**UNIVERZA V LJUBLJANI**
**EKONOMSKA FAKULTETA**

**MAGISTRSKO DELO**

# UPORABA STROJNEGA UČENJA PRI ANALIZI VREDNOSTNIH PAPIRJEV

**V Ljubljani, september 2006** **Dragan Šmigič**

**IZJAVA**

Študent Dragan Šmigič izjavljam, da sem avtor tega magistrskega dela, ki sem ga napisal pod mentorstvom **prof. dr. Matjaža Gamsa** in **doc. dr. Silve Deželan** in skladno s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

V Ljubljani, dne 14.09.2006

Podpis:_____

# KAZALO

# 1 UVOD

## 1.1 Problematika in namen magistrskega dela

Iskanje metod, s katerimi bi uspeli predvideti donosnost vloženih sredstev, prevzema vlagatelje ter akademsko sfero že od samega nastanka finančnih trgov. Težavo pri tem predstavlja značilnost vsakega konkurenčnega trga, predvsem pa kapitalskega, da so spremembe razmer na trgu nepredvidljive, nekateri bi rekli celo povsem naključne. Ta nepredvidljivost cenovnih gibanj pravzaprav kaže na učinkovito delovanje kapitalskega trga, katerega osrednji namen je učinkovita alokacija sredstev. Smotrno je namreč, da sredstva, ki jih prek trga investirajo vlagatelji, dobijo obetavna podjetja, ki so jih sposobna maksimalno izkoristiti in tako oplemenititi vložke investitorjev ter s tem ustvariti obojestransko korist.

Kljub temu pa je temeljno vprašanje, s katerim se sooča dobička željni vlagatelj, kako ločiti delnice uspešnih podjetij, ki mu lahko prinesejo dobiček, od tistih neuspešnih, pri katerih lahko ustvari izgubo. Rešitev tega vprašanja je za vlagatelja še težja, če ima opravka z učinkovitim trgom. Teorija učinkovitih trgov namreč pravi, da cena delnice odraža vse kar je o tej delnici v tem trenutku znanega. Na podlagi te teorije lahko sklepamo, da je nemogoče napovedati cene delnic (Mramor, 2000, str. 307). »Prave« cene na tako učinkovitem trgu odražajo vse informacije, ki so v danem trenutku relevantne za vrednost nekega vrednostnega papirja, njihove spremembe pa so zaradi popolnoma nepredvidljivih razlogov v prihodnosti povsem naključne.

Vendar pa empirične raziskave v financah kažejo, da tudi kapitalski trgi niso čisto popolni. Določene tržne anomalije namreč povzročajo, da cenovna gibanja na trgu le niso povsem naključna, saj se v njih dajo razbrati določeni vzorci, ponavljajoče se komponente in trendi, na podlagi katerih se investitor lahko odloči za poteze pri trgovanju in tako na daljši rok dosega nadpovprečne donose.

Identificiranih je bilo kar nekaj anomalij kapitalskega trga, ki nam lahko omogočijo doseganje dobička. Seveda ob pogoju, da jih dovolj natančno modeliramo. Med zanimivejše bi lahko šteli:

- Učinek preobratov, ko posamezne delnice, ki so se v določenem obdobju slabo odrezale, dosegajo v naslednjem obdobju značilno višje donosnosti (Mramor, 2000, str. 314).
- Medsebojne odvisnosti med delnicami, ki kažejo na določeno povezavo med gibanji tečajev posameznih delnic (Hellstrom, 1998).

Glede na to, da kapitalski trgi niso čisto popolni, lahko ugotovimo, da vsi vlagatelji le ne znajo v celoti povzeti bodoča dogajanja na trgu.

Temeljno vprašanje vlagatelja se tako lahko glasi: »Kako izbrati delnice uspešnih podjetij bolje od drugih vlagateljev?«.

Skozi teorijo in prakso so se oblikovale različne analitične metode, s katerimi investitorji poskušajo povzeti zakonitosti cenovnih gibanj, na podlagi katerih potem trgujejo. Poleg najpogostejših, kot sta temeljna in tehnična analiza, so v uporabi najrazličnejše kvantitativne metode, med katerimi se v zadnjem času vse bolj uveljavljajo t. i. metode strojnega učenja[1].

Pri teh metodah gre za algoritme v obliki računalniškega programa, katerih naloga je postopna izgradnja modela, ki dosledno opisuje določen pojav (Kononenko, 1997, str. 33). Ta način izgradnje modelov (učenja), ki ga omogoča stroj (računalnik), se v določenih primerih izkaže za uspešnejšega v primerjavi z abstraktnim mišljenjem (učenjem), ki ga je sposoben izvesti človek. Metode na podlagi danih vzorcev podatkov (vzročno-posledičnih, npr. v obliki izjavnega računa) izoblikujejo aproksimacijo funkcije vsebovane v teh podatkih (model podatkov). Naslednjič, ko tako naučenemu modelu predložimo nove vzročne podatke, ki niso bili vsebovani v učni množici podatkov, poskušamo predvideti posledice le-teh. Na ta način je možno na podlagi zgodovinskih tržnih podatkov, na katerih se algoritem nauči določenega modela, poskusiti predvideti bodoča gibanja na trgu.

Strojno učenje lahko na različne načine uporabljamo za analizo delnic. Najpogosteje smo priča uporabi nevronskih mrež pri modeliranju cenovnih gibanj posamezne delnice na borzi. Na to lahko gledamo kot na nelinearno regresijo, kjer poskušamo najti aproksimacijo funkcije skozi točke v večdimenzionalnem prostoru.

Vendar pa obstajajo številne druge možnosti uporabe metod strojnega učenja, s katerimi je možno bolje povzeti nekatere izmed anomalij kapitalskega trga. Eden izmed načinov, je tudi metoda analize delnic s pomočjo rangiranja le-teh (Hellstrom, 1998). Na ta način lahko poskusimo ugotoviti, katera delnica bi v prihodnosti lahko nadpovprečno dobro kotirala glede na ostale izbrane delnice. Metode strojnega učenja so primerne za uporabo skupaj z obravnavano metodo rangiranja delnic, in sicer v tistem delu, kjer je potrebno rang tako rangiranih delnic modelirati, ter na podlagi tega modela izvesti napoved bodoče nadpovprečno uspešne delnice.

Poleg prikladnejše ureditve podatkov za strojno učenje, je prednost te metode ravno v tem, da na ta način lahko pri modeliranju bolje »zajamemo« že omenjeni anomaliji preobratov na trgu in povezanost med gibanji tečajev posameznih delnic.

---

[1]Temu področju je posvečenih kar nekaj znanstvenih konferenc, kot sta CF (Computational Finance) od leta 1993 ter CIFEr (Computational Intelligence in Financial Engineering) od leta 1995.

Namen tega dela je teoretično predstaviti najbolj znane metode strojnega učenja ter možne načine njihove uporabe pri analizi gibanj na kapitalskem trgu. Sama možnost in način uporabe strojnega učenja je v veliki meri odvisen od načina delovanja in lastnosti kapitalskega trga, zaradi česar bo tudi temu dan določen poudarek v tej nalogi. Zaradi določenih prednosti, ki bi jih lahko imela metoda rangiranja delnic v kombinaciji s strojnim učenjem, bo to delo še posebej namenjeno preučitvi praktične uporabe te metode na konkretnih borznih podatkih.

## 1.2 Cilj magistrskega dela

Cilj magistrskega dela je ugotoviti, ali lahko s pomočjo metod strojnega učenja razberemo gibanja delniških tečajev v tolikšni meri, da si lahko pomagamo pri trgovanju z delnicami. Dodaten poudarek pa je na ugotavljanju primernosti analitične metode z rangiranjem delnic v kombinaciji s strojnim učenjem.

Uspešnost takšnih metod lahko preverimo na več načinov. Kot kriterij za uspeh na daljši rok je bila izbrana stopnja pravilno napovedanih donosnosti s pozitivnim predznakom (angl. hit rate), na krajši rok pa poleg tega kazalca še skupna donosnost, ki bi jo dosegli pri vlaganju na tej podlagi. Relativno visoka stopnja pravilno napovedanih pozitivnih donosnosti, kot tudi visoka skupna donosnost naj bi pokazale, ali je omenjeni način analiziranja delnic na podlagi podatkov o preteklem trgovanju z delnicami uporaben za trgovanje.

## 1.3 Metode dela

Metode dela, uporabljene pri izdelavi magistrskega dela, temeljijo na preučitvi teorije, na podlagi katere so razviti algoritmi strojnega učenja, teorije o finančnih trgih ter spoznanja o empirični uporabi strojnega učenja pri napovedovanju gibanj na borzah vrednostnih papirjev, s čimer so kvalitativno ovrednoteni možni načini uporabe strojnega učenja pri analizi vrednostnih papirjev.

Praktični preizkus uporabe strojnega učenja pri analizi delnic je izveden s pomočjo računalniškega programa, imenovanega “Napovedovalec”, razvitega v ta namen. Napovedovalec je implemenacija algoritma za rangiranje delnic, pri katerem bi glede na doseženi rang posamezne delnice v preteklosti poskušali izbrati delnico z nadpovprečnim rangom v prihodnosti. Za izbiro nadpovprečne delnice je potrebno zgraditi model preteklih rangov za vsako delnico posebej in se na podlagi tega modela odločiti za delnico z nadpovprečnim rangom v prihodnosti. Za modeliranje program uporablja že omenjene metode strojnega učenja, torej metode izgradnje odločitvenih dreves, uporabe nevronskih mrež ter metode najbližjih sosedov.

