UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Luka Jalšovec

Podatkovno rudarjenje in napovedovanje gibanja trga delnic z metodo podpornih vektorjev in nevronskih mrež

Diplomsko delo

Ljubljana, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Luka Jalšovec
Mentor: doc. dr. Damjan Škulj

Podatkovno rudarjenje in napovedovanje gibanja trga delnic z metodo podpornih vektorjev in nevronskih mrež

Diplomsko delo

Ljubljana, 2016

Zahvala

Zahvaljujem se mentorju doc. dr. Damjanu Škulju za vso pomoč, strokovno vodenje, nasvete in potrpežljivost ter staršem in partnerici za vso podporo.

Podatkovno rudarjenje in napovedovanje gibanja trga delnic z metodo podpornih vektorjev in nevronskih mrež

Finančni trgi so zelo kompleksni sistemi, ki proizvajajo množice podatkov z veliko šuma in visoko stopnjo negotovosti. Analiza finančnih trgov je zelo razvito raziskovalno področje in investitorjem nudi pomembna orodja pri načrtovanju investicij, ki lahko z dobro zastavljeno strategijo vlaganja prinesejo velike dobičke. Diplomsko delo predstavlja uporabo dveh izbranih metod strojnega učenja na področju časovnih vrst in posebej pri napovedovanju gibanja tečajev delnic. Namen je preveriti učinkovitost uporabe metode podpornih vektorjev in nevronskih mrež pri napovedovanju spreminjanja vrednosti in smeri borznih indeksov. V nalogi so zajeti trije borzni indeksi, in sicer Nikkei 225 z japonske borze, Veurx, ki predstavlja evropski delniški trg, in S&P 500 z borze v New Yorku. Podatki za analizo so vrednosti naštetih indeksov ob koncu dneva. Učinkovitost obeh metod primerjamo z izračunom normalizirane srednje kvadratne napake, sredinske absolutne napake in smerne simetrije. Metoda podpornih vektorjev se za napovedovanje delniških trgov izkaže kot bolj točna od metode splošnih nevronskih mrež.

Ključne besede: metoda podpornih vektorjev, nevronske mreže, strojno učenje, časovne vrste, delniški trgi.

Data mining and stock market predictions using support vectors machines and neuron networks

Financial markets are very complex systems, producing large quantities of data, with a lot of noise and high degree of uncertainty. Financial market analysis is an important and highly developed research area, providing tools for investors to make informed decisions, minimize risks and make better investment strategies leading to higher profits. This thesis will introduce the use of two selected machine learning methods for time series with the emphasis on stock market movements. The goal is to test the effectiveness of support vector machines and neural networks to predict future movements of stock market indexes. The analysis will be conducted on the following stock indexes: Nikkei 225 from Japan stock market, Veurx, European stock index and S&P 500 from the New York stock exchange. The data used for the analysis are the closing price of the selected indices. The effectiveness of methods is evaluated by the following measures: normalized mean squared error, mean absolute errors and directional symmetry. The results show that the methods of support vector machines are more accurate for predicting the future movements of stock markets than the back propagation neuron networks.

Key words: support vector machines, neural networks, machine learning, time series, stock market.

Kazalo

Kazalo slik

Kazalo tabel

1 Uvod

Človek se vsak dan srečuje s spremembami v svetu. Na posameznika in družbo imajo te različne vplive in vsak se z njimi spopada drugače. Izogniti se želimo negativnim posledicam sprememb, zato se je potreba po napovedovanju sprememb pojavila že zelo zgodaj. Dandanes pa se z napovedovanjem sprememb srečujemo še pogosteje. Z različnimi poskusi in metodami želimo napovedati, kaj nas čaka v prihodnosti. Vsakodnevna napoved, ki jo spremlja večina ljudi, je napoved vremena in vremenskih pojavov. Ljudje takim napovedim dajemo zelo velik pomen in po njih tudi načrtujemo naše življenje.

V zadnjih letih smo priča razmahu in razvoju informacijske tehnologije na vseh področjih življenja. Sočasno in prepleteno z informacijsko tehnologijo se razvijajo tudi različne tehnike strojnega učenja, ki se implementirajo v različne procese v IT-panogah. Za mojo diplomsko nalogo je pomembna predvsem integracija omenjenih metod v procese podatkovnega rudarjenja oziroma iskanja novih znanj po bazah podatkov. Tudi v poslovnem in finančnem svetu napredka v tehnologiji niso zanemarili, saj ga veliko podjetij vključuje v svoje delovanje. Tudi gospodarsko okolje se zelo hitro spreminja, kar ima velike posledice za delovanje podjetij. Če se ta želijo pripraviti na posledice gospodarskih sprememb, morajo z različnimi metodami napovedovati gibanja v gospodarskem svetu. Lahko rečemo, da je današnje gospodarsko okolje postalo zelo kompleksno in konkurenčno, predvsem pa težko predvidljivo. Zato se različni raziskovalci posvečajo prav implementiranju novih statističnih metod in metod strojnega učenja v modele za napovedovanje gibanja trgov.

S podobnimi pojavi se srečujemo tudi na delniških trgih. Kot že omenjeno, smo v zadnjih treh desetletjih priča spremembam v finančnem okolju. Prav razvoj metod napovedovanja in samega trgovanja z vrednostnimi papirji je investitorjem povečal možnost zaslužka. Prav tako so se spremenile same teorije o delniških trgih in metode finančnih analiz so precej napredovale. Napovedovanje gibanja cen delnic in gibanja delniških indeksov je postalo pomemben finančni subjekt, ki še danes, kot že omenjeno, privablja pozornost številnih raziskovalcev. Ker je veliko finančnih informacij postalo javnih, predvsem cene delnic, volumen trgovanja in vrednosti delniških indeksov ter ekonomskih spremenljivk, je prišlo tudi do razmaha metod napovedovanja trendov na finančnih trgih (Enke in Thawornwong 2005).

Finančni trgi proizvajajo velike količine podatkov, ki so v svoji naravi nelinearni, z veliko šuma in se nenehno kopičijo. Ti podatki vsebujejo visoko stopnjo negotovosti ter skritih relacij. Na finančne trge poleg samega gospodarstva vpliva veliko drugih dogodkov, kot so politične spremembe in dogodki v svetu. Zaradi tega je napovedovanje trga zelo oteženo in kompleksno, kljub temu pa zelo pomembno področje, saj nam analiza teh da pomembne informacije, na podlagi katerih lahko sprejemamo strateške odločitve, tako v smislu investicij v delnice kot načrtovanja delovanja podjetja.

1.1 Namen in cilj diplomskega dela

V diplomski nalogi želim predstaviti uporabo podatkovnega rudarjenja na področju delnic in delniškega trga. Bolj natančno se želim poglobiti v dve izbrani metodi podatkovnega rudarjenja, in sicer v metodo podpornih vektorjev in nevronskih mrež. Obe metodi sta se že uveljavili kot učinkovita mehanizma za napovedovanje gibanja delniških trgov in cen delnic. Predvsem me zanima, katera izmed metod bo natančneje napovedala izhodno vrednost izbranega finančnega instrumenta. Z izbranim finančnim instrumentom bom lahko določal spremembe vrednosti delniškega indeksa za določeno časovno obdobje. V diplomskem delu bom poskusil prikazati, katera izmed izbranih metod podatkovnega rudarjenja bo natančneje napovedovala vrednost spremembe indeksa v prihodnjih dneh. Natančnost oziroma uspešnost bom preverjal s statističnimi kazalniki, ki jih bom definiral pozneje v diplomski nalogi.

Raziskovalno vprašanje diplomske naloge je torej: Katera izmed izbranih metod podatkovnega rudarjenja bo natančneje napovedala vrednost spremembe borznega indeksa?

2 Koncept podatkovnega rudarjenja

Podatkovno rudarjenje je poznano tudi kot odkrivanje znanj v podatkih. V procesu »rudarjenja« odkrivamo zanimive vzorce v podatkovnih bazah, ki nam lahko pomagajo pri različnih odločitvah. Podatkovno rudarjenje je disciplina, ki postaja čedalje bolj pomembna in zanimiva. Metode podatkovnega rudarjenja lahko podjetjem, ki so jih upoštevala pri svojih odločitvah, pripomorejo k strateški prednosti (Bose in Mahapatra 2001).

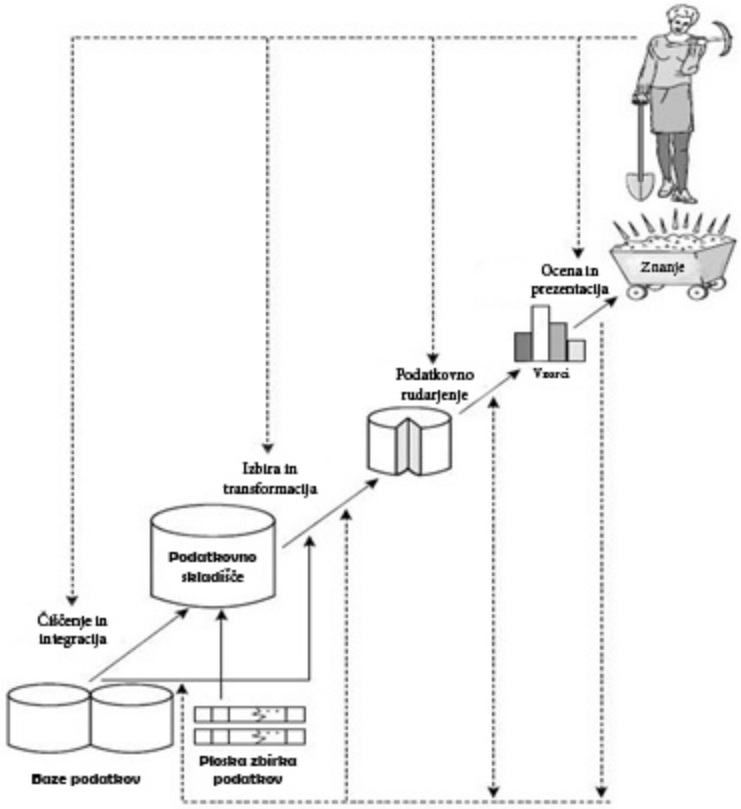
Ideja o iskanju zanimivih vzorcev v poslovnem podatkovju ni nova. Prvenstveno je bila to naloga analitikov, ki so v bazah podatkov iskali vzorce s pomočjo statističnih metod. Skozi čas pa so se te metode čedalje bolj spreminjale. Za tovrstne spremembe je v prvi vrsti zaslužen prav razmah uporabe spletnih tehnologij, ki tvorijo čedalje večje podatkovne baze (Bose in Mahapatra 2001).

Če preprosto povzamem, lahko rečemo, da je podatkovno rudarjenje iskanje znanja po velikih podatkovnih bazah. Pravilnejši izraz bi bil seveda »rudarjenje znanja po bazah podatkov«, vendar je ta predolg, zato se je obdržala fraza podatkovno rudarjenje. Zelo popularen pa je tudi izraz »razkrivanje znanj iz podatkov« oziroma angleško »Knowledge Discovery from Data«, KDD (Han in Kamber 2011).

Iskanje znanj je proces, ki je sestavljen iz naslednjih korakov:

1. Čiščenje podatkov (odstranjevanje šuma in nekonsistentnih podatkov)
2. Integriranje podatkov (združevanje več virov podatkov)
3. Izbor podatkov (izvlečejo se podatki, ki so relevantni za analizo)
4. Transformacija podatkov (transformacija podatkovja v obliko, primerno za rudarjenje)
5. Podatkovno rudarjenje (esencialni proces, pri katerem se uporabijo določene metode za iskanje vzorcev v podatkih)
6. Ocena vzorca
7. Predstavitev pridobljenih znanj (z vizualizacijo in znanjem predstavimo poiskano znanje uporabnikom)

Slika 2.1: Koraki podatkovnega rudarjenja.



Vir: Han in Kamber (2011).

Prvi štirje koraki so različne oblike procesiranja podatkov, s katerimi se jih pripravi na proces podatkovnega rudarjenja. Podatkovno rudarjenje je po zgornji shemi le eden izmed korakov v procesu iskanja znanj v bazah podatkov, vendar pa je ta najpomembnejši, saj nam priskrbi različna nova znanja, poiskana v podatkovnih bazah (Han in Kamber 2011).

V procesu podatkovnega rudarjenja se podatki analizirajo tako, da se med njimi iščejo vzorci in podobnosti oziroma modeli, ki bi predstavljali neke relacije med podatki. Dobljeni modeli se nato preverjajo z novimi seti podatkov, da lahko zagotovimo splošnost modela. Modele, ki se dobijo skozi proces podatkovnega rudarjenja, je treba pozneje prevesti v poslovne načrte, ki bi lahko organizacijam pomagali pri doseganju poslovnih ciljev. Modeli, ki zadovoljijo te potrebe, postanejo poslovno znanje (Bose in Mahapatra 2001).

Na področju podatkovnega rudarjenja so se razvile številne metode in algoritmi, denimo metode klasifikacije, odločitvena drevesa in asociacijska pravila, s katerimi odkrivamo vzorce v danih podatkih. Med omenjenimi so klasični statistični pristopi med najstarejšimi. Podatkovni seti, v katerih se uporabljajo klasične statistične metode, morajo izpolnjevati

različne kriterije porazdelitve, da bi lahko te metode uporabljali. Algoritmi, ki za odkrivanje vzorcev uporabljajo druge metode učenja, po drugi strani ne potrebujejo tako strogo urejenega podatkovja (Bose in Mahapatra 2001).

