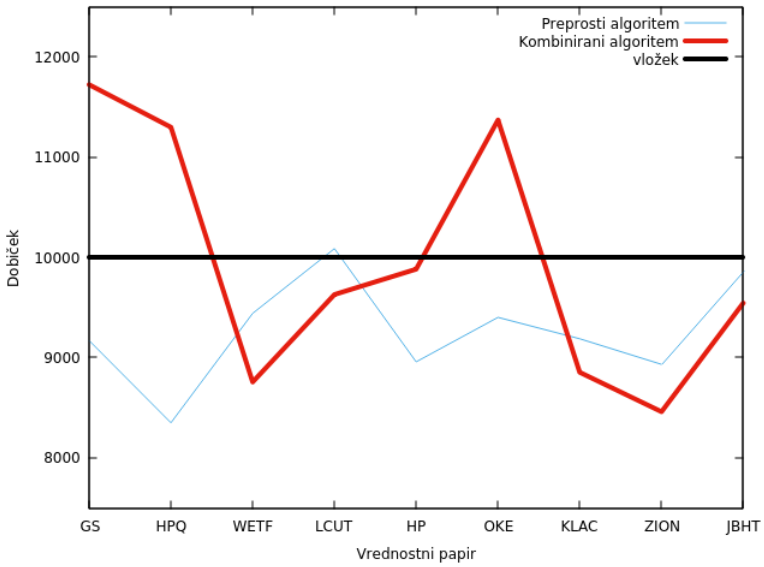


Slika 5.13: Graf mere dobička napovedovanja vrednostnih papirjev, originalni podatki.

pomeni, da prinaša izgubo.

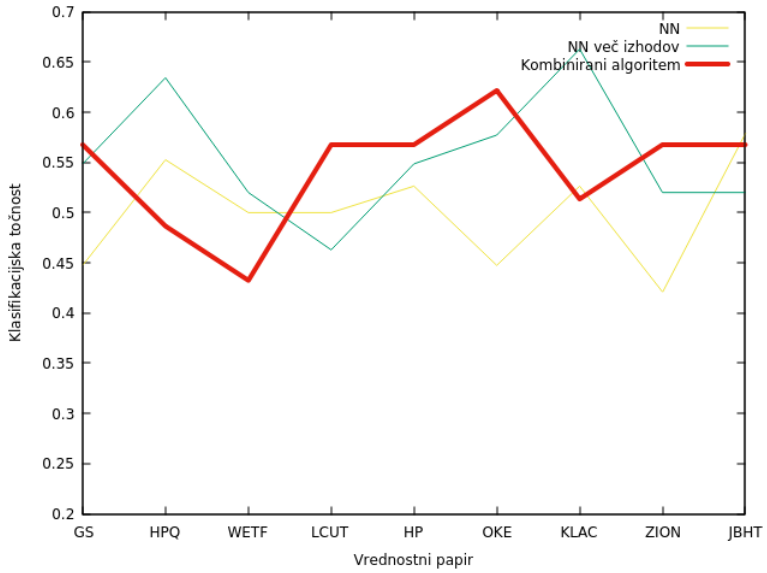


Slika 5.14: Graf klasifikacijske točnosti napovedovanja vrednostnih papirjev, preprosti algoritem.



Slika 5.15: Graf mere dobička napovedovanja vrednostnih papirjev, preprosti algoritem.