Pri izdelavi magistrskega dela je uporabljena literatura domačih in tujih avtorjev, objavljena v knjigah in prispevkih s področja teorije strojnega učenja ter poslovnih financ. Področje uporabe strojnega učenja pri analizi dogajanja na kapitalskem trgu pa je relativno novo, zato je posledično dan poudarek predvsem tujim znanstvenim člankom. Za praktični preizkus uspešnosti napovedovanja delnic pa so uporabljeni arhivski podatki o trgovanju na Ljubljanski borzi.

## 1.4 Vsebinska zasnova dela

Magistrsko delo je razdeljeno na sedem poglavij. Drugo in tretje poglavje, ki sledita prvemu uvodnemu, sta teoretične narave. Zaradi interdisciplinarnosti med informatiko (strojno učenje) in finančno ekonomijo je drugo poglavje posvečeno informatiki, tretje pa finančni analizi. V drugem poglavju je pojasnjen pojem znanja ter načini »pridobivanja« znanja iz večje količine podatkov s pomočjo stroja (računalnika), torej strojnega učenja. V tem poglavju so predstavljene metode strojnega učenja, ki bi lahko bile uporabne pri analizi cenovnih gibanj na kapitalskem trgu, predvsem že prej omenjena odločitvena drevesa, nevronske mreže, metoda k-najbližjih sosedov ter metoda podpornih vektorjev. Prve tri metode so tudi uporabljene v praktičnem delu magistrskega dela v četrtem poglavju. Prav tako so v drugem poglavju opisani kriteriji, s katerimi merimo uspešnost metod strojnega učenja, torej kako dobro se posamezni algoritem nauči znanja izdanih podatkov.

Tretje poglavje opisuje kapitalski trg in njegovo lastnost, ki je ključna za napovedovanje, njegove anomalije. V tem poglavju so opisane tudi klasične analitične metode, kot sta tehnična in temeljna analiza delnic, s katerimi analitiki poskušajo napovedati bodoča gibanja cen delnic, ter možni načini uporabe strojnega učenja pri napovedovanju delnic.

Četrto poglavje predstavlja konkretno uporabo predstavljenih interdisciplinarnih znanj pri razvoju novega algoritma napovedovanja delnic s pomočjo strojnega učenja in s tem predstavlja bistvo magistrskega dela, saj je v njem predstavljena uporaba prej opisanih metod strojnega učenja pri sami analizi. Poglavje je usmerjeno na predstavitev metode razvrščanja delnic in napovedovanja njihove razvrstitve v prihodnosti, kar bi lahko bila uporabna informacija za nakup oziroma prodajo. Na tem mestu je tudi podrobneje predstavljena implementacija sistema Napovedovalec, ter knjižnica razredov, ki je uporabljena pri programiranju. V petem poglavju so predstavljeni rezultati analize oziroma napovedovanja s tako razvitim algoritmom. V šestem poglavju pa so izražene sklepne misli glede uspeha in primernosti algoritma pri trgovanju.

# 2 STROJNO UČENJE

Naloga strojnega učenja je pridobivanje znanja. Znanje lahko definiramo na več načinov. Najbolj primerna definicija opredeljuje znanje kot interpretacijo informacije, ki jo nosijo podatki (Kononenko, 1997, str. 2). Znanje je bodisi dano vnaprej (npr. podedovano ali v primeru stroja vkodirano), bodisi je rezultat učenja. Učenje srečamo pri skoraj vseh živih bitjih, najbolj pa pride do izraza pri človeku. Učenju živih bitij pravimo naravno učenje. Če pa je učen stroj - računalnik, pravimo takemu učenju avtomatsko učenje ali tudi strojno učenje (angl. machine learning). Postopek naravnega učenja v najpreprostejši obliki lahko vidimo pri otroku, in sicer, kako iz podatkov inducira neko znanje. Lep primer induktivnega učenja lahko zasledimo pri Michalskem (Michalski et al, 1997, str. 12):

*»Recimo, da želimo otroku pojasniti, kaj je to ptica. Za začetek mu lahko pokažemo "kosa" kot pozitiven primer koncepta »ptič«. Seveda je prosto memoriranje lastnosti, ki jih ima kos, nezadovoljivo za prepoznavanje drugih ptic kot primerkov iste kategorije. Očitno je potrebna posplošitev tega primera. Vendar kako široka posplošitev? Da vzpostavimo meje posploševanja, bi bilo potrebno posamezniku predstaviti tudi negativen primer, nekaj kar ni ptica. Tako mu pokažemo "psa". Očitno je ena izmed razlik med psi in pticami ta, da psi nimajo kril. Da bi preveril, ali so vse krilate živali ptice, otrok vpraša, ali je tudi muha ptič. Torej je dejstvo, da ima neko bitje krila, preveč splošna lastnost za zadovoljivo ločevanje med pozitivnimi in negativnimi primerki. Tako je potrebna specializacija opisa. To lahko naredimo z dodajanjem novih lastnosti kakšnega novega primerka že obstoječemu opisu. Lastnost, ki jo opazimo pri kosu, ki pa je ni pri psu, ne pri muhi, je rumen kljun. Otrok lahko pri tem domneva, da vrana ni ptič, ker nima rumenega kljuna. Dejstvo, da je rumeni kljun lastnost, ki jo najdemo samo pri nekaterih pticah, nam daje vedeti, da je opis postal preveč specifičen in ga moramo dodatno posplošiti. Tako lahko preprosto izpustimo zahtevo po rumeni barvi in preprosto zaključimo, da so ptice krilata bitja s kljunom. Ta opis drži tudi za recimo vrabca, kot pozitiven primer koncepta »ptič«.*

Na tem preprostem primeru induktivnega učenja lahko vidimo, kako otrok z empiričnim postopkom kreira opis nekega koncepta in tvori znanje o njem s pomočjo zbiranja lastnosti o konceptu. Pri tem za vsako preveč posplošeno lastnost, ki obsega tudi tiste primere, ki ne spadajo v skupno kategorijo, dodaja lastnosti, ki zožijo opis na skupino primerov, ki ustreza zahtevani kategoriji. Na ta način je učenje nekega koncepta možno izvesti kot preiskovanje skozi prostor možnih opisov (hipotez) koncepta, dokler ne najdemo ustreznega opisa, ki pokriva ciljno skupino primerov. Pri preiskovanju uporabljamo dva osnovna operatorja, in sicer *generalizacijo (posplošitev)* ter *specializacijo*.

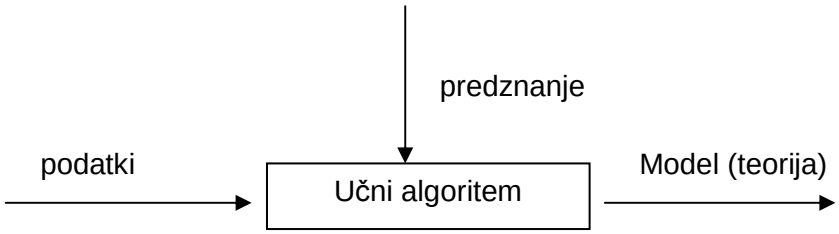
Koncept strojnega (avtomatskega) učenja v obliki računalniškega programa ima isto nalogo in jo na podoben način rešuje, le da pri tem procesu ne potrebuje človeka. Uporabnost strojnega učenja vidim predvsem v tistih problemih, kjer moramo znanje pridobiti iz velike količine podatkov.

## 2.1 Koncept strojnega učenja

Kadar govorimo o strojnem učenju, mislimo na modeliranje podatkov, vendar to učenje oziroma modeliranje poteka bolj ali manj avtomatično, brez pomoči človeka. Zgrajeni model povzema te podatke, torej na nek način predstavlja abstrakcijo teh podatkov in nam omogoča odzivanje na zakonitosti tako povzetih podatkov. Modelu dostikrat rečemo tudi hipoteza ali teorija. Postopku učenja oziroma gradnji modela pravimo algoritem (Kononenko, 1997, str. 34).

Sistemi za strojno učenje so navadno sestavljeni iz učnega algoritma, ki iz množice podatkov tvori novo znanje (model), in izvajalnega agoritma, ki avtomatsko naučeno znanje uporablja za reševanje novih problemov.

**Slika 1: Predstavitev strojnega učenja**




Vir: Kononenko, 1997, str. 34

Model v matematičnem smislu prepoznamo kot funkcijo, ki opisuje relacijo med množico odvisnih in množico neodvisnih podatkov. Glede na zveznost ločimo diskretne in zvezne funkcije.

### 2.1.1 Diskretne funkcije

Problemom, katerih ciljni model je diskretna funkcija, katere zaloga vrednosti, torej vrednosti, ki jih funkcija lahko zavzame, je diskretna neurejena množica, pravimo klasifikacijski problemi. Ko je funkcija naučena, jo uporabljamo za klasifikacijo. Sama klasifikacija pomeni ugotavljanje vrednosti te funkcije pri danih podatkih, torej

vrednostih neodvisnih spremenljivk (Kononenko, 1997, str. 34). Mnogi odločitveni problemi, diagnostični problemi, problemi vodenja in problemi napovedovanja se lahko predstavijo kot klasifikacijski problemi. Tipični primeri so medicinska diagnostika in prognostika, napovedovanje vremena, diagnostika industrijskih procesov, klasifikacija izdelkov po kakovosti (Michalski, 1997str. 34). Tudi napovedovanje uspešnosti delnic, ki je opisano in implementirano v 4. poglavju, je rešeno kot klasifikacijski problem.