2.1 Podatkovno rudarjenje časovnih vrst

V diplomski nalogi se bom posvečal predvsem podatkovnemu rudarjenju finančnih podatkov. Ti so predstavljeni kot niz podatkov, zbranih v nekem določenem času. Tako zaporedje podatkovnih točk v času imenujemo *časovna vrsta*. Primer časovnih vrst so na primer cene delnic, ki se zbirajo v točno določenih intervalih v nekem času, nihanje plime in celo podatki o vremenu. Obstaja zelo veliko primerov, kjer se podatki zbirajo skozi čas v obliki časovnih vrst. Analiza časovnih vrst se uporablja, kadar se spremenljivka opazuje z namenom, da bi iz nje razbrali statistične ali kakšne druge pomembne informacije. Analiza časovnih vrst pa se uporablja tudi za primerjanje različnih dogodkov v času in pa za napovedovanje vzorcev v prihodnosti, kar je za mojo diplomsko nalogo ključnega pomena (Zissis in drugi 2015).

Časovna vrsta se od preprostih statističnih podatkov loči v tem, da so vrednosti v časovnih vrstah zbrane v različnih časovnih točkah, ki so med seboj enakomerno oddaljene v času. Pri sami analizi moramo biti pozorni predvsem na to, da so točke v časovnih vrstah medsebojno povezane. Temu rečemo serijska korelacija. Take spremenljivke imajo po navadi ponavljajoče se vzorce skozi čas. Pri analizi s preprosto regresijo bi nam rezultati lahko prikazali navidezno korelacijo. Poleg tega pa se je treba zavedati, da v časovni vrsti obstaja trend gibanja, ki ga lahko uporabimo v svojo korist in z njim razlagamo »obnašanje« podatkovja, velikokrat pa se ga želimo tudi znebiti, saj bi lahko popačil rezultate in jih prikazal napačno.

Kot že omenjeno, podatkovno rudarjenje pomeni iskanje novih znanj in skritih vzorcev v podatkih, v našem primeru so ti predstavljeni v obliki časovnih vrst. Najpogostejša naloga podatkovnega rudarjenja je iskanje vzorcev, najpogostejša metoda za dosego tega cilja pa je razvrščanje v skupine. Za rudarjenje časovnih vrst pa se zelo pogosto uporabljajo tudi metode klasifikacije in asociacijska pravila (Fu 2011).

2.2 Strojno učenje

V diplomski nalogi za analizo uporabljam metodi, ki se med drugim uvrščata tudi med metode strojnega učenja. Za razumevanje metod in analize v diplomski nalogi bom v

naslednjih nekaj odstavkih predstavil osnovne koncepte strojnega učenja. Bistvo strojnega učenja je v tem, da računalniku (stroju) podamo učne podatke, na katerih se uči in nam z naučenim znanjem napoveduje nove iskane vrednosti v novih podatkih.

Da računalnik reši neki problem, potrebuje algoritem, ki predstavlja navodila za transformacijo vhodnih podatkov v neke izhodne podatke. Primer je denimo algoritem za razvrščanje. Vhodne podatke predstavlja naključen niz števil in izhod predstavlja urejen niz teh števil (denimo po velikosti). Za izvedbo te operacije obstajajo različni algoritmi, nas pa zanima predvsem to, da najdemo najbolj učinkovitega glede na število ukazov oziroma porabo pomnilnika, lahko pa tudi glede na oba atributa (Alpaydine 2010).

Za nekatere probleme pa algoritmov preprosto nimamo. Dober primer tega je razvrščanje vsiljene pošte. Vemo, da je vhod neka spletna pošta, izhod procesa razvrščanja pa bi moral biti »DA« ali »NE«, glede na to, ali je prispelo sporočilo vsiljena pošta ali ne. Manjkajoče znanje o tem, kako razvrščati tako pošto, lahko nadomestimo z obsežnimi podatki. V tem primeru želimo, da se računalnik (stroj) nauči iz danih primerov vsiljene in nevsiljene pošte ter sam definira algoritem, po katerem bo razvrščal prispelo pošto (Alpaydine 2010).

Strojno učenje je programiranje računalnika tako, da optimizira neki kriterij uspešnosti glede na učne primere oziroma pretekle izkušnje. Imamo model, v katerem smo definirali neke parametre. Učenje predstavlja izvedba računalniškega programa, ki bo optimiziral definirane parametre s pomočjo *učnih primerov*. Model je lahko napovedovalen, torej da se z njim napovedujejo vrednosti v prihodnosti, ali pa deskriptiven, s katerim pridobimo novo znanje iz podatkov. *Učni primeri* predstavljajo množico vzorcev področja raziskovanja. Sestavljeni so iz enot neodvisnih in odvisnih spremenljivk, iz katerih se stroj uči relacij med njimi (Alpaydine 2010).

Pri strojnem učenju združujemo matematiko, statistiko in računalništvo. Z uporabo statistike in matematike gradimo modele in metode strojnega učenja, ki jih z pomočjo programskih jezikov prevedemo v jezik, ki je razumljiv tudi stroju. Vloga računalniške znanosti je pri strojnem učenju dvojna. Prvič, za učenje potrebujemo učinkovite algoritme tako za reševanje optimizacijskih problemov kot tudi hranjenje in obdelovanje velike količine podatkov, s katerimi delamo. Ko je model naučen, mora poleg točnih napovedi poskrbeti tudi za rešitev, s katero bomo pojasnjevali značilnosti v podatkih in podali logične zaključke o danem problemu. V nekaterih primerih je pojasnjevalna vloga algoritma veliko bolj pomembna kot sama napoved (Alpaydine 2010).

Uporabljajo se tri oblike strojnega učenja. Pri nadzorovanem učenju se računalnik uči iz podatkov, ki so predstavljeni v obliki vnaprej podanih urejenih parov vhodnih in želenih izhodnih vrednosti. Rezultat učenja je naučen model, ki nam podaja nove izhodne vrednosti glede na nove, neznane vhodne podatke. Primer omenjene vrste učenja je tudi klasifikacija, kamor sodijo tudi umetne nevronske mreže. Druga oblika strojnega učenja je nenadzorovano učenje, kjer gre za algoritem, ki dobljene vhodne podatke sam razdeli glede na določene kriterije, po različnih kategorijah. Število kategorij algoritem določa sam, razvršča jih po skupnih lastnostih. Primer takega učenja je metoda razvrščanja v skupine. Poznamo še vzpodbujevalno učenje. Po zadnji metodi se algoritem uči s pomočjo nagrajevanja in kaznovanja odločitev. Operacija želi preslikati neko začetno stanje v dejanje. Taka vrsta učenja se uporablja predvsem pri kontroli robotov (dinamični sistemi) (Alpaydine 2010).

V mojem primeru lahko strojno učenje in podatkovno rudarjenje obravnavamo kot pojma, ki se nanašata na iste operacije, saj se uporaba strojnega učenja na obsežnih bazah podatkov šteje za podatkovno rudarjenje. Poleg implikacije strojnega učenja za namene podatkovnega rudarjenja pa lahko omenim, da ima ta velik pomen pri reševanju problemov z računalniškim vidom in robotiko ter problematiki prepoznavanja govorice (Alpaydine 2010).

Ko gradimo model, želimo, da se ta prilagaja tako podatkom na učnih primerih kot tudi tem v testni množici. Cilj strojnega učenja namreč ni repliciranje učnih podatkov, ampak iskanje novega znanja oziroma novih želenih vrednosti. Končni rezultat učenja je sposobnost *posploševanja* (angl. generalization), kar pomeni, da bo naučen algoritem določal oziroma razvrščal vrednosti tudi na primerih, ki mu v učni množici niso bili predstavljeni. Za najboljše možno posploševanje modela želimo, da je model čim bolj preprost. Preveč kompleksni modeli z velikim številom parametrov glede na velikost učne množice velikokrat pripeljejo do *prenasičenja klasifikatorjev* (angl. overfitting). To pomeni, da model v učenje zajame naključne napake in šum v podatkih namesto skritih relacij, ki bi nam dale moč napovedovanja. Tak model ima zelo slabo uspešnost pri napovedovanju novih vrednosti, saj se preveč prilega učnim podatkom (Alpaydine 2010).

3 Delnice

Delnica je listina, prenosni pravni dokument, ki dokazuje določeno obliko stvarnega razmerja, iz katerega izhajajo za imetnika delnice določene premoženjske ali članske pravice v določeni organizaciji (Svilan 1990).

Delniške družbe si z izdajo delnic zagotovijo trajna finančna sredstva, od katerih se ne plačujejo obresti in jih ni treba vrniti, saj delnica in njena vrednost ne zapadeta, dokler obstaja izdajatelj. Tu ne gre za zadolževanje, ampak zagotavljanje tujih finančnih sredstev (Svilan 1990).

Delnice se izdajo predvsem z namenom, da se zberejo potrebna sredstva za opravljanje kakšnih novih gospodarskih dejavnosti, ali pa za razširitev že obstoječe gospodarske dejavnosti. Kupec oziroma imetnik delnice postane investitor za tovrstno investicijo. S tem imetnik dobi določene premoženjske in članske pravice. Premoženjske pravice mu omogočajo trajno udeležbo pri dobičku podjetja, ta se izplača v obliki dividend, ter prednostne pravice pri nakupu novih delnic in likvidacijski delež ob morebitnem prenehanju delovanja družbe. Članske pravice pa imetniku omogočajo nadzorovati delovanje družbe in sodelovati pri določanju organov uprave (volitve). Delnice se štejejo tudi kot tržni finančni instrument predvsem zaradi njihovih namenov in lastnosti in ker imetniku omogočajo prenos vseh pravic na drugo osebo (Svilan 1990).

Delnice lahko izda le delniška družba, kupec oziroma investitor pa so lahko različni subjekti, odvisno predvsem od zakonske ureditve v posameznih državah. Osnovni kupci delnic so posamezniki, sledijo pa jim druga podjetja, ki lahko prek nakupa delnic postanejo lastnik drugega podjetja. Delnica kot finančni instrument postane poglavitni predmet trgovanja predvsem na organiziranem sekundarnem trgu oziroma borzi (Svilan 1990).

Poznamo več različnih oblik delnic. Svilan v svojem delu omenja navadne delnice, prednostne delnice ter posebne oblike delnic, med katere uvršča kumulativne in nekumulativne prednostne delnice, participativne in neparticipativne prednostne delnice, prednostne delnice z gibljivim donosom, prednostne delnice brez dividende, zamenljive prednostne delnice in primarno odkupljive prednostne delnice (Svilan 1990). V posamezno obliko delnice se ne bom poglabljal, saj njihovo poznavanje sami diplomski nalogi ne daje nobene dodatne vrednosti.

Samo investiranje v nakup delnic kupcu ne prinaša nikakršnega posebnega donosa razen dividend od delnic in nekaterih članskih pravic, ki so določene s statusom podjetja, ki je delnico izdalo, kot sem omenil v prejšnjih odstavkih. To pomeni, da so lahko take investicije zelo negotove. Za investitorja oziroma imetnika delnic pa dividende pravzaprav niso tako pomembne, saj je to le del donosa, ki mu ga prinaša delnica. Najpomembnejša je cena delnice oziroma njen tečaj. Če tečaj delnice raste, ima investitor možnost velikega tečajnega dobička. Tega pa doseže s takojšnjo prodajo delnic po novih, višjih cenah. Tako doseže želen donos glede na vložena sredstva (Svilan 1990).

Delnice se v praksi ovrednotijo predvsem na podlagi dveh dejavnikov. Prvi je kazalnik dosedanjega razvoja in donosa delniške družbe in bilanca ter sam uspeh podjetja, drugi element pa kaže na zaupanje v prihodnost družbe ter gospodarski sistem, v katerem podjetje deluje. Lahko rečemo, da gre za prospektivno določanje vrednosti cene delnic. To je lahko optimistično ali pa pesimistično, kar je odvisno predvsem od preteklih in prihodnjih kazalnikov delovanja delniške družbe. Torej se vrednost delnice ne določa samo na podlagi realnih gospodarskih kazalnikov, ampak tudi na podlagi špekulativnih pričakovanj, ki jih imamo o podjetju in gospodarskem okolju, v katerem delniška družba deluje (Svilan 1990).

3.1 Koncept delniškega trga

Posamezniki, ki se ukvarjajo z napovedovanjem gibanja delniških trgov, se osredotočajo predvsem na razvoj metod, ki bodo uspešno napovedovale indeksne vrednosti oziroma cene delnic. Pri tem ciljajo na velike dobičke, v pomoč pa so jim dobro zastavljene menjalne strategije. Osrednja ideja za uspešno napovedovanje gibanja trga delnic predvideva, da se z minimalnimi podatki dobijo najboljši rezultati in najmanj kompleksni modeli delniških trgov. Indeks delniškega trga predstavlja povprečno gibanje več posameznih delnic. Indeks predstavlja večinoma gibanje trga kot gibanje same delnice (Atsalakis in Valavanis 2009).

Finančni trgi so kompleksni, nelinearni in stalno razvijajoči se sistemi. Kompleksnost trgov predstavlja predvsem veliko število akterjev oziroma procesov na trgu, ki so v medsebojni interakciji, vendar pa njihovega medsebojnega delovanja ne moremo zapisati z matematično linearno enačbo. Gre za procese znotraj trgov, ki jih med seboj ne moremo preprosto združevati, niti ne moremo napovedati njihovega delovanja. Na trgu delujejo tako kupci kot prodajalci, posredniki oziroma agenti, ki posredno vplivajo na same cene delnic. Na drugem koncu pa imamo ekonomske premike na posameznih trgih, ki prav tako vplivajo na cene

delnic, vendar s samimi kupci in prodajalci delnic nimajo neke posebne povezave. Teorija kompleksnosti nekega sistema je zelo obsežna in v diplomski nalogi ne bi imela neke dodatne vrednosti, zato se vanjo ne bom poglabljal. Področje finančnega napovedovanja je zaznamovano z množico podatkov in veliko šuma. Iz množice podatkov je treba izvleči relevantne informacije, ki jih bomo uporabili za gradnjo modelov napovedovanja. Šum v podatkih predstavljata prav nenehno spreminjanje cen in pa nihanje volumna podatkov, ki ga je možno spremljati na finančnih trgih. Prav šum pa lahko največkrat zmede analitika, da napačno oceni smer gibanja trga. Na finance vpliva veliko dejavnikov, kot so politični dogodki, splošne gospodarske razmere in pričakovane menjave. Zato je napovedovanje gibanja finančnih trgov zelo oteženo (Huang in drugi 2004). Kljub temu pa je samo napovedovanje gibanja trga pomembno, saj nam da pomembne informacije za investicijske odločitve (Yuan 2011).