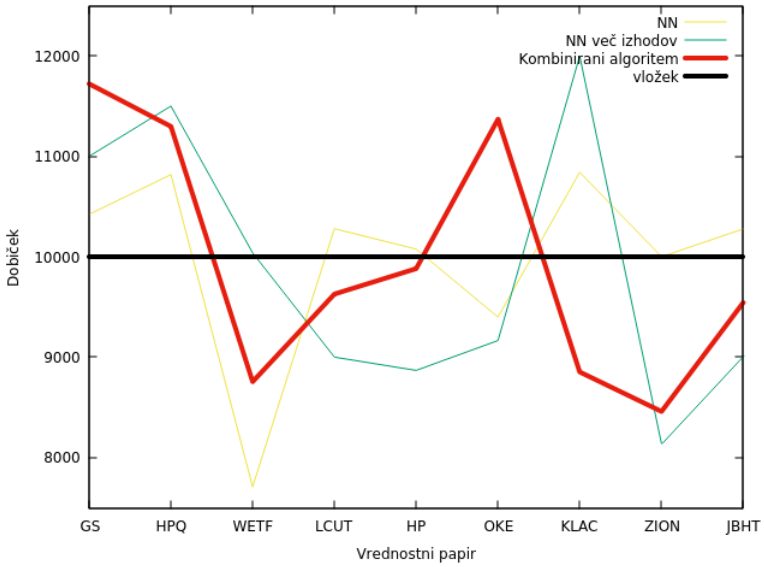
Pri analizi se je pokazalo, da je nevronska mreža nekoliko slabše napovedovala. Zato je nastala ideja, da bi nevronsko mrežo sestavili z več izhodi, ki predstavljajo devet napovednih vrednostnih papirjev. Na ta način bi se ena mreža lahko naučila napovedovati gibanja cen za vse vrednostne papirje naenkrat, če obstajajo zakonitosti, ki povezujejo te vrednostne papirje med seboj. Rezultate primerjave prikazujeta grafa 5.16 in 5.17. V tem primeru, je klasifikacijska točnost za nevronsko mrežo z več izhodi nekoliko boljša od nevronske mreže z enim izhodom. Toda po meri dobička nevronska mreža z več izhodi ni bistveno izboljšala napovedi.



Slika 5.16: Graf klasifikacijske točnosti napovedovanja vrednostnih papirjev, nevronska mreža z več izhodi.

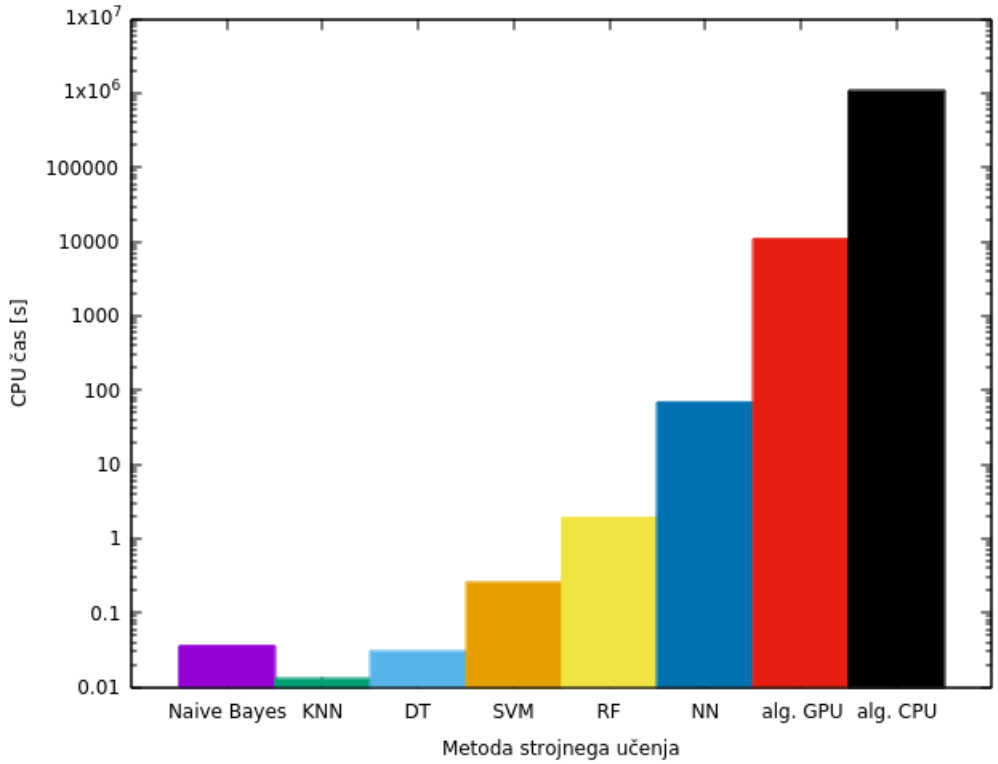
5.3 Časovna primerjava algoritmov

Algoritme smo primerjali tudi po času, ki ga porabijo skupaj za učenje in napovedovanje. Rezultati meritev so na grafu 5.18, ki ima logaritmično časovno os. Za izvajanje kombiniranega algoritma na CPU se je uporabila ocena, ki je predvidevala, da je izvajanje na GPU $640/2 = 320$-krat hitrejša. GPU, na



Slika 5.17: Graf mere dobička napovedovanja vrednostnih papirjev, nevronska mreža z več izhodi.

katerem se je primerjalo algoritme, ima 640 jeder pri 1096 MHz, pri čemer je imel CPU 2.60GHz. Pri tem je potrebno še všteti prenos podatkov iz in na GPU. Iz grafa je razvidno, da je kombinirani algoritem kljub pohitritvi zelo počasen.