### 2.1.2 Zvezne funkcije

Problemom, katerih ciljni model je zvezna funkcija, pravimo regresijski problemi. Zaloga vrednosti, torej vrednosti, ki jih neodvisna spremenljivka v modelu lahko zavzame, je neskončna urejena množica, ki ji pravimo tudi zvezni razred. Tako kot pri klasifikacijskih problemih tudi tukaj ugotavljamo vrednosti funkcije ob danih podatkih. Regresijske funkcije so tako najuporabnejše pri napovedovanju časovnih vrst. Poleg vrednosti funkcije je pogosto zahteva pri regresijskih problemih tudi interval zaupanja, ki nam opisuje zaupanje v vrednost, ki nam jo model predlaga kot rešitev danega problema (Kononenko, 1997, str. 35).

### 2.1.3 Predstavitev znanja in operatorji učenja

Učni algoritem na vhodu dobi predznanje in učne primere. Za tem preiskuje v prostoru možnih hipotez, ki bi lahko opisale učne primere, in kot rezultat vrne zaključno hipotezo. Za predznanje, učne primere in prostor hipotez potrebujemo ustrezno predstavitev. Med samim preiskovanjem prostora hipotez učni algoritem uporablja operatorje za spreminjanje trenutne hipoteze. Naučeno znanje lahko predstavimo na različne načine. V grobem ločimo naslednje predstavitve znanja (Kononenko, 1997, str 61):

1. izjavni račun
2. predikatni račun prvega reda
3. diskriminantne in regresijske funkcije
4. verjetnostne porazdelitve
5. neformalne predstavitve, npr. tekstovne

Ker se pri implementaciji strojnega učenja v sklopu analize vrednostnih papirjev v četrtem delu uporabljajo modeli na osnovi izjavnega računa (odločitvena drevesa), regresijskih funkcij (nevronske mreže) ter verjetnostnih porazdelitev (K-najbližjih sosedov), se bom v nadaljnjem posvetil predvsem tem. Poleg teh pa bo opisana tudi metoda podpornih vektorjev, kot ena izmed najnovejših metod. Kvaliteta naučenega

znanja je seveda eno izmed najpomembnejših vodil pri izbiri metode za rešitev našega problema. Zato so v nadaljevanju predstavljeni tudi nekateri kriteriji s katerimi lahko ocenimo tako pridobljeno znanje.

## 2.2 Odločitvena drevesa

Izjavni račun lahko uporabljamo tako za predstavitev učnih primerov kot tudi za oblikovanje opisa teh primerov, torej hipoteze. Največkrat to hipotezo oblikujemo v skupek odločitvenih pravil. Še večji simbolični pomen pa ta skupek odločitvenih pravil dobi v obliki odločitvenega drevesa.

Izjavni račun je predstavljen z množico atributov v obliki pogojnega dela, kjer so med seboj konjunktivno povezani atributi kot spremenljivke, ki imajo določeno množico možnih vrednosti, ter sklepnega dela, ki je prav tako atribut in mu pravimo razred (Mishalski et al, 1997, str 10). V tabeli 1 lahko vidimo učne primere v obliki podatkov o vremenskih razmerah ter posledični odločitvi, ali je bila v teh razmerah odigrana igra tenisa.

**Tabela 1: Podatki o vremenskih razmerah v obliki izjavnega računa**

| Razmere | Temperatura | Vlaga | Veter | Igra tenisa |
|---|---|---|---|---|
| Sončno | Toplo | visoka | ne | ne |
| Sončno | toplo | visoka | da | ne |
| Oblačno | toplo | visoka | ne | da |
| Deževno | srednje | visoka | ne | da |
| Deževno | mrzlo | normalna | ne | da |
| Deževno | mrzlo | normalna | da | ne |
| Oblačno | mrzlo | normalna | da | da |
| Sončno | srednje | visoka | ne | ne |
| Sončno | mrzlo | normalna | ne | da |
| Deževno | srednje | normalna | ne | da |
| Sončno | srednje | normalna | da | da |
| Oblačno | srednje | visoka | da | da |
| Oblačno | toplo | normalna | ne | da |
| Deževno | srednje | visoka | da | ne |

Vir: Witten, Eibe, 2000, str. 9

Pri odločitvenih pravilih, s katerimi predstavimo opis (hipotezo) na podlagi učnih primerov iz tabele 1, imamo prav tako pogojni del, ki je sestavljen iz konjunkcije atributnih pogojev in sklepnega dela (glej sliko 2).

**Slika 2: Odločitvena pravila, ki povzemajo učne primere iz tabele 1**

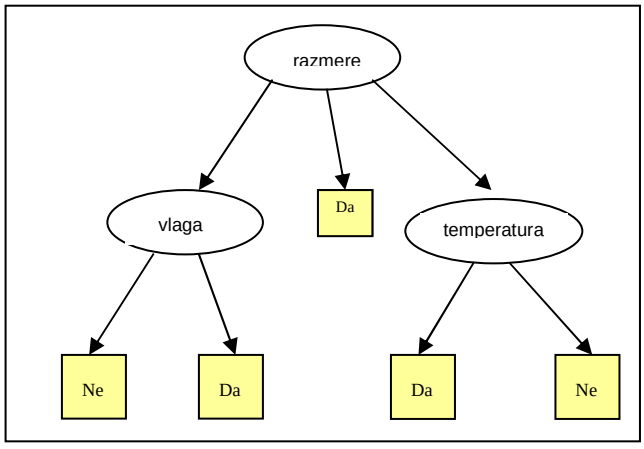
```
če razmere =  sončno in vlaga = visoka   potem igra tenisa = ne
če razmere =  deževno in veter = da      potem igra tenisa = ne
če razmere =  oblačno                    potem igra tenisa = da
če vlaga   =  normalna                   potem igra tenisa = da
```

Vir: Witten, Eibe, 2000, str. 9

Vendar pa predstavitev lahko posplošimo tako, da poleg konjunkcije v pogojnem delu dovolimo tudi druge logične operatorje, kot je disjunkcija, negacija, ekvivalenca itd. Taka posplošitev močno poveča prostor hipotez, kar pa vpliva na povečano kompleksnost učnega algoritma (Witten, Eibe, 2000, str. 9).

Odločitveno drevo kot posebna oblika množice odločitvenih pravil je sestavljeno iz notranjih vozlišč, ki ustrezajo atributom, vej, ki ustrezajo podmnožicam vrednosti atributov in listov, ki ustrezajo razredom. Ena pot v drevesu od korena do lista ustreza enemu odločitvenemu pravilu. Pri tem so pogoji konjunktivno povezani. Vsako drevo lahko pretvorimo v množico toliko odločitvenih pravil, kolikor ima drevo listov. Po drugi strani pa vsake množice pravil ni možno neposredno pretvoriti v odločitveno drevo. Slika 3 prikazuje odločitveno drevo, izdelano na podlagi vremenskih podatkov v obliki izjavnega računa iz tabele 1.

**Slika 3: Odločitveno drevo, izdelano na podlagi vremenskih podatkov**




Vir: Witten, Eibe, 2000, str. 9

Kot je bilo predstavljeno v primeru naravnega učenja pri oblikovanju koncepta »ptič«, tudi pri avtomatskem strojnem učenju poskušamo oblikovati hipotezo, ki bo pokrivala (opisovala) učne primere. Pri tem si predstavljamo, da imamo vse možne hipoteze, ki jih lahko sestavimo iz atributov, ki nastopajo v učnih primerih in je učenje tako

izvedeno s preiskovanjem tega prostora hipotez. Različni algoritmi uporabljajo različne metode preiskovanja. Za preiskovanje prostora hipotez potrebujemo začetno hipotezo in množico operatorjev, ki transformirajo hipotezo v množico naslednikov. Vsakič, ko imamo več naslednikov hipoteze, se mora algoritem na podlagi ocene kvalitete posameznega naslednika hipoteze odločiti, kateri je najboljši. Pri tem je zaradi obsežnosti prostora hipotez potrebno ubrati neko strategijo, kako priti do najboljše ali vsaj zadovoljivo dobre hipoteze, ne da bi bilo potrebno preiskati celoten prostor hipotez, saj bi bilo to ali preveč zamudno ali pa celo nemogoče v nekem normalnem času (Michalski et al, 1997, str. 13). Osnovna ocena kvalitete nekega pravila (hipoteze) je njegova splošnost. Kako velik del prostora pokriva oz. pri posameznem učnem primeru, ali ga pravilno pokriva ali ne. Pravimo, da pravilo pokriva učni primer, če učni primer ustreza pogojnem delu pravila, če razred ustreza sklepnemu delu pravila (če pravilo za dani primer pravilno napove razred) in ravno tako kot na primeru naravnega učenja koncepta “ptič” tudi tukaj uporabljamo operatorje pri učenju simboličnih pravil, posplošitev in obratno operacijo specializacija.

Kadar imamo pogojni del pravila, ki uporablja konjunkcijo pogojev, ga lahko posplošimo na več načinov. Izpustimo en ali več konjunktov iz pogojnega dela ali v pogoju, ki testira pripadnost vrednosti atributa dani množici, razširimo množico vrednosti. Ravno tako pogojni del pravila, ki uporablja konjunkcijo pogojev, specializiramo z obratnimi operacijami: dodamo en ali več konjunktov v pogoj ali pogoju, ki testira pripadnost vrednosti atributa dani množici, zmanjšamo množico vrednosti. Če uporabljamo bolj splošno sintakso pravil, kjer so dovoljene tudi druge operacije (negacije, disjunkcije itd.), se poveča tudi število možnih načinov posploševanja in specializacije pravil (npr. zamenjava konjunkcije z disjunkcijo ali pa disjunktivno podajanje pogojev).