V splošnem obstajajo trije koncepti napovedovanja gibanja delniških trgov. Prvi verjame, da noben investitor ne more pridobiti nadpovprečnih prednosti v menjavi delnic z informacijami iz preteklosti in sedanjosti. Tu se zagovorniki omenjenega koncepta sklicujejo na »*hipotezo učinkovitega trga«*, ki pravi, da so trgi delnic preveč kompleksni, da bi se lahko napovedovali, in teorijo »*slučajnega sprehoda*«, ki pravi, da so vsi premiki na trgu naključni in se jih ne da napovedovati (Peters v Yuan 2011). Drugi izmed treh konceptov je *fundamentalna analiza*. Ta spodbuja raziskovalca, da se loti poglobljene študije različnih makroekonomskih kazalnikov in finančnih odnosov ter posledice industrije na gibanje trgov. S tem razkrije korelacije, ki bi lahko obstajale med omenjenimi parametri in gibanjem trga (Majhi in Panda 2007). Tretji koncept verjame, da obstajajo modeli, ki jih lahko razvijemo in z njimi napovedujemo gibanje trga delnic, to imenujemo tudi *tehnična analiza.* Analitika zanimajo predvsem cene in gibanje trgov, na podlagi katerih razvija modele napovedovanja (Yuan 2011).

4 Teoretični vidiki nevronskih mrež

Metoda nevronskih mrež se je razvila na podlagi ideje, da računalnik pri procesiranju podatkov imitira delovanje človeških možganov. Človeški možgani obdelujejo podatke na precej drugačen način, kot jih obdelujejo sodobni računalniki, saj so ti zelo kompleksen, nelinearen in paralelni sistem za procesiranje informacij. Sposobni so prepoznavanja vzorcev in izvajanja motoričnih sposobnosti, in to veliko hitreje kot sodobni računalnik. Živčni sistem pri človeku se uči skozi izkušnje, se prilagaja okolju, ki človeka obdaja, podobno je z nevronskimi mrežami, ki so sestavljene iz umetnih nevronov (Haykin 2009). Umetne nevronske mreže so zelo podobne biološkim mrežam, odlikujejo jih visoka stopnja vzporednosti, asinhrono izvajanje, večsmerno izvajanje, prilagodljivost, robustnost na okvare in na manjkajoče podatke, sposobnost učenja in avtomatska generalizacija. Proučevano področje nevronskih mrež je zelo dobro podprto z matematično podlago, glavna pomanjkljivost pa je težavnost razlage njihove odločitve. Interpretacija rezultatov nevronskih mrež s stališča raziskovalca ni mogoča. Med vhodnimi in izhodnimi vrednostmi lahko ustvarimo preslikavo, vendar pa nimamo informacije, zakaj se je določen x preslikal v določen y. Zato rezultatov nevronske mreže ne moremo razložiti (Kononenko in Robnik Šikonja 2010).

Kot že omenjeno, sta razvoj nevronskih mrež in reševanje problemov z njimi povezana s poskusom oponašanja možganov ter učinkovitim razpletanjem kompleksnih in zahtevnih problemov. Nevronske mreže so abstrakcija in poenostavitev možganskih celic. Možganski nevroni se v različnih delih možganov med seboj razlikujejo po obliki, kljub temu pa njihova osnova ostaja ista. Kljub poenostavljenosti umetnih nevronskih mrež v primerjavi s človeškimi možgani so te zadosti učinkovite za modeliranje in analizo (Kononenko in Robnik Šikonja 2010).

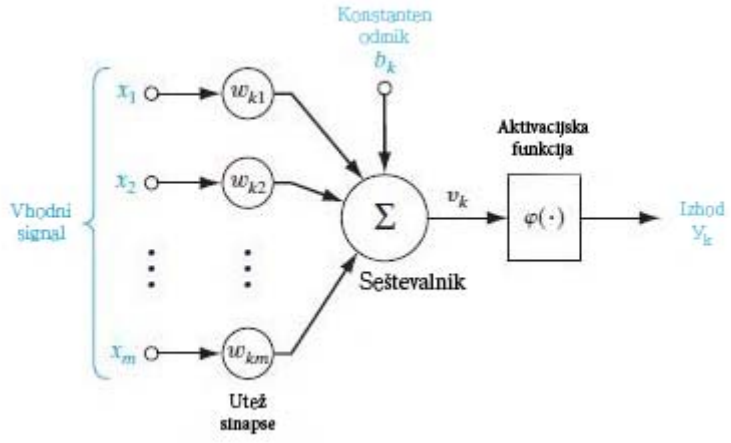
Nevronske mreže odlikuje tudi večsmerno izvajanje. V mreži, kjer je vsak nevron povezan z vsakim, so vsi nevroni hkrati vhodni in izhodni. Sama mreža ne dela razlike med vhodom in izhodom. Mreža deluje tako, da bo pri danem vhodu ocenila vrednosti, ki bodo v tem primeru postale izhod. Poleg tega lahko rečemo, da so nevronske mreže zelo robusten sistem, saj so robustne na okvare posameznih nevronov in sinaps. Večje ko je število okvarjenih nevronov in povezav, manjša bo točnost modela. Prav tako je mreža robustna tudi na pomanjkljive vhodne podatke, saj manjkajoče vrednosti nadomesti z ocenami preostalih vhodnih podatkov in že naučenega. Ta lastnost prihaja iz načina delovanja samih nevronskih mrež. V mreži,

kjer je vsak nevron povezan z vsakim, se bo manjkajoč podatek nadomestil že v prvi iteraciji (Kononenko in Robnik Šikonja 2010).

4.1 Nevron

Nevron je osnovni blok nevronske mreže, gre za enoto znotraj mreže, ki procesira vhodne informacije. Ima vsaj tri osnovne komponente. Pri razlagi delovanja si bom pomagal s spodnjo sliko, ki predstavlja nevron z več vhodi (Haykin 2009).

Slika 4.1: Sestava nevrona.



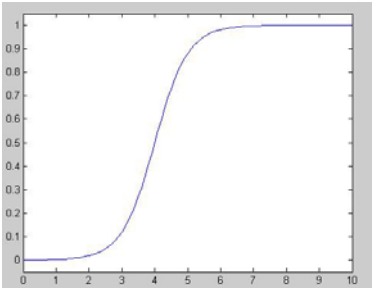
Vir: Haykin (2009).

Prva komponenta je sinapsa oziroma povezava z drugimi nevroni. Vsaka sinapsa ima svojo težo oziroma moč. To pomeni, da signal x_j na vhodu sinapse *j, ki je povezana z nevronom k,* pomnožimo z utežjo sinapse w_{kj}. Utež sinapse v umetnih nevronih lahko zavzema tako pozitivne kot negativne vrednosti (Haykin 2009).

Druga komponenta je seštevalnik, ki sešteva vhodne signale, utežene s sinaptično močjo nevrona. Imenujemo ga tudi *linearni kombinator*. Njegov izhod predstavlja aktivacijo nevrona oziroma nivo aktivacije nevrona (Haykin 2009). Tretja komponenta je aktivacijska oziroma transformacijska funkcija, ki združi vhode v vmesno vrednost. Izhodna funkcija pa preslika rezultate aktivacije v izhod nevrona (Kononenko in Robnik Šikonja 2010). Poznamo več vrst aktivacijskih funkcij. V diplomski nalogi se bom v eksperimentalnem delu opiral predvsem na sigmoidno funkcijo.

Sigmoidna funkcija je oblike: $f(X) = \frac{1}{1+e^{-X}}$

Slika 4.2: Graf sigmoidne funkcije.



Sigmoidna funkcija ima graf v obliki črke S in je med vsemi aktivacijskimi funkcijami najpogosteje uporabljena pri grajenju nevronskih mrež. Funkcijo sem že definiral v prejšnjih odstavkih. Vhodne vrednosti funkcije so lahko poljubne. Funkcija lahko zavzema vrednosti med 0 in 1. To pomeni, da je vsaka vhodna vrednost na izhodu predstavljena z neko svojo lastno vrednostjo. Torej ima vsak vhod točno določeno vrednost na izhodu. Spremenljivka *X* definira naklon funkcije, s spreminjanjem tega dobimo grafe sigmoidne funkcije z različnimi nakloni (Haykin 2009).

Slika 4.1, ki predstavlja nevronski model, vsebuje tudi konstanten odmik b_k. Ta vpliva na večanje ali manjšanje vrednosti aktivacijske funkcije, glede na to, ali je vrednost odmika pozitivna ali negativna. Kot matematični izraz bi lahko opisan nevron *k* zapisali kot:

$$u_k = \sum_{j=1}^{m} w_{kj} * x_j \text{ in } y_k = \varphi(u_k + b_k)$$

kjer so $x_1, x_2, \ldots, x_m$ vhodni signali, $w_{k1}, w_{k2}, \ldots, w_{km}$ uteži sinapse za nevron *k*, b_k je konstanten odmik in $\varphi(.)$ je aktivacijska funkcija ter y_k, ki predstavlja izhodni signal opisanega nevrona (Haykin 2009).

4.2 Topologije nevronske mreže

Breznivojska nevronska mreža je najbolj splošna oblika nevronskih mrež. Vsak nevron je hkrati vhoden in izhoden ter je dvosmerno povezan z vsemi preostalimi nevroni. Omenjena mreža deluje tako, da se nevronom v samem začetku vsili vrednost vhodnega vzorca, zatem pa nevroni spreminjajo svoje izhode in svoje stanje toliko časa, da se izhod mreže ustali (Kononenko in Robnik Šikonja 2010).

Enonivojska usmerjena nevronska mreža je sestavljena iz organiziranih nivojev nevronov. V taki mreži imamo vhodni nivo, kjer je vsak nevron enosmerno povezan na vsak izhodni nevron. Taki mreži se reče enonivojska, ker se vhodni nivo ne šteje kot posamičen nivo, saj se v njem nič ne izračuna (Haykin 2009). Izračuni v taki mreži potekajo tako, da na vhodnih

nevronih dobimo vrednost komponent vhodnega vzorca in zatem v enakem koraku vsi izhodni nevroni vzporedno izračunajo izhodne vrednosti. Primer take mreže je perceptron. Take mreže lahko rešujejo le linearne probleme (Kononenko in Robnik Šikonja 2010).

Večnivojske usmerjene nevronske mreže se od enonivojske nevronske mreže razlikujejo po tem, da med vhodne in izhodne nivoje dodamo še enega ali več skritih nivojev. S tem dobimo večnivojsko nevronsko mrežo. Tem nivojem rečemo skriti, ker se ta del nevronskih mrež ne more razbrati niti iz vhoda niti iz izhoda. Večnivojska mreža deluje podobno kot enonivojska, spremeni se le število korakov pri izračunu. To je enako številu skritih nivojev plus ena. Z omenjenimi mrežami lahko računamo tudi nelinearne probleme (Kononenko in Robnik Šikonja 2010).

Tretji primer topologije nevronskih mrež je dvosmerni asociativen pomnilnik. Gre za mrežo, sestavljeno iz dveh nivojev nevronov, ki se povezujeta z dvosmernimi vezmi. Izračun v taki mreži poteka po naslednjih korakih: nevronom prvega nivoja vsilimo vhodne vrednosti, nevroni drugega nivoja za tem izračunajo svoje izhode. Potem spet nevroni prvega nivoja izračunajo svoje izhode. To se ponavlja, dokler se izhod obeh nivojev ne ustali. Dvosmerni asociativni pomnilnik lahko deluje sinhrono ali asinhrono (Kononenko in Robnik Šikonja 2010).

4.3 Perceptron

4.3.1 Enonivojski perceptron

Enonivojski usmerjeni nevronski mreži pravimo tudi enonivojski perceptron. Mrežo sestavljajo vhodi $X_1, \ldots, X_n$ in izhodi oziroma izhodni nevroni $Y_1, \ldots, Y_M$. V sami mreži izračun poteka tako, da na vhode naslovimo vhodni vzorec $X = (x_1, \ldots, x_n, 1)$. Zadnja komponenta vzorca predstavlja prag θ izhodnega nevrona. Vsi izhodni nevroni v enem koraku, vzporedno in neodvisno drug od drugega, izračunajo svoje izhodne vrednosti:

$$Y = WX$$

Kjer je W matrika uteži. Torej vsak nevron izračunava funkcijo:

$$Y_i = \sum_{j=1}^{N} W_{ji} X_j + \theta_i$$

W_{ji} predstavlja utež med *j*-tim vhodom in *i*-tim izhodnim nevronom in θ_i je prag *i*-tega izhodnega nevrona (Kononenko in Robnik Šikonja 2010).

Vsak nevron računa svoj izhod neodvisno od preostalih nevronov in se lahko zaradi te lastnosti pri obravnavanju dvonivojske mreže omejimo na samo en izhodni nevron. Torej je lahko W vektor, in ne matrika uteži. Izhodni nevron računamo po naslednjem pravilu:

$$Y = W^T X$$

Če je vrednost nevrona $Y \geq 0$, se primer klasificira v prvi razred, če je $Y < 0$, ga nevronska mreža klasificira v drugi razred. Pri učenju enonivojske nevronske mreže je potrebna ustrezna nastavitev vektorja uteži *W* tako, da mreža pravilno razvrsti oziroma klasificira učne primere. Učenje mreže poteka po naslednjih korakih: že pred samim začetkom učenja mreže so uteži naključno izbrane, kar pomeni, da je vrednost vseh uteži lahko enaka nič. Učenje se izvaja na poznanih parih vhodnih in izhodnih vzorcev. Če mreža pravilno klasificira *n*-ti primer, se vrednosti uteži ne spremenijo, če primer napačno klasificira, se uteži spreminjajo glede na podana pravila učenja. Z iterativnim prikazovanjem vzorcev dosežemo, da se mreža nauči pravilno »odgovarjati« na vse vhode (Kononenko in Robnik Šikonja 2010).