Slika 5.18: Graf časovne potratnosti algoritmov, CPU čas je v logaritemski skali. Zadnja, črna vrednost, je samo ocenjena in ne dejansko izmerjena.

Poglavje 6

Sklepne ugotovitve

Novo predlagani kombinirani algoritem, ki je sestavljen iz genetskega algoritma in nelinearne regresije, je primerljiv glede točnosti in mere dobička z obstoječimi algoritmi strojnega učenja, v nekaterih primerih celo boljši. Prednost je v tem, da za vhodne podatke ne rabi posebnih predobdelav podatkov, dokler so ti polni (tabela nima manjkajočih podatkov). Prednost je tudi v tem, da ponuja razlago, kako so podatki odvisni drug od drugega. Funkcija, ki jo najde, se namreč prilagaja podatkom in kaže na odvisnost med vzporednimi časovnimi vrstami. Slabost algoritma je v časovni potratnosti, saj je v primerjavi z ostalimi algoritmi počasen. Prav tako je slabost v tem, da ne zna obravnavati manjkajočih vrednosti.

Sledijo odgovori na zastavljena vprašanja iz razdelka 1.1.

1. Ali je mogoče napovedovati vzporedne časovne vrste (vrednostni papirji)?

   Z obstoječimi algoritmi in s kombiniranim algoritmom je napovedovanje zelo težavno. Dobljeni rezultati kažejo nato, da na realnih vrednostih vrednostnih papirjev vsi algoritmi slabo napovedujejo novo vrednost. Poleg tega se nakazuje, da vrednostni papirji niso toliko odvisni drug od drugega ali pa je bil vzorec, na katerem se je kombinirani algoritem učil, premajhen. Samo z večjim vzorcem bi za iskanje optimalnega sita

potrebovali močnejše računalnike, ki bi bili povezani med sabo.

2. Katere metode strojnega učenja so boljše po točnosti in po časovni potratnosti?

   Na to vprašanje je težko odgovoriti, saj pri analizi po točnosti ni izstopal nobeden algoritem. Po časovni potratnosti je kombinirani algoritem, da je dosegel vsaj osem generacij, kljub pohitritvi zelo počasen. Vsekakor pa je kombiniran algoritem primerljiv z obstoječimi algoritmi. Tukaj je potrebno ponoviti, da je kombinirani algoritem deloval slabše na binarnih podatkih kot na originalnih. To si lahko razlagamo tako, da so bili vhodni podatki preveč diskretni za nelinearno regresijo.

3. Katera metodologija primerjanja algoritmov je najboljša?

   V magistrski nalogi sta uporabljeni dve metodi za primerjanje algoritmov: mera klasifikacijska točnost in mera "dobička", ki nista čisto korelirani. Ti meri imata eno skupno točko, in sicer če je klasifikacijska točnost 100-odstotna, je mera dobička v tem primeru maksimalna. Definirana mera "dobiček"ima slabost, da ne deluje na negativnih številih. Drug problem je v tem, da če algoritem napove, da bodo vse vrednosti časovne vrste padale, se vrednost dobička ne spreminja, ker v tem primeru strategija vlaganja nikoli ne kupuje. Zaradi tega tudi ni bilo možno izračunati dobička napovedi večinskega razreda.

4. Kako dober je novo predlagan kombiniran algoritem, ki je sestavljen iz genetskega algoritma in nelinearne regresije?

   Novo predlagani kombiniran algoritem je vsekakor primerljiv z obstoječimi algoritmi, saj dosega podobne rezultate na istih podatkih kot ostali, dokler so podatki zvezni.

5. Na kakšen način lahko pohitrimo kombiniran algoritem?

   V osnovi je bil algoritem razvit z visoko stopnjo paralelnosti na grafičnem procesorju. Toda kljub temu je za izračun osmih generacij v kombiniranem algoritmu potreboval vsaj tri ure. Če bi se algoritem izvajal na

centralnem procesorju, bi se izvajal vsaj nekaj 100-krat dlje. Možna dodatna pohitritev bi lahko bila uporaba več računalnikov, povezanih med seboj.