Pri preiskovanju prostora hipotez so potrebne hevristične hipoteze, ki usmerjajo iskanje. Algoritem za preiskovanje ocenjuje s pomočjo funkcij tako trenutno hipotezo kot tudi kvaliteto naslednikov hipoteze. Pri iskanju najboljšega naslednika hipoteze je osnovna naloga algoritma oceniti pomembnost posameznega atributa, ki se pojavlja v učnih primerih za dani učni problem.

V ta namen se uporabljajo (Kononenko, 1997, str 108):

- informacijski prispevek (Hunt et al., 1966),
- razmerje informacijskega prispevka (Quinlan, 1986),
- razdalja dogodkov (Mantaras, 1989),
- povprečna absolutna teža evidence (Good, 1950),
- mera najkrajšega opisa – MDL (Rissanen, 1983),
- J-ocena (Smyth in Goodman, 1990),

- statistiki $\chi^2$ in *G (Press et al., 1988)*,
- ortogonalonost vektorjev porazdelitev – ORT (Fayyad, 1991),
- Gini-indeks (Breiman et al., 1984) ter
- ReliefF (Kira in Rendell, 1992).

Strategije, ki jih algoritmi uporabljajo pri preiskovanju prostora hipotez, so poleg izčrpnega preiskovanja, ki preišče cel prostor hipotez, lahko tudi (Kononenko, 1997, str. 87):

- omejeno izčrpno iskanje (razveji in omeji),
- metoda “najprej najboljši”,
- požrešno iskanje,
- iskanje v snopu,
- lokalna optimizacija,
- gradientno iskanje,
- simulirano ohlajanje,
- genetski algoritmi.

Odločitveno drevo predstavlja klasifikacijsko funkcijo, ki je hkrati simbolični opis in povzetek zakonitosti v nekem danem problemu. Zato je drevo ponavadi zanimivo za strokovnjaka iz problemskega področja, saj lahko iz drevesa razbere določene zakonitosti in strukturo. Zgrajeno drevo lahko uporabimo za klasifikacijo novih primerov. Od korena potujemo po ustreznih vejah do listov. List vsebuje informacijo o številu učnih primerov iz posameznih razredov. Iz frekvenc učnih primerov se oceni verjetnostna porazdelitev primerov. Koristen podatek je tudi število vseh učnih primerov v listu, ki kaže na zanesljivost ocene verjetnostne porazdelitve razredov.

Večina algoritmov, ki so bili razviti za učenje (gradnjo) odločitvenih dreves, je različica osnovnega algoritma, ki uporablja metodo od zgoraj navzdol. Med najbolj znanimi sta algoritem ID3 (Iterative Dichotomizer 3) in njegov naslednik C4.5, ki ju je razvil Ross Quinlan leta 1979 in 1984 (Quinlan, 1986).

Če si kot osnovni algoritem za gradnjo odločitvenih dreves pogledamo algoritem ID3, vidimo, da zgradi odločitveno drevo od zgoraj navzdol, kjer na začetku rešuje vprašanje, kateri atribut naj bo testiran ob korenu drevesa. Da dobi odgovor na to vprašanje, se vsak posamezni atribut pri učnem primeru ovrednoti z uporabo statističnega testa. S tem se določi, kako dobro posamezni atribut klasificira učne primere. Najboljši atribut je tako izbran in postavljen v korenu drevesa. Naslednik korenskega vozla se nato kreira za vsako možno vrednost tega atributa in učni primeri se sortirajo k primernemu novemu nasledniku navzdol po veji skladno z vrednostjo atributa dotičnega učnega primera. Celoten proces se nato ponovi tako,

da algoritem uporabi učne primere, povezane z vsakim naslednjim vozlom, da bi izbrali najboljši atribut, ki bo uporabljen na tej točki drevesa (Mitchell, 1997, str. 55). Tovrstnemu iskanju rečemo požrešno iskanje v prostoru odločitvenih dreves, kjer se algoritem nikoli ne vrača nazaj, da bi znova ocenil prejšnje odločitve. Osrednja funkcija algoritma je torej vsakokratna odločitev, kateri atribut naj bo odločitveni kriterij v posameznem vozlu, ki sestavljajo veje drevesa. Potrebno je izbrati atribut, ki bo kar najbolje klasificiral učne primere (Kononenko, 1997, str. 90). ID3 pri tem kot kvaliteto atributa pri razlikovanju učnih primerov uporablja kvantitativno merilo, in sicer informacijski prispevek. ID3 to merilo uporablja, da bi izbral med kandidati (atributi) pri vsakem koraku med razraščanjem drevesa. Informacijski prispevek atributa pri njegovem potencialnem dodajanju k drevesu nam pove, za koliko se bo zmanjšala nečistost množice podatkov, če bomo učne primere razdelili glede na ta atribut. Nečistost podatkov v neki množici označujemo z *entropijo*. Kadar imamo zbirko učnih primerov *S*, ki vsebuje pozitivne in negativne primere nekega koncepta, označujemo entropijo kot (Mitchell, 1997, str. 55):

$$Entropija(s) = -p_+ \cdot \log_{2\cdot} p_+ \;\; - \;\; p_- \cdot \log_2 p_- \qquad (1)$$

kjer *p+* predstavlja delež pozitivnih primerov v množici *S* in *p-* delež negativnih primerov v množici *S*. Za ilustracijo recimo, da množica S vsebuje 14 primerov nekega dvoznačnega (binarnega) koncepta, ki vključuje 9 pozitivnih primerov, ki pravilno označujejo koncept in 5 negativnih primerov ( [ 9+, 5-] ).

Tako dobimo entropijo množice *S*:

$$Entropija\left([9+, 5-]\right) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0{,}940\,.$$

V primeru, da vsi primeri pripadajo enemu razredu koncepta (so bodisi vsi pozitivni p+ bodisi vsi negativni p-), pa je entropija nič, saj so podatki povsem “čisti”. Entropija je v nasprotnem primeru največja (vrednost entropije je 1), kadar ravno polovica primerov pripada enemu in polovica drugemu razredu, saj so podatki povsem nečisti. Podobno lahko izmerimo entropijo tudi v večznačnih skupinah podatkov, kjer ciljni atribut (razred) lahko zavzame več različnih vrednosti v obliki (Mitchell, 1997, str. 55):

$$Entropija(S) = \sum_{i=1}^{C} - p_i \log_2 p_i \qquad (2)$$

Kot je že bilo rečeno, informacijski prispevek dodanega atributa pove, za koliko se z dodajanjem tega atributa zmanjša entropija. V primeru atributa A tako dobimo informacijski prispevek na sledeč način (Mitchell, 1997, str. 56):

$$Informacijski\ prispevek = Entropija(S) \quad - \sum_{v \in vrednosti(A)} \frac{|Sv|}{|S|} Entropija(Sv), \qquad (3)$$

kjer je:

$Vrednosti\ A$ = množica vseh možnih vrednosti atributa $A$,

$S_v$ = podmnožica $S$, za katero ima atribut $A$ vrednosti $v$.

Prvi del enačbe predstavlja entropijo celotne množice. Drugi del pa pričakovano vrednost entropije po razdelitvi množice $S$ z uporabo atributa $A$. Pričakovana entropija, opisana z drugim delom enačbe, je preprosto vsota entropij vseh podmnožic množice podatkov $S$, kjer je vsaka od njih tehtana glede na delež, ki ga predstavlja v celotni množici $S$.

**Primer:**
Predpostavimo, da je $S$ zbirka učnih primerov, ki predpostavljajo dneve, opisane z vremenskimi atributi, vključno z atributom "*veter*" . Ta lahko zavzame vrednosti močan ali šibek. Množica $S$ vključuje 14 primerov dni ( [ 9+, 5-] ), od katerih je 9 dni opisanih pozitivno, kjer smo imeli dobro vreme, in 5 dni negativno, saj na teh negativnih dneh nismo izvajali aktivnosti zaradi slabega vremena. Od teh 14 učnih primerov predpostavimo, da je 6 pozitivnih in 2 negativna dneva imelo atribut »veter« z vrednostjo šibek, preostanek učnih primerov pa vrednost močan. Informacijski prispevek po sortiranju 14 učnih primerov na osnovi atributa veter lahko izračunamo kot (Mitchell, 1997, str. 58):

Vrednosti (veter) = šibek, močan

$S = ([9+, 5-])$

$S_{šibek} = ([6+, 2-])$

$S_{mocan} = ([3+, 3-])$

$$\begin{aligned} \text{inf. } prispevek &= Entropija(S) \quad - \sum_{v \in (šibak, mocan)} \frac{|Sv|}{|S|} \cdot Entropija(Sv) \\ &= Entropija(S) - (8/14) \cdot Entropija(S_{šibak}) - (6/14) \cdot Entropija(S_{mocan}) \\ &= 0{,}940 - (8/14) \cdot 0{,}811 - (6/14) \cdot 1{,}00 \\ &= 0{,}048 \end{aligned}$$

Na ta način algoritem ID3 izbira najboljši atribut ob vsakem novem vozlu, ki ga doda odločitvenemu drevesu. Kadar se mora odločiti med večimi atributi, za vsakega izmed njih izračuna informacijski prispevek in se odloči za tistega z najvišjim prispevkom.