Omenjena mreža se lahko uči tudi po principu *srednje kvadratne napake* (angl. least-mean-square), ker v limiti in pri zadosti majhni *stopnji učenja*, ki podaja hitrost algoritma, konvergira v optimalno mrežo po kriteriju srednje kvadratne napake. Do preskoka optimalne rešitve v tem primeru lahko pripelje prevelika stopnja učenja same mreže. Učenje v tem primeru niha okoli optimuma (Kononenko in Robnik Šikonja 2010).

Poznamo pa še paketno pravilo delta. Za nevronsko mrežo pri nekem danem vektorju uteži *W(n)* izračunamo razliko med želenim in dejanskim izhodom za vse učne primere *X(1),...X(m)*. Po izračunu pa se uteži spreminjajo v smeri negativnega odvoda. Paketno pravilo upošteva več informacij naenkrat in zato po navadi hitreje konvergira. Omenjen algoritem izračuna napako na vseh učnih primerih, zato se ne implementira v same nevronske mreže (Kononenko in Robnik Šikonja 2010).

4.3.2 Večnivojski perceptron

Izraz večnivojski perceptron se nanaša na večnivojske nevronske mreže, katerih delovanje sem opisal na prejšnji strani. Za večnivojske mreže je bilo razvito posplošeno pravilo delta, ki mu pravimo tudi *pravilo vzvratnega razširjanja napake* (angl. backpropagation of error). To

omogoča učenje nevronske mreže, ki ima v svoji topologiji poljubno število nivojev. Posplošeno pravilo delta ima osnoven princip delovanja enak kot paketno pravilo delta, opisano v prejšnjem poglavju. Zaradi računanja odvoda napake, tudi pri skritih nivojih, se pojavi potreba po tem, da je izhodna funkcija zvezna in zvezno odvedljiva. V ta namen se uporablja sigmoidna funkcija (Kononenko in Robnik Šikonja 2010).

Večnivojsko mrežo bom uporabljal tudi sam pri eksperimentalnem delu diplomske naloge, prav tako bom v skritih nevronih definiral sigmoidno funkcijo.

4.4 Učenje nevronske mreže s pravilom vzvratnega postopka razširjanja

V eksperimentalnem delu diplomske naloge bom primerjal metodo podpornih vektorjev in nevronske mreže na primeru napovedovanja gibanja delniškega trga. Kot že omenjeno, me zanima predvsem to, katera izmed učnih metod bo bolj točno opravila svoje delo. Za primerjavo z metodo podpornih vektorjev sem izbral standardno trinivojsko nevronsko mrežo s *pravilom vzvratnega postopka razširjanja* (angl. Backpropagation neural networks).

Omenjeni tip nevronske mreže je sestavljen iz vhodov in procesiranih enot, ki so nam, kot že omenjeno, znani kot nevroni. Nevroni se med seboj tudi v tem primeru povezujejo s sinapsami in njihovimi utežmi. Zadnje poleg topologije nevronske mreže hranijo znanje o naučeni nevronski mreži. Poleg procesiranih nevronov so v skritih plasteh nevronske mreže definirani še nevroni z vrednostjo konstantnega odmika. Ti so povezani z vsemi nevroni, tako v skritih plasteh kot v izhodnih nivojih. Število skritih nivojev in nevronov v vsakem nivoju mreže se razlikuje glede na samo velikost in naravo podatkov, s katerimi delamo (Kaastra in Boyd 1996).

Število vhodnih enot mora biti enako številu neodvisnih spremenljivk, število izhodnih nevronov je enako številu odvisnih spremenljivk. Nevronske mreže s pravilom vzvratnega postopka razširjanja so kategorija usmerjenih nevronskih mrež s pravili nadzorovanega učenja. Pri tem je pomembno vedeti, da se beseda »usmerjeno« nanaša samo na smer toka podatkov od vhodov proti izhodom (Kaastra in Boyd 1996). Bistvo omenjenih nevronskih mrež je, da se mreža uči skupaj z optimizacijskimi metodami, kot je recimo gradientna funkcija. Metoda računa gradientno funkcijo napake glede na uteži v mreži. Dobljen gradient se pošlje nazaj po mreži do optimizacijske funkcije, ki glede na dobljeno vrednost priredi uteži, tako da se funkcija napake zmanjša, kolikor se le da (Haykin 2009).

5 Teoretični vidiki metode podpornih vektorjev

5.1 Teorija SVM

Metoda podpornih vektorjev (angl. Support Vector Machines; v nadaljevanju SVM) je uporabna tehnika za klasifikacijo podatkov, regresijo in napovedovanje. SVM je učna metoda, ki analizira podatke in prepoznava vzorce v tem podatkovju. Trenutne standardne algoritme sta predstavila Cortes in Vapnik leta 1995. Metoda se izogne večini problemov, s katerimi se srečujejo tradicionalni ekonometrični modeli. Ti modeli so delovali na velikem številu mikro- in makroekonomskih kazalnikov, ki so že sami po sebi zelo nestabilni, kar je posledično vplivalo na robustnost modela. Največja težava ekonometričnih modelov je bila, da so napovedi, ki so jih podajali na podlagi izvenvzorčnih podatkov, zelo netočne in negotove. Modeli so sicer do neke meje delovali, če je bila v analizo vključena tudi učna množica podatkov, pri povsem novih prej neznanih podatkih pa je bila napoved zelo slaba. Predhodne študije na področju napovedovanja delniških trgov so za analizo največkrat uporabljale nevronske mreže (artifical neural networks; ANN), metodo najbližjih sosedov in odločitvena drevesa. Zaradi ogromnega šuma in kompleksnih dimenzij, ki jih najdemo na delniškem trgu, ima metoda ANN svoje omejitve. Nevronske mreže so namreč zelo občutljive za parametre, ki jih nastavimo. Aktivacijska funkcija, skriti nivoji, uteži, stopnja učenja so le nekateri parametri, ki jih je treba optimizirati, da bi nevronske mreže podale najboljši rezultat. SVM nam omogoča, da prebrodimo večino teh problemov, s katerimi se srečamo pri aplikaciji nevronskih mrež (Yuan 2011).

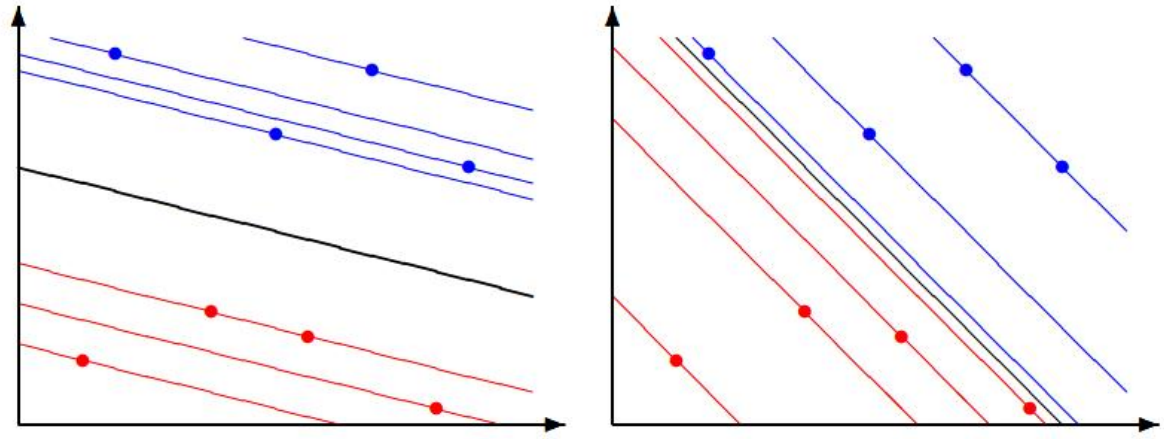
Metode podpornih vektorjev so bile prvič predstavljene v devetdesetih letih in veljajo za eno izmed najuspešnejših metod za klasifikacijo in regresijo. Večina algoritmov strojnega učenja stremi po tem, da jim podamo minimalno število atributov, nato pa poiščejo ustrezno podmnožico pomembnih atributov, nad katerimi zgradijo model. Za metodo SVM uporabimo vse razpoložljive atribute, tudi tiste manj pomembne, in jih z linearno kombinacijo uporabimo za napovedovanje odvisne spremenljivke. Pri omenjeni metodi je zato toliko bolj pomembno samo kombiniranje atributov kot pa izbira, saj bo sama metoda z ustrezno kombinacijo izluščila želeno informacijo (Kononenko in Robnik Šikonja 2010).

Metoda SVM je torej primerna za učenje na obsežnem podatkovju, ki ima lahko več pomembnih ali malo manj pomembnih spremenljivk. Slaba stran omenjene metode pa je, da je interpretacija naučenega precej težavna, prav tako je težavna razlaga odločitev same metode (Kononenko in Robnik Šikonja 2010).

5.2 Delovanje SVM

Ideja, ki stoji za metodo SVM, je, da v danem prostoru spremenljivk poiščemo *optimalno hiperravnino, ki je enako in najbolj oddaljena od najbližjih primerov iz obeh razredov*. Hiperravnina je (*m*-1)-razsežen prostor v *m*-razsežnem prostoru, ki ta prostor razdeli tako, da pozitivne primerke loči od negativnih. Če so primeri linearno ločljivi, obstaja več hiperravnin, ki ločujejo dva razreda med seboj. Primerkoma, ki sta najbližje hiperravnini, pravimo podporna vektorja. Razdaljo od podpornega vektorja in hiperravnino pa imenujemo *rob* (angl. margin). Za nas je torej najbolj optimalna tista hiperravnina, ki ima maksimalni rob (Kononenko in Robnik Šikonja, 2010).

Slika 5.1: Dve optimalni hiperravnini glede na klasifikacijski problem.



Metode podpornih vektorjev učenje izvedejo kot reševanje optimizacijskega problema, zato bom, preden bom predstavil algoritem metode SVM, v tem odstavku na kratko predstavil bistvo optimizacije. Gre za proces, pri katerem želimo izbrati najboljši element izmed skupine alternativ, ki so nam na voljo. Pri najpreprostejši obliki optimizacije gre za iskanje minimuma ali maksimuma realne funkcije v dovoljeni množici vektorjev (Boyd 2004).

Optimizacijski problem lahko zapišemo kot minimizacijo funkcije: $f_0(x)$

Pri pogojih $f_i(x) \leq b_i,\ i = 1,\dots,n$

Vektor $x = (x_1,\dots,x_n)$ je spremenljivka optimizacijskega problema, funkcija $f_0 : R^n \to R$ je *namenska funkcija*, torej je tista funkcija, ki jo bomo optimizirali (angl. objective function), $f_i : R^n \to R, i = 1,\dots,n$ predstavlja predpisan pogoj namenske funkcije (angl. constraint function) in konstanta $b_1,\dots,b_m$ predstavlja omejitev te funkcije. Optimalen vektor predstavlja rešitev optimizacijskega problema, če ima najmanjšo možno vrednost med vsemi podanimi

vektorji, ki ustrezajo danim pogojem. Pri maksimizaciji funkcije gre za enak postopek, vendar v tem primeru iščemo vektor z največjo možno vrednostjo med vsemi vektorji glede na dane pogoje (Boyd 2004).

5.3 Algoritem metode podpornih vektorjev

V tem delu naloge bo predstavljena metoda SVM za klasifikacijo v dva razreda. Če bi želeli imeti več razredov, postopek ponovimo za vsak razred, ki ga želimo ločiti od preostalih. V ozadju metode SVM so kompleksne matematične izpeljave. V diplomski nalogi bom izpostavil samo najpomembnejše formule.