6.1 Bodoče delo

V bodočem delu bi bilo dobro poleg pohitritve izboljšati algoritem še v smislu, da bi genomom in funkcijam dodali hevristično oceno, ki bi usmerjala preiskovanje, kajti prostor funkcij in sita je še vedno ogromen. Druga zanimiva rešitev, ki bi potrebovala veliko časa, je modeliranje s pomočjo parcialnih diferencialnih enačb z robnimi pogoji. Za ta problem bi lahko vzeli podatke iz CERNa o trkih osnovnih delcev. Trki osnovnih delcev in njihova trajektorija namreč pripadajo neki parcialni diferencialni enačbi, ki jih opisuje fizikalni model. Algoritem bi se lahko prilagodil tako, da bi namesto iskanja optimalne funkcije iskal optimalno diferencialno enačbo.

Literatura

[1] Khalid Alkhatib, Hassan Najadat, Ismail Hmeidi, Mohammed K. Ali Shatnawi, *Stock Price Prediction Using K-Nearest Neighbor (kNN) Algorithm*, International Journal of Business, Humanities and Technology, 3 (2013) 32–44.

[2] Bonde, G. , Khaled, R., *Stock price prediction using genetic algorithms and evolution strategies*, Proceedings of the 2012 International Conference on Genetic and Evolutionary Methods, (2012) 10–15.

[3] Steven C. Chapra, Raymond P. Canale, *Numerical Methods for Engineers, sixth edition*, McGraw-Hill, 2010.

[4] Ayon Dey, *Machine Learning Algorithms: A Review*, International Journal of Computer Science and Information Technologies, 7 (2016) 1174–1179.

[5] Sadegh Bafandeh Imandoust, Mohammad Bolandraftar, *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background*, Journal of Engineering Research and Application, 3 (2013) 605–610.

[6] Igor Kononenko, Marko Robnik Šikonja, *Inteligentni sistemi*, Založba FE in FRI, (2010).

[7] Igor Kononenko, *Strojno učenje*, Založba FE in FRI, (2005).

[8] Jason W. Leung, *Application of Machine Learning: Automated Trading Informed by Event Driven Data*, Master's thesis, Massachusetts Institute of Technology, Massachusetts (2016).

[9] Luckyson Khaidem, Snehanshu Saha, Sudeepa Roy Dey, *Predicting the direction of stock market prices using random forest*, Applied Mathematical Finance, 3 (2016) 10–30.

[10] Zabir Haider Khan, Tasnim Sharmin Alin, Akter Hussain,*Price Prediction of Share Market using Artificial Neural Network (ANN)*, International Journal of Computer Applications, 22 (2011) 42–47.

[11] Saahil Madge, *Predicting Stock Price Direction using Support Vector Machines*, Independent Work Report Spring (2015) Princeton.

[12] Nirbhey Singh Pahwa, Neeha Khalfay, Vidhi Soni,Deepali Vora, *Stock Prediction using Machine Learning a Review Paper*, International Journal of Computer Applications, 163 (2017).

[13] Vishal Parikh, Parth Shah, *Stock Prediction and Automated Trading System*, IJCS, 6 (2015) 104–111.

[14] Mahajan Shubhrata, Deshmukh Kaveri, Thite Pranit, Samel Bhavana, Chate P., *Stock Market Prediction and Analysis Using Naïve Bayes*, International Journal on Recent and Innovation Trends in Computing and Communication, 4 (2017) 121–124.

[15] Shai Shalev-Shwartz, Shai Ben-David, *Understanding Machine Learning:From Theory to Algorithms*, Cambridge University Press, (2014).

[16] Tzung-I Tang, Gang Zheng, Yalou Huang, Guangfu Shu, Pengtao Wang, *A Comparative Study of Medical Data Classification Methods Based on Decision Tree and System Reconstruction Analysis*, IEMS, 4 (2005) 102–108.

[17] C. Tsai, S. Wang, *Stock Price Forecasting by Hybrid Machine Learning Techniques*, Proceedings of the International Multi Conference of Engineers and Computer Scientists, 1 (2009) 755–760.

[18] Jan Adamowski, Hiu Fung Chan, Shiv O. Prasher, Bogdan Ozga-Zielinski, Anna Sliusarieva, *Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada*, Water Resources Research, 48 (2012).