## 2.3 Metode najbližjih sosedov

Koncept učenja z metodo najbližjih sosedov zahteva shranitev množice učnih primerov. Ko je potrebno klasificirati novi primer, se poišče podmnožica podobnih učnih primerov, ki se uporabijo za napoved razreda novega primera. Ker učenja pri teh metodah skorajda ni, pravimo tej vrsti učenja leno učenje (angl. lazy learning). Največji del obdelave je tako potreben pri klasifikaciji novega primera in ne pri izgradnji modela. Ko želimo napovedati razred novemu primeru, poiščemo med učnimi primeri k-najbližjih primerov in pri klasifikaciji napovemo večinski razred, tj. razred, ki mu pripada največ izmed izbranih k-najbližjih sosedov. Več sosedov kot izberemo, manjša je možnost, da bo algoritem izbral napačen razred. Po drugi strani pa z večanjem števila $k$ povečujemo možnost, da h klasifikaciji prispevajo tudi tisti učni primeri, ki niso dovolj podobni novemu primeru.

Zato je potrebno za vsak problem posebej eksperimentalno določiti optimalni parameter $k$. Velikost okolice novega primera se spreminja dinamično v odvisnosti od gostote učnih primerov v danem podprostoru primerov. S fiksno velikostjo okolice novega primera bi namreč v nekaterih delih prostora lahko dobili preveč bližnjih sosedov in v drugih delih prostora nobenega. Algoritem $k$ najbližjih sosedov (ankl. k-nearest neighbours) je občutljiv na izbrano matriko pri računanju razdalj med novim primerom in učnimi primeri. Največkrat se uporablja Evklidska razdalja. Pri zveznih atributih se uporablja razdalja med normaliziranimi vrednostmi atributov, pri atributih z diskretnimi vrednostmi pa je razdalja med različnima vrednostma enaka 1, med enakima vrednostma pa 0.

**Primer:**
Če vzamemo učni primer $x$, ki je opisan z vektorjem v nekem prostoru učnih primerov z $a_1(x), a_2(x), ..., a_n(x)$, kjer $a_r(x)$ označuje vrednost r-tega atributa posameznega učnega primera $x$. Razdaljo med dvema primeroma $x_i$ in $x_j$ algoritem izračuna na sledeč način:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

V realnih življenjskih primerih vsi atributi niso vedno enako pomembni, zato se pogosto uporablja uteževanje vrednosti atributov na celotno razdaljo med dvema primeroma, tako da bolj podobni atributi bolj vplivajo na razdaljo.

## 2.4 Nevronske mreže

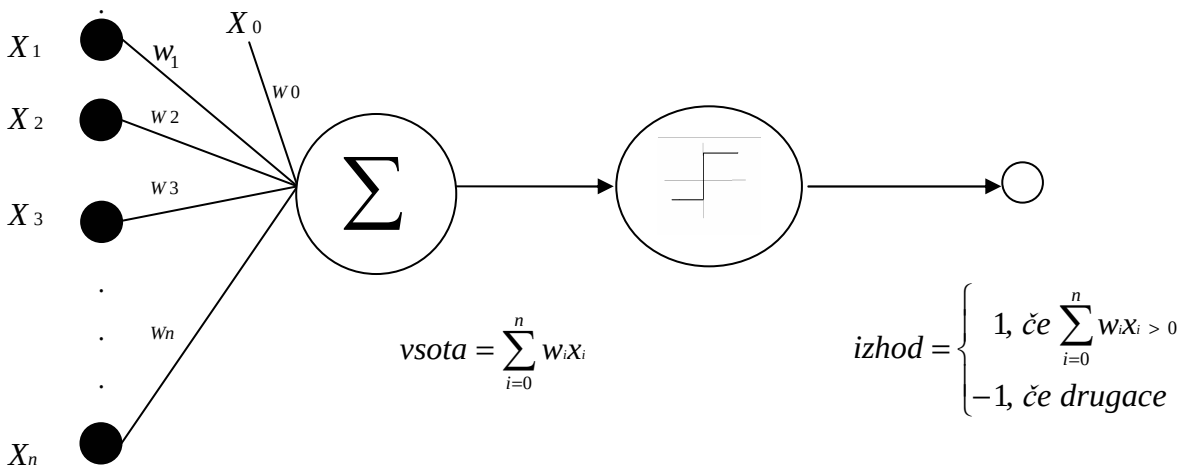
Študija umetnih nevronskih mrež se je deloma zgledovala po bioloških učnih sistemih (možganih), ki so sestavljeni iz zapletenih mrež med seboj povezanih nevronov. Po grobi analogiji so tudi umetne nevronske mreže sestavljene iz med seboj povezanih preprostih računskih enot, kjer vsaka od enot prevzame vhodne vrednosti (po možnosti so to izhodne vrednosti drugih računski enot) ter ustvari eno izhodno vrednost (ki je lahko vhod v druge računske enote). Takemu osnovnemu gradniku umetnih nevronskih mrež po biološki analogiji pravimo nevron.

Najpreprostejši obliki nevrona rečemo perceptron. Gre za procesni element, ki zna izračunati linearno kombinacijo vhodnih vrednosti $x_1,...,x_n$. Največkrat gre za uteženo vsoto, kjer za vsako vhodno vrednost uporabimo utež, ki določi prispevek posamezne vhodne vrednosti $x_i$ k izhodu nevrona. Nato pa z določeno odločitveno funkcijo preslikamo tako dobljeno uteženo vsoto v eno izmed dveh vhodnih vrednosti (glej sliko 4). Uteženi vsoti pravimo tudi funkcija aktivacije, odločitveni funkciji $f$ pa izhodna funkcija. Matematično lahko izhodno funkcijo zapišemo kot:

$$0(x,..., x_n)\begin{cases} 1, & \check{c}e\ w_0 + w_1x_1 + w_2x_2 +,...,+w_nx_n > 0 \\ -1, & drugace \end{cases}, \qquad (4)$$

kjer je vsaka utež $w_i$ konstanta realne vrednosti. Vrednost $w_0$ v bistvu predstavlja prag, ki jo mora utežena vsota vhodnih vrednosti $w_1x_1 + w_2x_2 +,...,+w_nx_n$ preseči, da bo nevron na izhodu sprožil vrednost 1.

**Slika 4: Prikaz perceptrona**

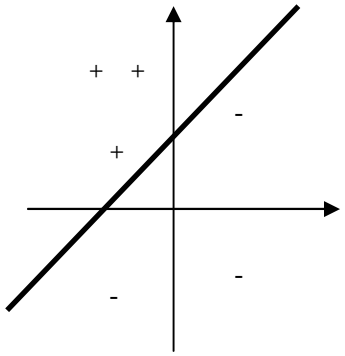



Vir: Mitchell, 1997, str. 87

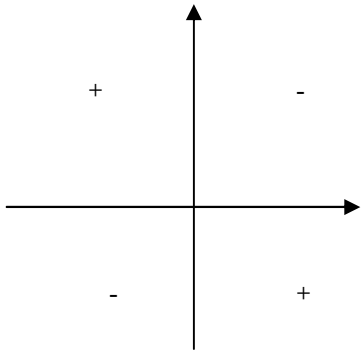
### 2.5.1 Predstavitvena moč perceptrona

Na perceptron lahko gledamo tudi kot na večdimenzionalno odločitveno hiperravnino v koordinatnem sistemu, kjer perceptron sproži pozitiven (+1) signal za primere, ki ležijo na eni strani hiperravnine ter negativen signal (-1) za primere na drugi strani. Odločitvene ploskve na slikah 4a in 4b predstavljajo perceptron z dvema vhodoma $x_1$ in $x_2$. Pozitivni učni primeri so prikazani s plusom, negativni pa z minusom. Slika 5a prikazuje klasifikacijo vhodnih primerkov, ki jih je perceptron zmožen v celoti klasificirati pravilno, slika 5b pa prikazuje vhodne primere, ki jih linearno ni možno ločiti oz. klasificirati.

**Slika 5: Klasifikacija dvorazrednih primerkov s pomočjo hiperravnine**




Slika 5a




Slika 5b

Vir: Mitchel, 1997, str. 87

### 2.5.2 Učenje nevrona

Zgoraj opisani koncept preprostega nevrona predvideva določene uteži, s pomočjo katerih nevron klasificira učne podatke. Postopku določanja natančnih vrednosti posameznih uteži, ki omogočajo, da nevron sproži pravilen pozitiven ali negativen vhod za vsak učni primer, pravimo učenje nevrona.

Vendar pa uteži za klasifikacijo niso kar tako dane, pač pa jih je treba nekako določiti. Eden izmed načinov za določitev vrednosti uteži je ta, da začnemo z naključnimi vrednostmi uteži in jih nato za vsak učni primer ustrezno modificiramo, kadar perceptron klasificira dotični primer. Ta proces se ponavlja iterativno skozi množico učnih primerov toliko časa, kolikor je potrebno, da perceptron pravilno klasificira vse primere.

Uteži so torej pri vsakem koraku spremenijo v skladu z učnim pravilom, ki popravi uteži $w_i$, ki so povezane z vhodnim parametrom xi po formuli:

$$w_i \leftarrow w_i + \Delta w_i \quad \text{kjer je sprememba } w_i \text{ enaka:} \quad \Delta w_i = \eta(t-o)x_i \qquad (6)$$

Pri tem je $t$ ciljna izhodna vrednost, ki jo ima učni primer, $o$ pa je izhodna vrednost, ki jo generira perceptron na osnovi obstoječih uteži. $\eta$ pa je pozitivna konstanta, imenovana stopnja učenja, katere naloga je, da omeji stopnjo spremembe, za katero se spremenijo uteži pri vsakem koraku. Dokazano je, da po opisanem postopku uteži uspešno kovergirajo na tako raven, da lahko perceptron pravilno klasificira vse primerke, kadar so le-ti linearno ločljivi.