Naj bo kot vhod algoritma podana množica urejenih parov vhodnih in izhodnih vrednosti, ki jo imenujemo učna množica:

$$(u_j, y_j), j - 1..n$$

kjer je $y_j = 1$, če primer pripada prvemu razredu, in $y_j = -1$, če pripada drugemu razredu. Z n označimo število učnih primerov, z a pa število atributov oziroma spremenljivk. Vsi atributi morajo biti zvezni, torej $u_j \in R^a, j = 1..n$. Enačba hiperravnine pa je podana z:

$$(w \cdot u) - b = 0$$

Primeri, ki pripadajo prvemu razredu, imajo pozitivno vrednost, primerki, ki pripadajo drugemu razredu, pa imajo negativno vrednost:

$$y_{j+} \rightarrow w \cdot u - b \geq +1$$

$$y_{j-} \rightarrow w \cdot u - b \leq -1$$

Te pogoje lahko združimo v omejitve, ki jih mora zadovoljiti optimalna hiperravnina in ki zahtevajo pravilno klasifikacijo vsakega učnega primera:

$$y_j[(w \cdot u) - b] \geq 1, j = 1..n$$

ter hkrati minimizirati

$$\frac{1}{2}(w \cdot w)$$

Zgornja enačba predstavlja pol kvadrata norme. Zaradi lažje minimizacije formulo zapisujemo v zgornji obliki. Ker želimo poiskati najširši rob, moramo poiskati w z najmanjšo normo. Torej lahko iskanje ravnine zastavimo kot optimizacijski problem. Zgornji problem minimizacije z omejitvami $y_i[(w \cdot u) - b] \geq 1, j = 1..n$ se prevede na problem maksimizacije funkcionala, kjer gre za kvadratni problem:

$$W(\alpha) = \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j (u_i \cdot u_j)$$

pri omejitvah

$$\alpha_j \geq 0, j = 1..n,$$

kjer α_j predstavlja Lagrangove multiplikatorje (več o tem si lahko preberete v petem poglavju knjige Cristianini, Nello in John Shawe-Taylor. 2000. *Suport Vector Machines and other kernel-based learning methods*. Cambridge. Cambridge Univesity Press) in

$$\sum_{j=1}^{n} \alpha_j y_j = 0$$

Vsak koeficient α_j ustreza enemu učnemu primeru. Večina koeficientov je enaka 0, tisti, ki niso, pa določajo podporne vektorje. Naj bo vektor $\alpha_0 = (\alpha_1^0, \ldots, \alpha_n^0)$ rešitev tega kvadratnega optimizacijskega problema. Klasifikacijsko pravilo na osnovi optimalne hiperravnine je podano z:

$$Y(u) = sign(w_0 \cdot u - b_0) = sign \left(\sum_{podporni\ vektor\ u_j} y_j \alpha_j^0 (u_j \cdot u) - b_0 \right)$$

in je prag b_0 podan z:

$$b_0 = \frac{1}{2} [(w_0 \cdot u_*(1)) + (w_0 \cdot u_*(-1))]$$

kjer smo z $u_*(1)$ označili poljubni podporni vektor iz prvega razreda in z $u_*(-1)$ poljubni podporni vektor iz drugega razreda. »Sign« predstavlja signum funkcijo, ki nam poda predznak realnega števila. Vektor koeficientov optimalne hiperravnine w_0 je podan s podpornimi vektorji:

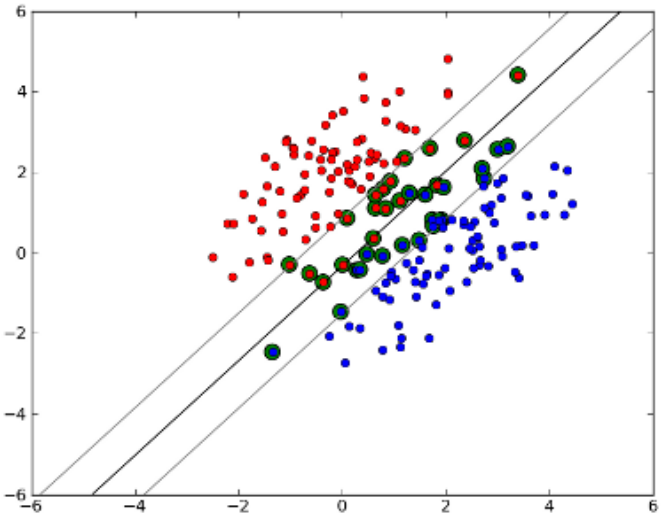
$$w_0 = \sum_{\text{podporni vektorji } u_j} y_j \alpha_j^0 u_j$$

Kompleksnost rešitve optimalne hiperravnine je podana s številom podpornih vektorjev. Ker je v primerjavi z vsemi učnimi primeri število podpornih vektorjev relativno majhno, je tudi kompleksnost rešitve majhna. Tudi če ostranimo vse vektorje iz učne množice primerov in pustimo samo podporne vektorje, dobimo povsem enako rešitev, kot če vseh preostalih vektorjev ne bi odstranili (Kononenko in Robnik Šikonja 2010).

5.4 Razširitev na neseparabilno klasifikacijo

Podatki pa niso vedno linearno ločljivi. To pomeni, da med dvema skupinama točk v nekem prostoru ne moremo zarisati linije, ki bi ločevala skupini točk med seboj. V takem primeru algoritem v zgornjem poglavju ne bo deloval in ne moremo poiskati hiperravnine, ki bi podatke točno ločila v razrede. Imamo problem neseparabilne klasifikacije. Problema se lotimo tako, da poskusimo poiskati tako hiperravnino, ki bo imela najmanjšo možno napako (Alpaydine 2010).

Slika 5.2 : Neseparabilna klasifikacija.



Vir: Stackowerflow.com (2016).

Pri neseparabilni klasifikaciji omejitve $y_j[(w \cdot u) - b] \geq 1, j = 1..n$, ki zahteva klasifikacijo vsakega učnega primera, spremenimo tako, da dovolimo napako pri klasifikaciji j-tega učnega primera za ε_j, ki je uporabniško določena:

$$y_j[(w \cdot u) - b] \geq 1 - \varepsilon_j, j = 1..n$$

Optimizacijski problem se spremeni v minimizacijo:

$$\frac{1}{2}(w \cdot w) + C\sum_{j=1}^{n} s_j$$

pri čemer je C vnaprej podana vrednost. S parametrom C uravnavamo razmerje med kompleksnostjo rešitve (prvi sumand) in napako rešitve (drugi sumand). Problem se prevede na maksimizacijo funkcionala, kot pri eksaktnem primeru, le omejitev $\alpha_j \geq 0, j = 1..n$ se spremeni v:

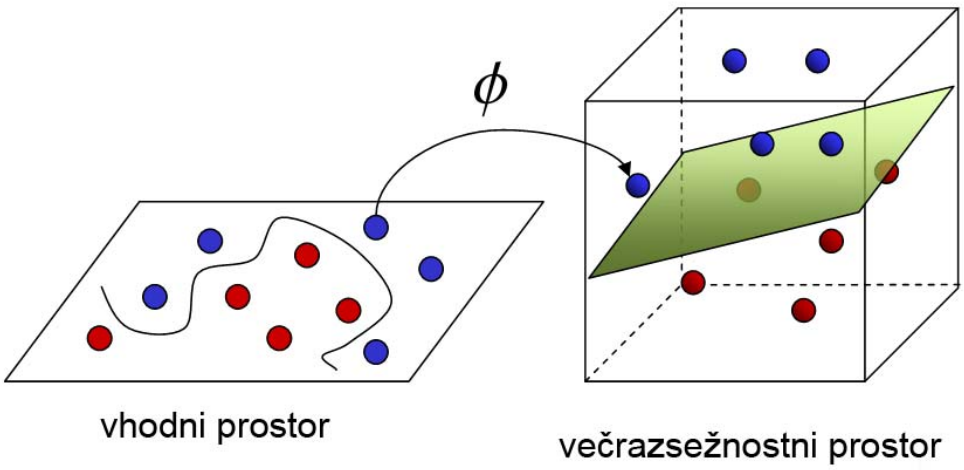
$$C \geq \alpha_j \geq 0, j = 1..n$$

medtem ko omejitev $\sum_{j=1}^{n} \alpha_j y_j = 0$ ostaja nespremenjena. Tudi pri neseparabilni rešitvi velja, da so samo koeficienti α_j, ki določajo podporne vektorje, različni od nič (Kononenko in Robnik Šikonja 2010).

5.5 Nelinearna posplošitev SVM

V prejšnjem poglavju sem prikazal, kako se z metodo SVM lotimo klasifikacije primerov, ki niso linearno ločljivi. Kaj pa naredimo, ko primerov nikakor ne moremo ločiti linearno, niti če dopustimo možnost napake? Metoda SVM nam omogoča, da se tega problema lotimo z uporabo jedrnih funkcij. Naš problem lahko preslikamo v večdimenzionalni prostor in v njem uporabimo linearno klasifikacijo.

Slika 5.3: Preslikava v večdimenzionalni prostor, ko primeri niso linearno ločljivi.



Vir: Stackoverflow.com (2016).

Osnovna ideja metode podpornih vektorjev je poleg uporabe podpornih vektorjev uporaba implicitne transformacije atributnega prostora v kompleksnejši atributni prostor. V originalnem prostoru pogosto linearna hiperravnina ne zadošča za sprejemljivo klasifikacijsko točnost. V tem primeru nam pomaga nelinearna transformacija prostora. S to metodo lahko prostor priredimo tako, da ga ločimo z linearno hiperravnino. Različne transformacije pa nam tudi omogočajo, da lahko tako rešujemo različne nelinearne probleme (Kononenko in Robnik Šikonja 2010).

Moč metode SVM pa se pokaže prav pri transformaciji prostora, saj ni treba, da to naredimo eksplicitno, kot je to treba pri preostalih metodah strojnega učenja. Z eksplicitno transformacijo dobimo veliko število atributov. Problemu prevelikega števila atributov pravimo tudi *prekletstvo dimenzionalnosti* (Kononenko in Robnik Šikonja 2010).

Pri metodah podpornih vektorjev zadošča, da izračunamo skalarne produkte podpornih vektorjev z novim primerom v transformiranem atributnem prostoru. Če je u_j *j*-ti vektor v originalnem prostoru in z_j ta isti primer v transformiranem atributnem prostoru, potem moramo zagotoviti izračun skalarnega produkta s pomočjo jedrne funkcije *K,* ki omogoča implicitno transformacijo:

$$z_j \cdot z = K(u, u_j)$$

Namesto skalarnih produktov uporabljamo konvolucijo skalarnih produktov. Konvolucija se nanaša na združevanje oziroma množenje dveh funkcij. Torej iz dveh funkcij dobimo eno novo funkcijo. Funkcional za maksimizacijo se torej prevede v:

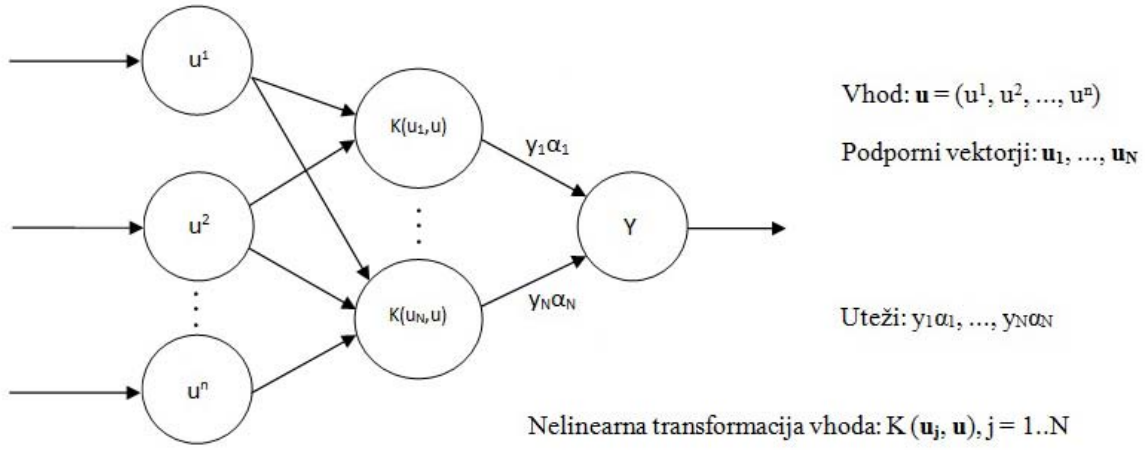
$$W(\alpha) = \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j K(u_i, u_j)$$

pri enakih omejitvah za koeficient α_j kot prej. Odločitvena funkcija za klasifikacijo novih primerov **u** je:

$$Y(u) = sign\Big(\sum_{podporni\ vektorji\ u_j} y_j \alpha_j^0 K(u_j, u) - b_0 \Big)$$

za dane α_j^0, ki predstavljajo rešitev optimizacijskega problema. Spodnja slika ponazarja odločitveno pravilo metode SVM.

Slika 5.4: Odločitveno pravilo metode SVM.



Vir: Kononenko in Robnik Šikonja (2010)

Z različnimi jedrnimi funkcijami dobimo različne transformacije prostora atributov in s tem različne variante metode SVM. Primeri jedrnih funkcij so: **linearna, polinomska, radialna, sigmoidna.**

Če se odločimo za uporabo sigmoidne funkcije, dobimo arhitekturo trinivojskih nevronskih mrež. Število skritih nevronov je enako številu podpornih vektorjev, saj metoda SVM sama določa optimalno arhitekturo. Uteži prvega nivoja ustrezajo vrednostim atributov podpornih vektorjev, uteži drugega nivoja pa izračunom koeficienta α_j^0 (Kononenko in Robnik Šikonja 2010).

V praksi se izkaže, da različne jedrne funkcije pri izračunu optimalne hiperravnine v dani problemski domeni izberejo v veliki meri (okoli 80 %) iste podporne vektorje (Vapnik 1998).

5.6 Regresijski SVM

Metoda podpornih vektorjev se prav tako aplicira tudi na reševanje regresijskih problemov. Pri tem obdrži vse poglavitne lastnosti osnovnega algoritma SVM, ki pripomorejo k iskanju maksimalnega roba. Torej algoritem želi poiskati nelinearno funkcijo s pomočjo linearnih funkcij v večdimenzionalnem prostoru. Reševanje regresijskega problema se nadzira s parametrom, ki je neodvisen od dimenzij prostora, v katerem iščemo omenjeno nelinearno funkcijo. Kot pri primeru klasifikacije želi učni algoritem poiskati in optimizirati pogoje, v tem primeru definirane za reševanje regresije. Več o danih pogojih si lahko preberete v knjigi Cristianini, Nello in John Shawe-Taylor. 2000. *Suport Vector Machines and other kernel-*

based learning methods. Cambridge. Cambridge Univesity Press (Cristianini in Shawe-Taylor 2000).

Problem regresije je poiskati funkcijo, ki bo v učni množici podatkov uspešno preračunala preslikavo vhodnih vrednosti modela v realne izhodne vrednosti. Ker izhodne vrednosti niso več binarna števila, to pomeni, da razlika med hipotetično vrednostjo in vrednostjo, ki jo bo podal model, ne bo več diskretna. Razliko med omenjenima vrednostma imenujemo *rezidual* izhoda. To je indikator natančnosti in prilagajanja modela. Pri regresijskem SVM je treba določiti, kako bomo merili to razliko, saj so majhni reziduali neizogibni, velikih pa se želimo čim bolj izogniti. V ta namen uvedemo *funkcijo izgube* (angl. loss function) (Cristianini in Shawe-Taylor 2000).

Funkcija napake zanemari majhne odklone (manj kot ε) od prave vrednosti. Tak tip funkcije je definiran kot ε neobčutljiva funkcija napake. Z uporabo te funkcije je verjetnost, da narobe klasificiramo naključni testni primerek, reda velikosti ε (Cristianini in Shawe-Taylor 2000).

Pri metodi SVM za reševanje regresijskih problemov lahko izpostavimo tri bistvene značilnosti. Omenjena metoda regresijo računa s pomočjo linearnih funkcij, v večdimenzionalnem, večatributnem prostoru. Metoda SVM predvideva vrednost regresije s pomočjo minimizacije tveganja, kjer se tveganje meri z Vapnikovo ε neobčutljivo funkcijo napake . Zadnja značilnost regresijskega SVM se nanaša na uporabo *funkcije tveganja* (angl. risk function), ki se opira na uporabo *empirične napake* (angl. empirical error) in *regulacijsko konstanto* (angl. regularization term) (Tay in Cao 2001).

Tudi tu imamo podano učno množico urejenih parov $G = (x_t, d_t).t = 1 \dots n$, kjer je x_t vhodna vrednost, vhodni vektor in d_t je neka želena vrednost. Funkcija je podana z naslednjo enačbo:

$$y = f(x) = w\phi(x) + b$$

Zgornja funkcija je definirana v nelinearnem večdimenzionalnem prostoru, v katerega smo preslikali vhodne primere. Koeficienta w in b se določata z minimizacijo naslednjih enačb:

$$R_{SVM}(C) = C\frac{1}{n}\sum_{t=1}^{n} L_\varepsilon(d_t, y_t) + \frac{1}{2}||w||^2,$$

$$L_\varepsilon(d,y) = \begin{cases} |d-y| - \varepsilon & |d-y| \geq \varepsilon \\ 0 & sicer \end{cases}$$

Prva izmed zgornjih dveh funkcij je funkcija *regulacije tveganja* (angl. regularized risk function), prvi del enačbe predstavlja empirično napako, ki se meri z ε neobčutljivo funkcijo napake, ki je predstavljena z drugo enačbo izmed zgornjih dveh. Kot že omenjeno, sta konstanta C in ε uporabniško določena. Parameter C skrbi za to, da bo algoritem težil k čim širšemu robu in da bo napaka v postopku učenja čim manjša. Če izberemo preveliko vrednost parametra C, se nam to pri točnosti rezultatov ne bo poznalo, le učni čas bo nekoliko daljši. Do neugodnih rezultatov pride, če izberemo premajhen C. Optimalna vrednost parametra se določa glede na empirično znanje, izkušnje in na podlagi testiranj (Tay in Cao 2001).

Da dobimo vrednosti koeficientov w in b, se funkcija regulacije tveganja prevede v naslednjo enačbo, v katero se uvedejo pozitivne dopolnilne spremenljivke ξ_i in ξ_i^*:

Torej minimiziramo funkcijo:

$$R_{SVM_\varepsilon}(w,\xi^{(*)}) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$

pri pogojih:

$$d_i - w\phi(x_i) - b_i \leq \varepsilon + \xi_i$$

$$w\phi(x_i) + b_i - d_i \leq \varepsilon + \xi_i^*, \xi^* \geq 0$$

Z vpeljavo Lagrangeovih multiplikatorjev in ob upoštevanju danih omejitev se odločitvena funkcija $y = f(x) = w\phi(x) + b$ prevede v končno enačbo:

$$f(x,\alpha_i,\alpha_i^*) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x,x_i) + b$$

6 Primerjava metode podpornih vektorjev in nevronskih mrež na primeru napovedovanja gibanja trga delnic

6.1 Programska oprema

Za analiziranje in procesiranje podatkov v diplomski nalogi sem uporabil tri različne programe. Prvi je programski paket MATLAB podjetja MathWorks, ki je namenjen izračunu kompleksnejših matematičnih problemov. Temelji na svojem programskem jeziku, ki je integriran v interaktivno okolje. MATLAB sem uporabljal predvsem v postopku priprave in urejanja podatkov ter pri računanju spremenljivk. Prav tako sem ga uporabljal pri izračunih kriterijev zmogljivosti.

Drugi program, ki sem ga uporabil, je Orange Data Mining. To je odprtokodni program za podatkovno analizo in podatkovno rudarjenje. Deluje skozi vizualno interaktivno okno, prav tako ima integrirano skriptiranje v programskem jeziku Python. Orange Data Mining sem uporabljal pri procesiranju podatkov z metodami podpornih vektorjev.

Tretji uporabljeni program pa je bil Multiple Back-Propagation. Gre za brezplačen program za učenje nevronskih mrež z algoritmom vzvratnega popravljanja. Program sem uporabljal z namenom procesiranja podatkov z nevronskimi mrežami.

6.2 Zbiranje in urejanje podatkov

V eksperimentalnem delu sem primerjavo obeh metod opravil na treh različnih delniških indeksih. Prvi je japonski delniški indeks Nikkei 225, drugi je ameriški S&P 500, tretji pa je evropski indeks Veurx. Podatke sem prenesel s spletne strani finance.yahoo.com, in sicer za obdobje petih let, od 1. januarja 2010 do 31. decembra 2014. V samem procesu pripravljanja vhodnih in izhodnih spremenljivk modela sem zaradi izračunov, ki so zahtevali zamaknjene dneve, izgubil dvajset dni v januarju 2010 in zadnjih pet dni v decembru 2014. Podatki, na katerih je potekala konkretna analiza, so obsegali čas od 21. januarja 2010 do 26. decembra 2014.

Kot podatkovni set sem določil *tržno ceno ob koncu dneva* (angl. closing price). Prvotne podatke sem pretvoril v *petdnevno relativno razliko v ceni*, ki se izraža v odstotkih (angl. five-day relative in percentage of price; RDP). S takšno transformacijo bodo podatki postali bolj simetrični in bodo bolj sledili normalni porazdelitvi, poleg tega bo takšna transformacija povečala moč napovedi nevronskih mrež (Thomason v Tay in Cao 2001; Parsad Das in Padhy 2012).

Vhodni model določa pet spremenljivk. Prve štiri predstavljajo vrednost relativne razlike v ceni v razmiku petih dni (RDP5, RDP10, RDP15 in RDP20), zadnja pa je transformirana tržna cena ob koncu dneva, ki jo dobimo z določitvijo petnajstdnevne *eksponentne drseče sredine* (angl. exponential moving average; EMA15) omenjene tržne cene. Transformacijo izvedemo zato, da odstranimo trend iz cene. Z uporabo spremenljivke EMA15 želimo obdržati čim več informacij, ki se pridobijo z osnovno ceno, saj pretvorba cen v spremenljivke RDP lahko pripelje do izgube nekaj uporabnih informacij (Thomason v Tay in Cao 2001; Parsad Das in Padhy 2012).

Izhodno spremenljivko RDP+5 sem izračunal tako, da sem najprej zgladil tržno ceno ob koncu dneva s tridnevno eksponentno drsečo sredino, saj glajenje odvisne spremenljivke velikokrat poveča napovedno moč nevronskih mrež. Odstranil sem tudi vse *osamelce* (angl. outliner), saj bi zaradi njih le s težavo prišli do učinkovite rešitve. Vrednosti, večje oziroma manjše od +/- 2, so izbrane kot osamelci in so nadomeščene s +/-1,99. Poleg tega sem vse podatke razvrstil med vrednosti -0,9 in 0,9, saj imam v podatkih tako pozitivne kot negativne vrednosti (Thomason v Tay in Cao 2001; Parsad Das in Padhy 2012). Izračuni za vse spremenljivke so podani spodaj, sam sem jih opravil s programom MATLAB, napisana koda je priložena v prilogah.

Vhodne spremenljivke:

EMA15 = $P(t) - EMA_{15}(t)$

RDP5 = $\frac{p(t)-p(t-5)}{p(t-5)} * 100$

RDP10 = $\frac{p(t)-p(t-10)}{p(t-10)} * 100$

RDP15 = $\frac{p(t)-p(t-15)}{p(t-15)} * 100$

RDP20 = $\frac{p(t)-p(t-20)}{p(t-20)} * 100$

Izhodna spremenljivka:

RDP+5 = $\overline{(p(t+5)} - \frac{\overline{p(t)}}{p(t)*100}\ \overline{p(t)} = \overline{EMA_3(t)}$

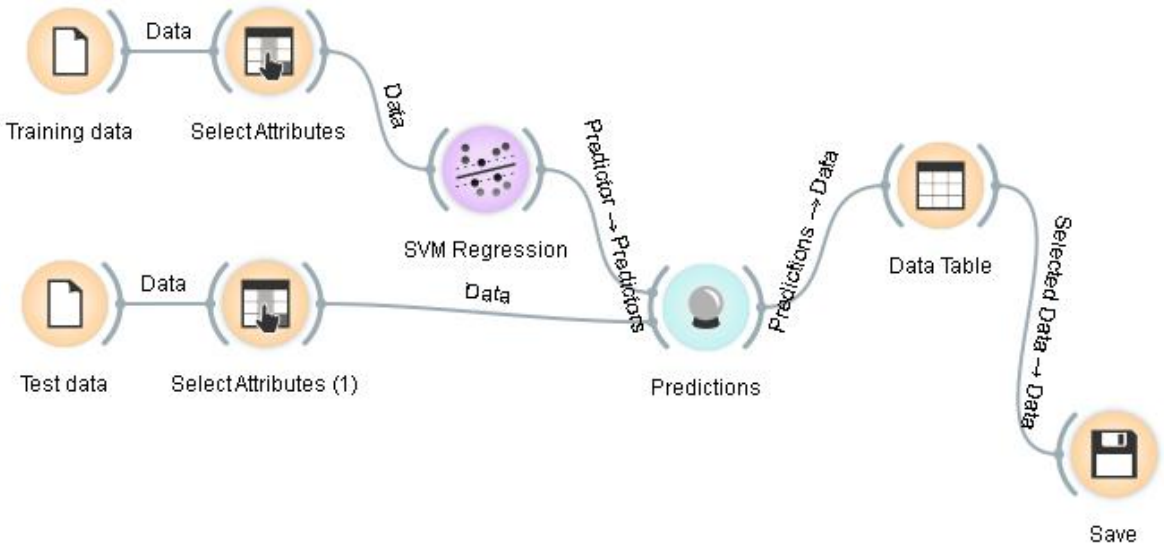
$EMA_n(t)$ je n-dnevna eksponentna drseča sredina i-tega dneva in p(i) je tržna cena ob koncu i-tega dneva.

Vsi trije podatkovni seti so bili razdeljeni na podatke za učenje določene metode (80 % podatkov) in podatke za testiranje metod (20 %). Vsak indeks je v podatkih imel 1213 enot, od tega jih je bilo 970 namenjenih učenju, 243 enot pa je bilo namenjenih testiranju.

6.3 Učenje metode podpornih vektorjev in nevronskih mrež

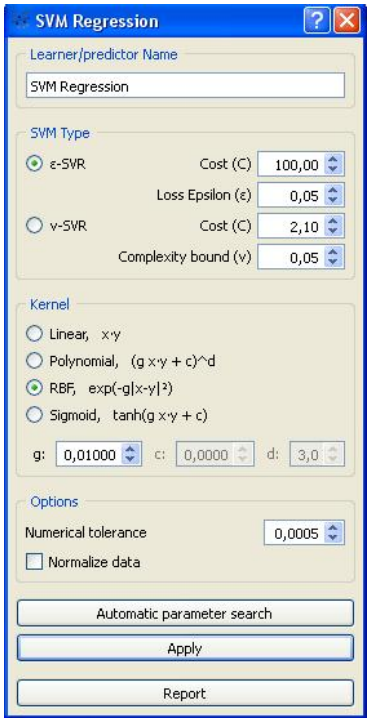
V diplomski nalogi primerjam omenjeni metodi in njuno učinkovitost na področju napovedovanja delniških trgov oziroma finančnih časovnih vrst. Metodo SVM sem implementiral, kot že omenjeno, s programskim orodjem Orange Data Mining. Vhod sta predstavljali datoteka z učnimi primeri in datoteka s testnimi primeri. Kot odvisno spremenljivko oziroma »class« v programu Orange sem nastavil RDP+5.

Slika 6.1: : Diagram poteka za analizo SVM v programu Orange Data Mining.



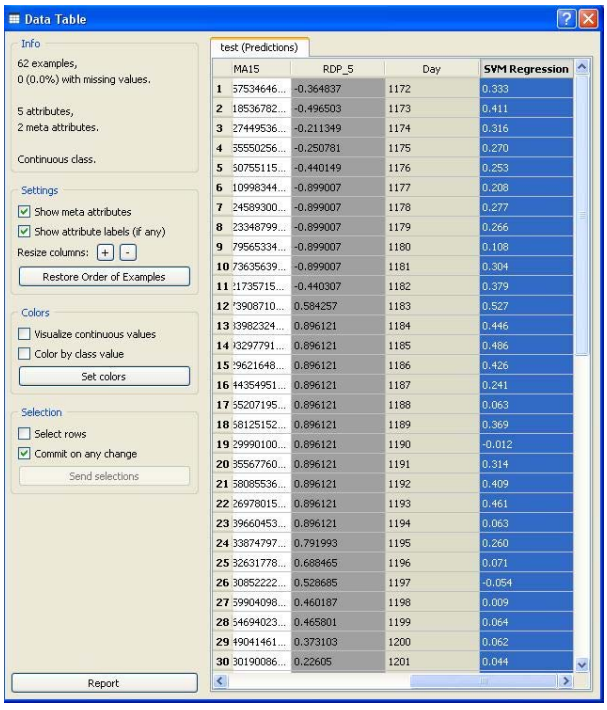
Za treniranje SVM sem izbral radialno jedrno funkcijo, saj deluje zelo dobro, če delamo z zglajenimi podatki (Tay in Cao 2001). Radialna jedrna funkcija je za izbrano vrednost parametra δ podana s $K(x,y)e^{-\delta|x-y|^2}$ (Kononenko in Robnik Šikonja 2010). V primerjavi s polinomsko jedrno funkcijo radialna funkcija daje boljše rezultate in se hitreje uči. Tay in Cao predpostavljata, da vrednosti $\delta = 10$, C=100 in ε = 0.001 dajejo najboljše možne rezultate (Tay in Cao 2001). Sam sem nastavil naslednje vrednosti: $\delta = 1$, saj mi je ta vrednost podajala nekoliko boljše rezultate kot predlagana vrednost, C = 100 in ε = 0.05, saj je v programu Orange to najmanjša vrednost, ki jo lahko nastavim.