### 2.5.3 Pravilo delta

Kadar učni primeri niso linearno ločljivi, torej kadar funkcija, ki se jo mora nevron naučiti, ni linearna, potem ni zagotovila, da bodo uteži konvergirale k vrednostim, ki bodo omogočale nevronu, da bo pravilno klasificiral vse primerke. V tem primeru je uporabno učno pravilo delta, ki v primeru nelinearnosti učnega problema konvergira k najboljši aproksimaciji ciljnega koncepta.

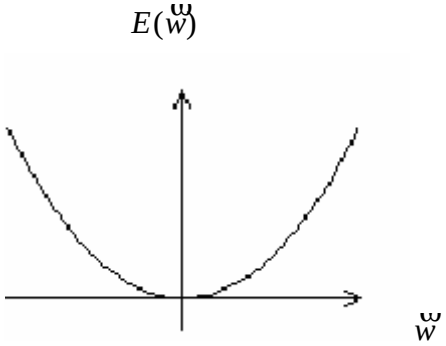
Z namenom korigirati večje izhodne napake hitro, postopoma pa zmanjšati manjše napake proti minimalni vrednosti, se definira funkcija napake. Najpogosteje se uporablja preprosta funkcija, ki povzema kvadrirano vrednost odstopanja dejanske vrednosti od ciljnih (Mitchell, 1997, str. 89):

$$E(\vec{w}) = \sum_{d \in D} (t_d - o_d)^2 \quad , \qquad (7)$$

- kjer $D$ predstavlja množico učnih primerov,
- $t_d$ ciljni izhod, ki ga želimo doseči za posamičen učni primer $d$,
- $o_d$ pa je izhodna vrednost nevrona za učni primer $d$.

$E(\vec{w})$ napaka v odvisnosti od vektorja uteži tako predstavlja kvadrirano odstopanje med ciljno in dejansko izhodno vrednostjo nevrona, skupaj za vse učne primere iz množice. Na ta način lahko okarakteriziramo E kot funkcijo w, saj je izhod nevrona odvisen od utežnostnega vektorja. Funkcija napake da vedno pozitivno vrednost. Cilj pa je najti vrednosti uteži, pri katerih je funkcija napaka E za dano množico učnih primerov najnižja.

**Slika 6: Iskanje minimuma funkcije napake**




Vir: Morton, 2003

Smer nižanja oz. višanja uteži je odvisna od tega, na kateri točki krivulje napake se nahajamo. Smer ugotovimo z odvodom funkcije E po uteži $\frac{\partial E}{\partial w_i}$ oz. za več vhodnih uteženih vrednosti. Za vektor w, sestavljen iz $w_1 + ... + w_n$, moramo torej izračunati odvod za vsako komponento vektorja $\vec{w}$.

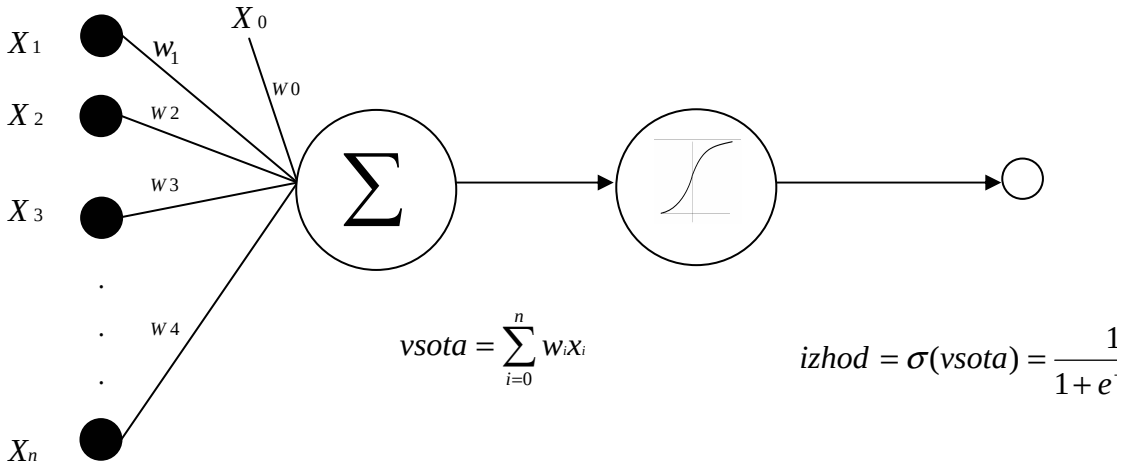
$$\Delta E(\vec{w}) = \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n} \right] \tag{8}$$

Tako dobljeni rezultat $\Delta E(\vec{w})$ je prav tako vektor, ki mu rečemo gradient funkcije E in se zato opisano pravilo učenja nevrona imenuje tudi gradientno pravilo. Tako dobljeni vektor $\Delta E(\vec{w})$ nam kaže smer, po kateri pridemo do minimalne vrednosti napake ob tako izbranih utežeh nevrona. Z odvodom pridobljeni gradient E nam kaže smer največje rasti napake, saj kaže navzgor, kar pa ni problematično, saj moramo samo negirati rezultat, da dobimo obratno smer največjega zmanjšanja napake. Potrebno je vedeti, da se ob večih vhodih nahajamo v večdimenzionalnem prostoru, kar pomeni, da se lahko gibljemo v večih smereh in ne samo levo in desno, kot je to vidno na sliki 6, ko je prikaz omejen na eno samo utež.

### 2.5.4 Sestavljene mreže

Posamezni nevroni so sposobni izraziti samo linearne odločitvene ploskve. Pri kombinacijah nevronov, sestavljenih v nevronske mreže, poskušamo doseči rešljivost nelinearnih problemov, ko so primeri med seboj mešano razporejeni. Vendar pa enostavna povezava perceptronov z linearno izhodno (pragovno) funkcijo ne omogoča rešitve nelinearnih problemov. V teh primerih se pogosto uporabljajo drugačne funkcije, med katerimi najbolj izstopa sigmoidna odločitvena funkcija (Slika 7).

**Slika 7: Pragovna enota s sigmoidno odločitveno funkcijo**




Vir: Mitchell, 1997, str. 87

Sigmoidna funkcija, podobno kot stopničasta pragovna funkcija, omogoča izhod nevrona glede na vhodne spemenljivke, vendar pa je zaradi zglajenosti odredljiva in omogoča izvedbo gradientnega pravila v iskanju najvišje napake napovedi (Slika 8). Njena naloga je, da kot izhodna funkcija nevrona, preslika uteženo vsoto vhodnih parametrov v izhod po sledeči formuli:

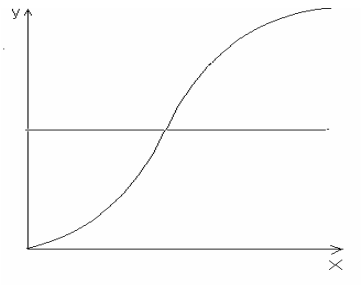
$$o = \sigma(\vec{w} \cdot \vec{x}), \tag{9}$$

kjer je:

$$\sigma(y) = \frac{1}{1+e^{-y}}. \tag{10}$$

Rezultat funkcije se nahaja v razponu med 0 in 1 ter narašča monotono glede na izhod.