Slika 6.2: : Definiranje parametrov za analizo SVM v programu Orange Data Mining.



Na testnih podatkih sem opravil še napoved glede na naučene SVM, dobljene podatke sem izvozil za nadaljnjo analizo oziroma preverjanje učinkovitosti metode.

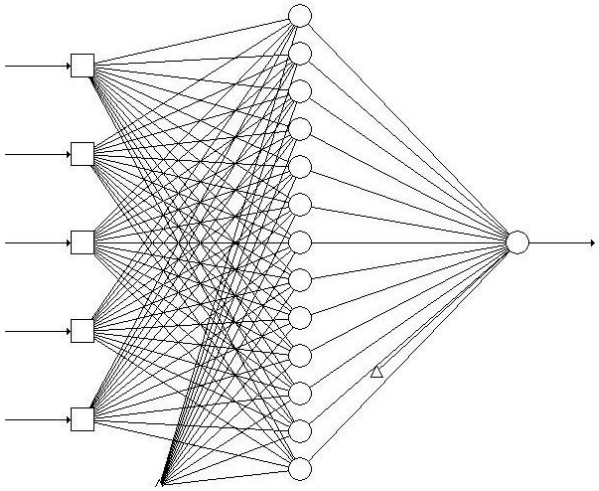
Slika 6.3: : Napoved metode SVM v programu Orange Data Mining



Metodo SVM v diplomski nalogi primerjam z nevronskimi mrežami. Za analiziranje podatkov sem izbral standardno trinivojsko nevronsko mrežo z *algoritmom vzvratnega popravljanja* (angl. back propagation; BP). Pet nevronov je na vhodnem nivoju, kar je enako

številu indikatorjev oziroma neodvisnih spremenljivk. Izhodni nevron je samo en, saj imam v modelu samo eno odvisno spremenljivko, in sicer RDP+5. Število skritih nevronov sem izračunal po naslednji formuli: $W \leq \frac{M}{1B}$, kjer W predstavlja število medsebojno povezanih uteži, ki morajo zadoščati naslednji enačbi: $W = (I + O) * H$, kjer je M število učnih elementov, I je število vhodnih nevronov, O predstavlja število izhodnih nevronov in H je število skritih nevronov (Tay in Cao 2001). Glede na podano enačbo sem nevronski mreži definiral trinajst skritih nevronov.

Slika 6.4: Arhitektura uporabljene nevronske mreže v programu Multiple Back-Propagation.



Za aplikacijo nevronskih mrež na zbrane podatke sem uporabil programsko orodje Multiple Back-Propagation. Parametre za učenje nevronskih mrež sem definiral v komunikacijskem oknu samega programa. *Učno hitrost* (angl. learning rate) sem nastavil na 0,1 in momentni člen (ena izmed modifikacij algoritma) na 0.9. S takimi nastavitvami ima BP nevronska mreža najboljšo napovedno moč z najmanjšim številom učnih iteracij. Skriti in izhodni nevroni za procesiranje uporabljajo sigmoidno funkcijo (Tay in Cao 2001).

6.4 Rezultati

Ker cilj moje diplomske naloge ni točna napoved vrednosti indeksa oziroma točna napoved cene delnice, sem učinkovitost obeh metod preverjal z naslednjimi statističnimi merami: *normalizirana srednja kvadratna napaka* (angl. normalized squared error; NMSE), *sredinska absolutna napaka* (angl. mean absolute error; MAE) in *smerna simetrija* (angl. directional symmetry; DS). NMSE in MAE merita odklon med dejansko vrednostjo in napovedano vrednostjo. Manjša ko je vrednost obeh kazalnikov, manjša je razlika med napovedanimi rezultati in dejanskimi vrednostmi časovne vrste. Torej so manjše vrednosti kazalnikov bolj optimalne. DS je kazalnik, ki nam pove, ali so bile smeri gibanja spremenljivke RDP+5

pravilno napovedane. Vrednost je izražena v odstotkih, kar pomeni, da so večje vrednosti bolj optimalne (Tay in Cao 2001). Sledijo obrazci za izračun omenjenih statističnih količin.

$$\text{NMSE} = 1/(\delta^2 n) * \sum_{t=1}^{n}(a_t p_t)^2 \quad \delta^2 = \frac{1}{n-1} * \sum_{t=1}^{n}(a_t - \bar{a})^2$$

$$\text{MAE} = \frac{1}{n} * \sum_{t=1}^{n} |a_t - p_t|$$

$$\text{DS} = \frac{100}{n} * \sum_{t=1}^{n} d_t \quad d_t = \begin{cases} 1 \ (a_t - a_{t-1})(p_t - p_{t-1}) \geq 0 \\ 0 \ \check{c}e \ je \ drugače \end{cases}$$

V zgornjih enačbah vrednost a_t predstavlja dejansko vrednost in p_t napovedano vrednost.

Kot že omenjeno, sem podatke vseh treh borznih indeksov analiziral s programoma Orange Data Mining in Multiple Back-Propagation. Po učenju obeh metod sem s programoma izvedel še napoved novih vrednosti. Te vrednosti sem izvozil in jih primerjal z dejanskimi v programu MATLAB. Napisana programska koda je v prilogah. S programom MATLAB sem izračunal predhodno omenjene statistične mere. Rezultati so v spodnji tabeli.

Tabela 6.1: Izračunane vrednosti kazalcev uspešnosti.

	SVM			BP		
Delniški indeks	NMSE	MAE	DS	NMSE	MAE	DS
Nikkei 225	1,2574	1,1920	66,1157	1,3523	1,2766	63,1148
S&P 500	1,0091	0,9445	63,9344	1,0226	0,9280	62,9032
Veurx	1,6492	1,2602	62,6016	2,1990	1,5772	60,4839

Iz zgornje tabele lahko razberemo, da se je metoda SVM bolje izkazala pri napovedovanju spremembe vrednosti delniškega indeksa od nevronskih mrež. Obe statistični vrednosti (normalizirana srednja kvadratna in srednja absolutna napaka), ki kažeta odklon napovedi od dejanske vrednosti, sta manjši pri metodi SVM. Največja razlika med metodama se kaže prav pri indeksu Veurx, saj se normalizirana srednja kvadratna napaka razlikuje kar za 0,5498, tudi razlika pri srednji absolutni napaki je za omenjeni indeks največja med vsemi. Najmanjšo razliko med vrednostjo normalizirane srednje kvadratne napake ima indeks S&P 500, razlikuje se le za 0,0949. Iz tega lahko sklenem, da je metoda podpornih vektorjev učinkoviteje predvidevala spremembo vrednosti indeksa, čeprav je vrednost srednje absolutne napake za indeks S&P 500 manjša pri aplikaciji nevronskih mrež.

Smer gibanja delniških indeksov je metoda podpornih vektorjev prav tako napovedovala bolj natančno. Najbolj točno napoved smeri gibanja indeksov je metoda dosegla za indeks Nikkei 225. Ta znaša 66% točnost napovedi.

6.5 Komparativna analiza

V tem delu diplomske naloge želim primerjati dobljene rezultate z rezultati drugih avtorjev, ki so v svoji študiji uporabili enako metodologijo za preverjanje delovanja metode podpornih vektorjev na delniških trgih. V primerjalni analizi bom primerjal rezultate naslednjih kriterijev uspešnosti: normalizirana srednja kvadratna napaka, srednja absolutna napaka in smerna simetrija. Za primerjavo rezultatov sem si izbral naslednji študiji:

- Francis E. H. Tay in Lijuan Cao: Application of support vector machines in time series forecasting (2001)
- Shom Prasad Das in Sudarsan Padhy: Support Vector Machines for Prediction of Future Prices in Indian Stock Market (2012)

Prva raziskava (avtorja Tay in Cao) se ukvarja z ustreznostjo aplikacije metode SVM na napovedovanje finančnih časovnih vrst. Efektivnost metode primerjata z nevronskimi mrežami (nevronske mreže z algoritmom vzvratnega učenja). Za podatkovne sete vzameta indekse s *chicaškega menjalnega trga* (angl. Chicago mercantile market). Uspešnost preverjata z že omenjenimi statističnimi ocenami. Avtorja ugotavljata, da je bila metoda SVM uspešnejša od BP nevronskih mrež (Tay in Cao, 2001). V spodnji tabeli so povzeti rezultati njune analize.

Tabela 6.2: : Ocene uspešnosti za raziskavo »Application of support vector machines in time series forecasting«.

Delniški indeks	SVM			BP		
	NMSE	MAE	DS	NMSE	MAE	DS
CME-SP	0,9365	0.2361	58,29	1,0485	0,2556	55,27
CBOT-US	1,297	0,3425	45,22	1,3424	0,3609	44,72
CBOT-BO	1,1349	0,2989	46,73	1,1877	0,3149	41,7
EUREX-BUND	1,3337	0,3502	43,21	1,3478	0,352	41,2
MATIF-CAC40	1,2083	0,4105	45,22	1,2379	0,4224	42,22

Iz zgornje tabele lahko razberemo, da je bila metoda SVM uspešnejša od metode nevronskih mrež glede na prav vse ocene uspešnosti. Avtorja raziskavo zaključujeta s sklepom, da bi bila izbira metode SVM ugodnejša za napovedovanje gibanja trga delnic. Če primerjamo rezultate

z mojimi, lahko ugotovimo, da so si vrednosti precej podobne, tako pri napakah razlike kot pri odstotku napovedi smeri indeksov. Izpostavim lahko le, da imam v svojih rezultatih nekoliko višje vrednosti pri srednji absolutni napaki in normalizirano srednjo kvadratno napako tako pri metodi SVM kot pri metodi nevronskih mrež. Kljub temu pa lahko izpostavim, da sem sam dobil nekoliko boljše ocene pri napovedovanju gibanja smeri borznega indeksa. Razlika med indeksoma z najboljšo oceno smerne simetrije je 7,8 %.

Avtorja druge študije, ki sem si jo izbral, sta Shom Prasad Das in Sudarsan Padhy. Raziskujeta aplikacijo metod SVM za napovedovanje cen delnic na indijskem borznem trgu. Zanima ju, kako uspešna bo metoda SVM v primerjavi z nevronskimi mrežami glede na to, da se narava borznih trgov razlikuje glede na regijo. Avtorja ugotavljata, da je tudi na indijskem borznem trgu metoda SVM uspešnejša od metode BP nevronskih mrež glede na dane statistične ocene uspešnosti (Prasad Das in Padhy 2012).

Tabela 6.3: : Ocene uspešnosti za raziskavo »Support Vector Machines for Prediction of Future Prices in Indian Stock Market«.

Delniški indeks	SVM			BP		
	NMSE	MAE	DS	NMSE	MAE	DS
BANK NIFTY	0,929	0,238	91,251	1,335	0,250	83,880
CNX 100	1,012	0,382	87,170	1,333	0,381	85,672
CNX INFRA	1,032	0,293	88,192	1,471	0,299	80,895
S&P CNX 500	1,152	0,332	85,671	1,378	0,336	80,597
S&P CNX NIFTY	1,019	0,389	85,170	1,100	0,403	88,059

Zgornja tabela prikazuje zbrane rezultate normalizirane srednje kvadratne in srednje absolutne napake ter smerne simetrije za omenjeno študijo. Razberemo lahko, da se vrednosti NMSE gibajo med 0,929 za indeks BANK NIFTY in 1,152 za S&P CNX 500 za metodo SVM. Opazimo, da so vrednosti srednje absolutne napake v povprečju nekoliko manjše od tistih, objavljenih v študiji avtorjev Tay in Cao. Enako velja tudi za srednjo absolutno napako tako pri metodi SVM kot BP nevronskih mrežah. V primerjavi z mojimi rezultati lahko rečemo, da moji rezultati pri normalizirani srednji kvadratni napaki ne odstopajo veliko, najbolj kritičen je rezultat za borzni indeks Verurx, kjer je napaka precej visoka. Precej slabše pa so ocene srednje absolutne napake. Če se posvetimo rezultatom smerne simetrije, hitro ugotovimo, da so vrednosti precej višje v primerjavi z mojimi rezultati in rezultati avtorjev Tay in Cao. Vidimo, da so napovedali smer gibanja indeksa pri vseh indeksih in obeh

metodah z 80 % točnosti ali več. Iz tega lahko sklepamo, da je metoda SVM na indijskih trgih toliko bolj učinkovita. Zanimivo pa je tudi, da je metoda BP nevronskih mrež pri indeksu S&P CNX NIFTY bolj točno napovedala smer gibanja indeksa od metode SVM. Do neskladja najverjetneje prihaja tudi zaradi izbire različnih parametrov pri učenju metode SVM. Kot sem že omenil, sem moral epsilon nastaviti na vrednost 0,5 $\varepsilon = 0.05$, saj mi program ni dopuščal nižje vrednosti, kar posledično dopušča možnost večje napake pri učenju metode SVM. Poleg tega sem pri svoji analizi uporabil enake parametre za analizo vseh treh indeksov, medtem ko sta avtorja omenjene študije, Shom Prasad Das in Sudarsan Padhy, za vsak podatkovni set določala drugačne parametre na podlagi testiranj, ki sta jih naredila.

6.6 Ugotovitve

Cilj diplomskega dela je primerjati učinkovitost dveh metod strojnega učenja, in sicer metode podpornih vektorjev in BP nevronskih mrež, pri napovedovanju gibanja trga delnic. Raziskovalno vprašanje je bilo: Katera izmed izbranih metod bo natančneje napovedala spremembo vrednosti borznega indeksa in smer njegovega gibanja? Iz rezultatov je razvidno, da je metoda podpornih vektorjev na danem modelu učinkoviteje napovedovala vrednosti spremembe delniških indeksov. To nam kažeta srednja normalizirana kvadratna in srednja absolutna napaka, ki sta bili izračunani za obe metodi. Vrednosti omenjenih kriterijev uspešnosti so bile nižje pri metodi podpornih vektorjev. Pri napovedi smeri gibanja delniških indeksov je v mojem primeru metoda SVM prav tako prekosila napoved nevronskih mrež. Iz tega lahko sklepam, da je metoda podpornih vektorjev primernejša za napovedovanje gibanja trga delnic v primerjavi z BP nevronskimi mrežami. Podobno ugotavljajo tudi drugi avtorji. Tay in Cao predstavita več razlogov, zakaj je bila metoda SVM učinkovitejša. BP nevronske mreže imajo v primerjavi z metodo SVM več nastavljivih kontrolnih parametrov in je težko najti optimalno kombinacijo, ki bi nam vrnila najboljše rezultate. Pri metodi SVM nastavljamo samo tri parametre. Poleg tega ima metoda SVM prednost še zaradi implementacije minimizacij tveganja, saj tako naučen algoritem ni preveč splošen, kar pomeni, da je slednji enako učinkovit tako pri učnih podatkih kot na testnih. Pri BP nevronskih mrežah prihaja do problema prevelikega prilagajanja učnim primerom in so zato tudi slabše pri implementaciji na testne podatke (Tay in Cao 2001).

7 Sklep

Analiza delniških trgov pomeni zelo pomembno panogo na področju financ. Razmah informacijskih tehnologij omogoča lažji dostop do zbirk podatkov kot kdaj prej. Tako so cene delnic in delniških indeksov na voljo skoraj vsakomur z dostopom do spleta. Že dalj časa se raziskovalci ukvarjajo z implementacijo metod strojnega učenja za predvidevanje gibanja delniških trgov. Cilj je poiskati čim bolj preproste modele, ki bi s čim manj informacij dali čim bolj točne napovedi. Modeli, ki imajo moč kakovostne in točne napovedi, so postali pomembno strateško orožje v rokah investitorjev.

V diplomski nalogi sem se osredotočil predvsem na dva algoritma oziroma metodi strojnega učenja. Metoda podpornih vektorjev in metoda nevronskih mrež z vzvratnim algoritmom razširjanja ter njuna aplikacija v analizo delniških trgov so bistvo diplomske naloge. Osredotočil sem se predvsem na to, katera izmed izbranih metod bo natančneje napovedala spremembo vrednosti borznega indeksa in smer njegovega gibanja. Moji podatki so zajemali vrednost indeksa ob koncu dneva, ki sem jo prilagodil izbranemu modelu za napovedovanje gibanja trga. Pokaže se, da med izbranima metodama pride do odstopanj v točnosti napovedi. Za napovedovanje gibanja trga delnic se je kot učinkovitejša metoda izkazala metoda podpornih vektorjev. Podobno ugotavljajo tudi drugi avtorji na omenjenem področju. Prednost te metode je predvsem v implementaciji minimizacije tveganja v učni algoritem. S to operacijo poskrbimo, da model ni preveč splošen in nam tako da boljšo napovedovalno moč. Metode podpornih vektorjev so torej primernejši algoritem za analizo finančnih časovnih vrst.

8 Literatura:

1. Alpaydin, Ethem. 2010. *Introdution to Machine Learning*. Massachusetts: The MIT Press.

2. Atsalakis, George S. in Kimon P. Valavanis. 2009. Surveying stock market forecasting techniques– Part II: Soft computing methods. *Expert Systems with application* 36: 5932–5941.

3. Bose, Indranil in Radha K. Mahapatra. 2001. Business data mining – a machine learning prespective. *Information & Management* 39: 211–225.

4. Boyd, Stephen in Lieven Vandenberghe. 2004. *Convex Optimization*. New York: Cambridge University Press.

5. Cristianini, Nello in John Shawe-Taylor. 2000. *Support Vector Machines and other kernel-based learning methods*. Cambridge. Cambridge Univesity Press.

6. Enke, David in Suraphan Thawornwong. 2005. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications* 29: 927–940.

7. Han, Jiawei in Micheline Kamber. 2006. *Data Minig: Concepts and Techniques*. San Francisco: Elsevier.

8. Haykin, Simon. 2009. *Neural Network and Learnig Machines*. New Jersey: Person Education, Inc.

9. Huang, Wei, Yoshiteru Nakamori in Shou-Yang Wang. 2005. Forecasting stock market movement direction with support vector machine. *Computer & Opration Research* 32: 2513–2522.

10. Kaastra, Iebeling in Milton Boyd. 1996. Designing a neural netowrk for forecasting financial and economic time series. *Neurocomputing* 10: 215–236.

11. Kononenko, Igor in Marko Robnik Šikonja. 2010. *Inteligentni sistemi*. Ljubljana: Založba FE in FRI.

12. Majhi, Ritanjali in Panda Ganapati. 2007. Stock market prediction of S&P 500 and DJIA using bacterial foraging optimization technique. *IEEE Congres on Evolutionary Computation*: 2569–2575.

13. Stackoverflow. 2016. *Machine learning*. Dostopno prek: http://stackoverflow.com /questions/9480605/what-is-the-relation-between-the-number-of-support-vectors-and-training-data-and (8. januar 2016).

14. Prasad Das, Shom in Sudrsan Padhy. 2012. Support Vector machines for Prediction of Futures Prices in Indian Stock Market. *International Journal of Computer Applications* 41 (3): 22–26.

15. Rey, Tim in Chip Wells. 2012. Integrating data mining and forecasting. *ORMS* 39 (6). Dostopno prek: https://www.informs.org/ORMS-Today/Public-Articles/December-Volume-39-Number-6/Integrating-data-mining-and-forecasting (9. avgust 2015).

16. Sapankevych, I. Nicholas in Ravi Sankar. 2009. Time Series Prediction; Using Support Vector Machines: A survey. *Computational Intelegance Magazine* 4 (2): 24–38.

17. Svilan, Sibil. 1990. *Vrednostni papirji: namen in vrste, oblikovanje in trgovanje ter upravljanje*. Ljubljana: Gospodarski vestnik.

18. Tay, Francis E. H. in Lijuan Cao. 2001. Application of support vector machines in financial time series forcasting. *Omega* 29: 309–317.

19. Vapnik, Vladimir. 1998. *Statistical Learning Theory*. New York: Wiley.

20. Yuan, Charles. 2011. *Predicting S&P 500 Returns Using support Vector Machines: Theroy and Empirics.* St. Louis: Center for Research in Economics and Strategies Fellowship.

21. Zissis, Dimitrios, Elias Xidias in Dimitrios Lekkas. 2015. Real-time vessel behavior prediction. *Evolving Systems*: 1–12.

Priloge

Priloga A: Algoritem za pripravo spremenljivk v modelu:

```
stock = csvread('nikkei225.csv',1,6);
dim = size(stock,1)
Close = flipud(stock)

RDP5_1=[NaN;NaN;NaN;NaN;NaN]
for i=6:dim
RDPt = (Close(i)-Close(i-5))/Close(i-5)*100
RDP5_1 = [RDP5_1;RDPt]
end

RDP10_1=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN]
for i=11:dim
RDPt = (Close(i)-Close(i-10))/Close(i-10)*100
RDP10_1 = [RDP10_1;RDPt]
end

RDP15_1=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN]
for i=16:dim
RDPt = (Close(i)-Close(i-15))/Close(i-5)*100
RDP15_1 = [RDP15_1;RDPt]
end

RDP20_1=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;Na
N;NaN;NaN;NaN]
for i=21:dim
RDPt = (Close(i)-Close(i-20))/Close(i-20)*100
RDP20_1 = [RDP20_1;RDPt]
end


EMA=[]
window_size=15;
EMA = tsmovavg(Close,'e',window_size,1);

EMA15=[]
for i=1:dim
  p=(Close(i))-(EMA(i))
  EMA15=[EMA15;p]
end

close_ema_smooth=[]
close_ema_smooth = smooth(Close,3,'moving');

RDP_5_1=[]
for i=1:dim-5
  p=(close_ema_smooth(i+5)-close_ema_smooth(i))/close_ema_smooth(i)*100
  RDP_5_1=[RDP_5_1;p]
end

RDP5=[NaN;NaN;NaN;NaN;NaN]
for i=6:dim
  if RDP5_1(i) > 2
    i=1.99
  elseif RDP5_1(i)< -2
    i=-1.99
```

```
  else
    i=RDP5_1(i)
  end
 RDP5=[RDP5;i]
end

RDP10=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN]
for i=11:dim
  if RDP10_1(i) > 2
    i=1.99
  elseif RDP10_1(i)< -2
    i=-1.99
  else
    i=RDP10_1(i)
  end
 RDP10=[RDP10;i]
end

RDP15=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN]
for i=16:dim
  if RDP15_1(i) > 2
    i=1.99
  elseif RDP15_1(i)< -2
    i=-1.99
  else
    i=RDP15_1(i)
  end
 RDP15=[RDP15;i]
end

RDP20=[NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;NaN;
NaN;NaN;NaN]
for i=21:dim
  if RDP20_1(i) > 2
    i=1.99
  elseif RDP20_1(i)< -2
    i=-1.99
  else
    i=RDP20_1(i)
  end
 RDP20=[RDP20;i]
end

RDP_5=[]
for i=1:dim-5
  if RDP_5_1(i) > 2
    i=1.99
  elseif RDP_5_1(i)< -2
    i=-1.99
  else
    i=RDP_5_1(i)
  end
 RDP_5=[RDP_5;i]
end

%RDP5 = scaledata(RDP5,-0.9,0.9);
%RDP10 = scaledata(RDP10,-0.9,0.9);
%RDP15 = scaledata(RDP15,-0.9,0.9);
%RDP20 = scaledata(RDP20,-0.9,0.9);
%RDP_5 = scaledata(RDP_5,-0.9,0.9);
```

```
RDP5s = RDP5 - min(RDP5(:));
RDP5s = (RDP5s/range(RDP5s(:)))*(0.9-(-0.9));
RDP5s = RDP5s + (-0.9);

RDP10s = RDP10 - min(RDP10(:));
RDP10s = (RDP10s/range(RDP10s(:)))*(0.9-(-0.9));
RDP10s = RDP10s + (-0.9);

RDP15s = RDP15 - min(RDP15(:));
RDP15s = (RDP15s/range(RDP15s(:)))*(0.9-(-0.9));
RDP15s = RDP15s + (-0.9);

RDP20s = RDP20 - min(RDP20(:));
RDP20s = (RDP20s/range(RDP20(:)))*(0.9-(-0.9));
RDP20s = RDP20s + (-0.9);

RDP_5s = RDP_5 - min(RDP_5(:));
RDP_5s = (RDP_5s/range(RDP_5s(:)))*(0.9-(-0.9));
RDP_5s = RDP_5s + (-0.9);

EMA15s = EMA15 - min(EMA15(:));
EMA15s = (EMA15s/range(EMA15s(:)))*(0.9-(-0.9));
EMA15s = EMA15s + (-0.9);
xlswrite('RDP5.xlsx',RDP5s);
xlswrite('RDP10.xlsx',RDP10s);
xlswrite('RDP15.xlsx',RDP15s);
xlswrite('RDP20.xlsx',RDP20s);
xlswrite('EMA15.xlsx',EMA15s);
xlswrite('RDP_5.xlsx',RDP_5s);
```

Priloga B: Algoritem za izračun ocen uspešnosti metode SVM:

```
M = csvread('verurxtest.csv',1,6);
dim = size(M,1);

N = csvread('verurx_svm.csv',0,0);
%N(:,2) = []; %pobrišem drugi stolpec, prebrane N datoreke
%izra?un NMSE
Mpop = mean(M)
vsota=[]
for i=1:dim
 p=(M(i)-Mpop).^2
 vsota=[vsota;p]
end
vso=sum(vsota);
sig=1/(dim-1)*vso

NMSE=[]
vsota=[]
for i=1:dim
  p=(M(i)-N(i)).^2
  vsota=[vsota;p]
end
vso=sum(vsota);
NMSE=1/(sig*dim)*vso

%izra?un MAE
MAE=[]
```

```
for i=1:dim
  p=M(i)-N(i)
  s=norm(p)
  vsota=[vsota;s]
end
vso=sum(vsota);
MAE=1/dim*vso

%izra?un DS
vsotaDI=[]
for i=2:dim

  if (M(i)-M(i-1))*(N(i)*N(i-1))>=0
    p=1
  else
    p=0
  end
  vsotaDI=[vsotaDI,p]
end
sumDI=sum(vsotaDI)
DS=100/(dim-1)*sumDI
```

Priloga C: Algoritem za izračun ocen uspešnosti nevronskih mrež:

```
M = csvread('nikkei225test.csv',1,5);
dim = size(M,1);

N = csvread('nikkei225bpnn.csv',0,0);
%izra?un NMSE
Mpop = mean(M)
vsota=[]
for i=1:dim
 p=(M(i)-Mpop).^2
 vsota=[vsota;p]
end
vso=sum(vsota);
sig=1/(dim-1)*vso

NMSE=[]
vsota=[]
for i=1:dim
  p=(M(i)-N(i)).^2
  vsota=[vsota;p]
end
vso=sum(vsota);
NMSE=1/(sig*dim)*vso

%izra?un MAE
MAE=[]
for i=1:dim
  p=M(i)-N(i)
  s=norm(p)
  vsota=[vsota;s]
end
vso=sum(vsota);
MAE=1/dim*vso
```

```
%izra?un DS
vsotaDI=[]
for i=2:dim

  if (M(i)-M(i-1))*(N(i)*N(i-1))>=0
    p=1
  else
    p=0
  end
  vsotaDI=[vsotaDI,p]
end
sumDI=sum(vsotaDI)
DS=100/dim*sumDI
```