**Slika 8: Sigmoidna funkcija**

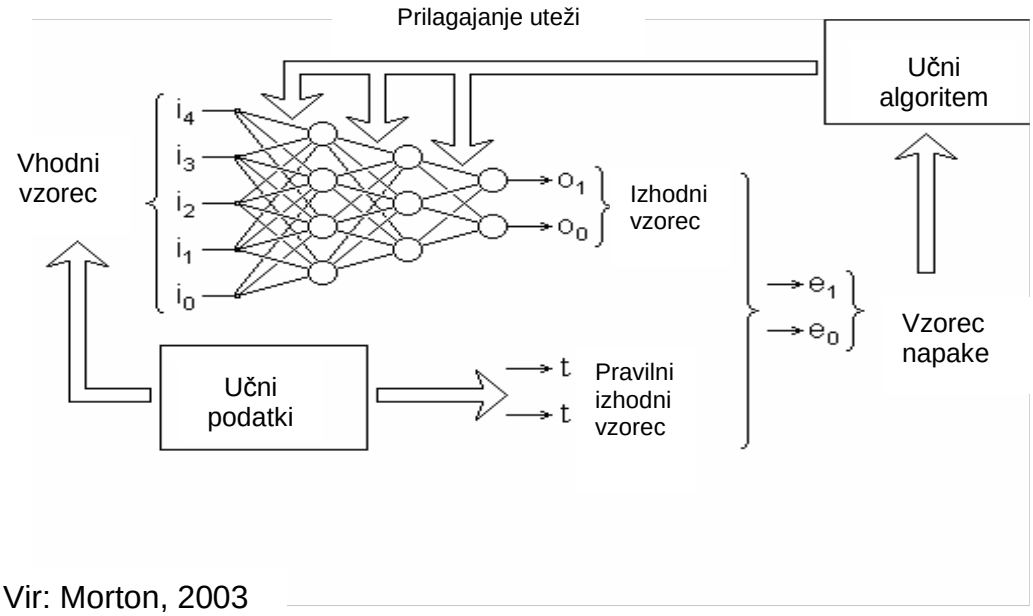



Vir: Mitchell, 1997, str. 96

### 2.5.5 Algoritem vzvratnega širjenja napake

Ko imamo definirano mrežo na zgoraj opisani način, lahko oblikujemo izhodne vrednosti glede na vhodne. Vendar pa je potrebno korigirati nastavljene uteži glede na odstopanja med ciljnimi in dejanskimi vrednostmi. Torej naučiti nevronsko mrežo pravilnih uteži. Podobno kot pri enem samem nevronu moramo poiskati minimum funkcije napake. Ker pa imamo več med seboj povezanih nevronov, potrebujemo sistematičen pristop spreminjanja uteži posameznega nevrona znotraj mreže. Sprememba pri enem nevronu namreč vpliva na celotno mrežo. Postopku sistematične korekcije uteži, s katerimi zmanjšujemo razliko med ciljnimi in dejanskimi izhodnimi vrednostmi nevronske mreže, pravimo »Algoritem vzvratnega širjenja napake« (Slika 9). Algoritem vzvratnega širjenja napake (angl. backpropagation algorithm) se poskuša naučiti pravilnih vrednosti uteži v večnivojski nevronski mreži, torej večjem številu nevronov med seboj povezanih v nivoje, kjer imamo poleg vhodnega in izhodnega sloja nevronov tudi enega ali več “skritih” slojev).

**Slika 9: Prikaz učenja nevronske mreže**

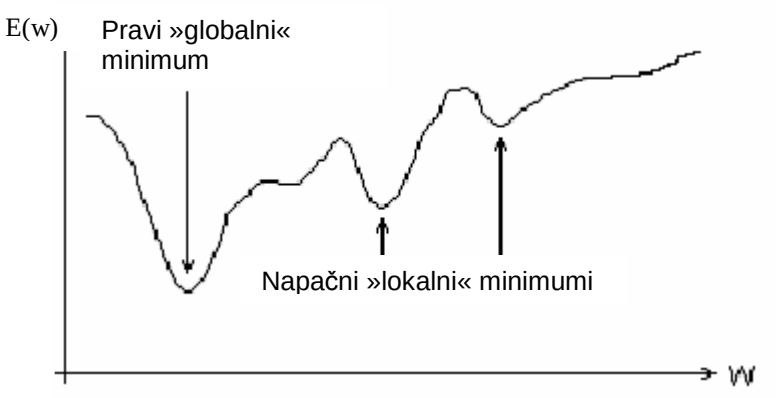



Vir: Morton, 2003

Zahteva pa je, da je med procesom učenja število nevronov in njihovih povezav stalno. Pri tem algoritem vzvratnega širjenja napake uporablja gradientno pravilo, s katerim poskuša zmanjšati kvadrat napake med izhodom celotne mreže ter ciljnimi učnimi vrednostmi. Pomembna razlika v primeru večslojnih mrež je v tem, da ima lahko »funkcija napake«, ki jo poskušamo minimizirati, več lokalnih minimumov v primerjavi s samostojnim nevronom, kjer imamo en sam minimum, ki ga moramo doseči. To pomeni, da lahko v postopku učenja, torej določanja uteži, algoritem minimizira napako, vendar se pri tem znajde v lokalnem minimumu in ne v globalnem

(Slika 10). Za rešitev tega problema ne obstaja metoda, ki bi s stoodstotno verjetnostjo zagotovila absolutni minimum napake.

**Slika 10: Iskanje absolutnega minimuma funkcije napake**




Vir: Morton, 2003

Kljub temu pa obstajajo dokaj dobre hevristične metode, s katerimi lahko omilimo problem lokalnih minimumov:

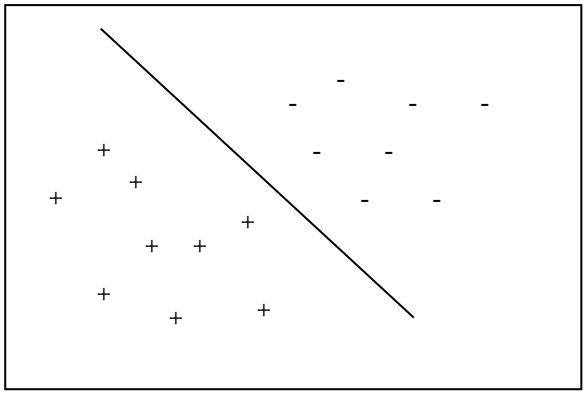
- Dodajanje momenta pri spremembi uteži, kjer pri tekoči spremembi upoštevamo v določeni meri tudi prejšnjo spremembo. S tem učno pravilo pri manjših lokalnih optimumih ne “odneha” takoj, ampak vztraja še nekaj časa v tej smeri, ki so jo določile prejšnje iteracije sprememb in se tako po možnosti izkoplje iz lokalnega minimuma.
- Uporaba stohastičnega gradientnega pravila, ki problem rešuje na ta način, da za vsak učni primer uporablja različno ploskev funkcije napake in se pri tem nasloni na povprečno vrednost funkcij, po katerih išče najnižjo vrednost. Različne ploskve funkcij napake imajo različne lokalne minimume, kar zmanjšuje verjetnost, da se bo proces ustavil v katerem koli od njih.
- Obstaja tudi možnost, da naučimo več različnih mrež z uporabo iste množice učnih podatkov, vendar pa pri tem inicializiramo uteži v mreži z različnimi naključnimi vrednostmi. Če postopki učenja posamezne mreže vodijo do različnih minimumov, lahko izberemo mrežo z najboljšim rezultatom pri preizkusu na preizkusni množici.

Poleg te slabosti nevronskih mrež, da postopek minimiziranja napake ne konvergira vedno k optimalnim vednostim uteži, je problematična tudi izbira topologije mreže, saj je največkrat potrebno empirično določiti število skritih nivojev in število nivojev na vsakem nivoju. Učenje nevronskih mrež je tudi dokaj zamudno, saj se za uspešno iskanje minimuma napake zahteva veliko število prehodov preko učnih primerov. To pomeni, da moramo mreži velikokrat pokazati iste učne primere, tipično nekaj tisoč do nekaj desettisočkrat (Kononenko, 1997, str 232).

## 2.6 Metoda podpornih vektorjev

Metoda podpornih vektorjev SVM (angl. support vector machine) je algoritem, katerega naloga je, podobno kot perceptron, poiskati funkcijo (linearno), ki ločuje dva razreda primerkov (glej sliko 11).

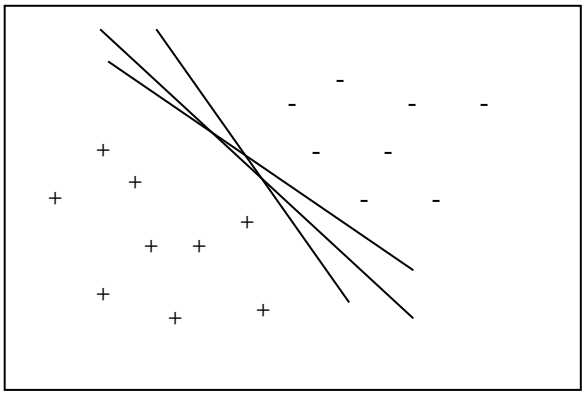
**Slika 11: Ločitev pozitivnih in negativnih primerkov z linearno funkcijo**



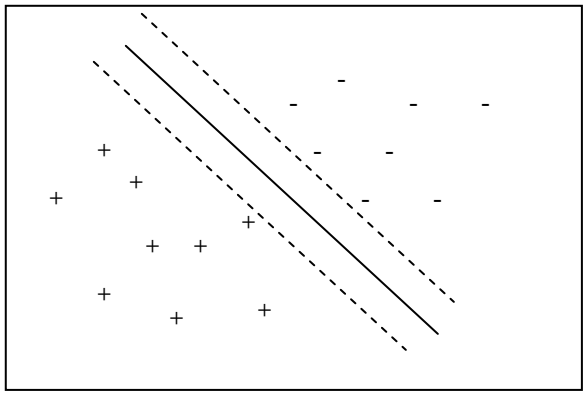
Vir: Lozano-Perez, Kaelbling, 2003

Na splošno lahko pri linearno rešljivih problemih najdemo veliko različnih hiperravnin[2], ki lahko uspešno ločijo dva različna razreda. Perceptron načeloma nima nobenih preferenc glede vseh možnih hiperravnin, ki bi lahko to ločitev uspešno opravile (glej sliko 12a).

**Slika 12: Ločitev primerkov z večimi možnimi hiperravninami in ločitev z ravnino, ki ima maksimalni rob**



Slika 12a



Slika 12 b

Vir: Lozano-Perez, Kaelbling, 2003

[2] V dvodimenzionalnem prostoru linearno obliko funkcije imenujemo črta (linea), v tridimenzionalnem prostoru ravnina, v večdimenzionalnem pa tovrstno funkcijo označujemo kot hiperravnina. Zaradi poenostavljenih dvodimenzionalnih prikazov v tem magistrskem delu so linearne funkcije prikazane kot črte, kljub temu pa se zaradi splošne veljavnosti algoritma v večdimenzionalnem prostoru uporablja ime hiperravnina.

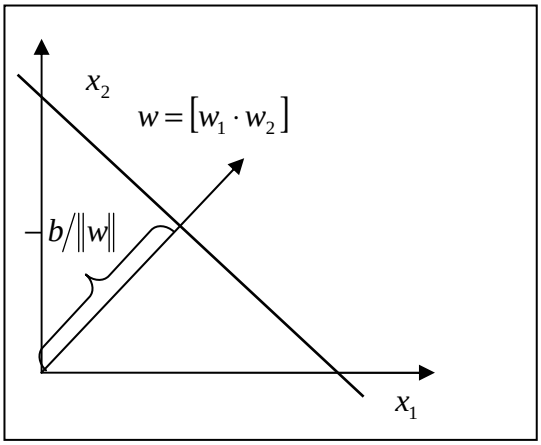
Prednost metode SVM pa je v tem, da omogoča izbiro točno določene rešitve, in sicer izbere tisto ločitveno hiperravnino, pri kateri lahko dosežemo maksimalni rob (angl. Margin), kot to kaže slika 12b (Lozano Perez, Kaelbling, 2003).

Rob je definiran kot razdalja med najbližjima točkama dveh različnih razredov, pravokotno glede na ločitveno linijo. Ob dani ločitveni liniji lahko definiramo klasifikacijsko funkcijo, ki nam da pozitivne vrednosti za primerke na eni strani ločitvene linije in negativne vrednosti za primerke na drugi strani.

$$Napoved(x) = sign(w \cdot x + b) \qquad (11)$$

Klasifikacijska funkcija v linearni obliki, kot je to razvidno iz enačbe 8, zahteva množenje z normalo na hiperravnino z utežnostnim vektorjem $w$ in konstantnim odklonom $b$ in predznakom vrednosti primerka, ki ga poskušamo klasificirati. Grafično je klasifikacijska funkcija prikazana na sliki 13. S prilagajanjem dolžine vektorja $w$ in velikostjo odklona lahko priredimo vrednost klasifikatorja za najbližje pozitivne točke vrednost 1 in več kot ena za ostale pozitivne točke.

**Slika 13: Vektorska predstavitev ločitvene hiperravnine**



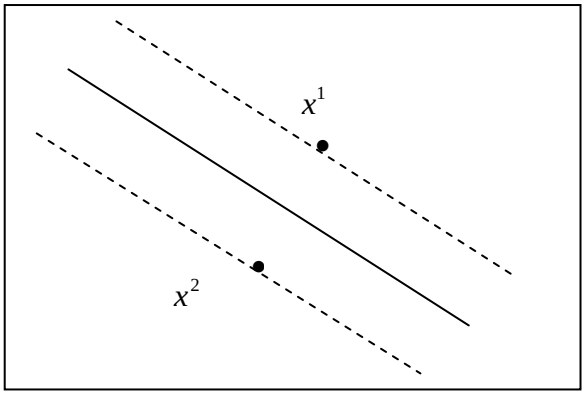

Vir: Lozano-Perez, Kaelbling, 2003

Prav tako se istočasno lahko priredijo vrednosti klasifikatorja za najbližje negativne točke v vrednosti 1 in več kot ena za ostale negativne točke.

$$+1(w \cdot x^1 + b) = 1 \qquad (12)$$
$$-1(w \cdot x^2 + b) = 1$$

Grafično so najbližje (robne) točke prikazane na sliki 14.

**Slika 14: Najbližje točke na robu**




Vir: Lozano-Perez, Kaelbling, 2003

Ko poznamo vrednosti klasifikacijske funkcije za dva najbližja vzorca, dobimo preprosto razmerje med klasifikacijskim vektorjem $w$ in dvema vzorčnima vektorjema $x^1 \text{ in } x^2$.

$$w \cdot (x^1 \cdot x^2) = 2 \qquad (13)$$

Z delitvijo obeh strani enačbe z vrednostjo $w$ dobimo enoto (vektor). Ta enota pomnožena z razliko med vzorčnima vektorjema predstavlja vsoto razdalj pravokotno glede na posamezno vzorčno točko in hiperravnino. Ta vsota razdalj pa je količina, ki jo želimo maksimizirati, pri tem pa ohraniti vrednost klasifikacijske funkcije glede na dva najbližja vzorčna primerka.

Pri tej metodi se zaradi lažjega računanja ta maksimizacija predstavi kot optimizacijski problem, kjer želimo minimirati vrednost $1/2\|w\|^2$ pri prej opisanem pogoju $y_i(w \cdot x_i + b) \geq 1$.

Rešitev tovrstnega problema optimizacije je izvedljiva z vpeljavo Lagrangeovih multiplikatorjev, kjer nam rešitev da želeno normalo hiperravnine. Klasifikacijska funkcija pri tem ni odvisna od vseh točk (vektorjev) v prostoru, pač pa le od tistih, ki so blizu razmejitvene ravnine, jo z obeh strani podpirajo, torej podpornih vektorjev (Lozano-Perez, Kaelbling, 2003).

## 2.7 Kriteriji za merjenje uspešnosti strojnega učenja

Kreiranje približka neke funkcije se ustvari na podlagi množice primerov pojava kot primarne informacije. Funkcijski približki – modeli imajo lahko različne oblike: algebraični ali triognometrični polinomi, nevronske mreže, sestavljene stopničaste linearne regresijske funkcije itd. Mnoge od naštetih tehnik opisovanja pojavov imajo

skupno značilnost v tem, da znajo univerzalno posplošiti oz. se približati kateri koli funkcijski obliki, če je seveda zvezna na intervalu, ki ga preučujemo. Seveda pa gre za teoretične rezultate, ki zahtevajo različno kompleksne modele za doseganje konvergence. Približevanje pravemu stanju, torej originalni funkciji, zahteva neskončno število učnih primerov brez »šuma«. Ti pogoji so v praksi redkokdaj povsem izpolnjeni.

Če hočemo oceniti neko neznano funkcijo $f(p)$, ki nam da neko končno množico primerov $P(P,T)$, se lahko lotimo induktivnega učenja na sledeč način (Hellstrom, 1998):

1. izberemo naključno podmnožico naključnih primerov iz množice vseh primerov,
2. zgradimo model $h_i(P)$ (npr. nevronska mreža),
3. ocenimo model na naključni testni množici primerov.

Če večkrat izvedemo zgoraj opisani postopek, vsakič dobimo funkcijo, ki vsakokrat znova opisuje različno podmnožico učnih podatkov. Tako dobimo hipotetične funkcije $h_{1,}.....,h_n$.

Sedaj lahko pogledamo, kako se vrednosti hipotetičnih funkcij med seboj razlikujejo v točki $p$. Ker so vse funkcije od $h_1(p),....,h_n(p)$ odvisne od naključne izbire učnih podatkov, lahko na njih gledamo kot na izide naključne spremenljivke $h_p$ z aritmetično sredino $m_p$ in varianco $V_p$. Srednja vrednost kvadrata napake $E_p$ je pri tem odvisna od dveh komponent, in sicer (Hellstrom, 1998):

- točnost: $\left(m_p - f(p)\right)^2$ vrednost, za katero povprečni model odstopa od prave vrednosti,
- natančnost: (varianca $Vp$ ) variabilnost med različnimi modeli.

Tukaj je pomembna kompleksnost modela, s katerim poskušamo ocenjevati. Porazdelitev je namreč v povezavi s tem, kjer lahko dobimo dva ekstrema:

- Prenizka kompleksnost modela (npr. linearni model ali pa nevronska mreža z majhnim številom uteži in nevronov). Izračunani modeli $h_1(p),....,h_n(p)$ napovejo približno iste vrednosti, ker preprosti model nima izračunane moči v tej meri, da bi izrazil manjše razlike med različnimi učnimi primeri. Tako imamo visoko točnost predvidevanja, pri čemer je varianca nizka. Ravno tako pa imamo tudi nenatančne modele predvidevanja, kjer se vsi modeli razlikujejo v isti meri od prave vrednosti $f(p)$.

- Preveč podroben model (nevronska mreža z večimi skritimi plastmi). Izračunani modeli $h_1(p),....,h_n(p)$ se prilagodijo točno na vsako učno množico primerov in s tem ustvarijo visoko varianco $V_p$ oz. so vrednosti precej razpršene, saj vsak model operira z različno podmnožico učne množice primerov. Temu rečemo, da imajo kompleksnejši modeli nizko natančnost, vendar pa so modeli precej točni, saj je srednja vrednost zelo blizu prave vrednosti $f(p)$.

V realni situaciji, kjer imamo samo eno oceno $h(P)$, to še vedno lahko štejemo kot rezultat naključne spremenljivke $h(P)$ ter z njo povezani aritmetično sredino in varianco. Navadno moramo izbirati med kombinacijo, ki ima nizko natančnost in visoko točnost ali pa natančno, vendar netočno oceno.

Kadar se znajdemo v problematični situaciji, je vedno potrebno poiskati določen kompromis med natančnostjo in točnostjo. Nizka natančnost je potrebna, da dosežemo visoko točnost in obratno. V analizah se statistiki v takih situacijah raje odločajo za nižjo natančnost ter večjo točnost. Tovrstne odločitve so posledica dokaj preprostih modelov z relativno majhnim številom parametrov. Ti modeli so primerni, kadar gre za primerljivo preproste primerjave, ki jih modeliramo, in je tisto obnašanje pojava, ki nas zanima, v večji meri posledica zunanjih vplivov na sam pojav in ne interno obnašanje pojava. V primeru zapletenejših pojavov pa je potrebno uporabiti ustrezno kompleksnejše modele, da bi s tem dobili sprejemljivo nizka odstopanja od napovedi. Cena, ki jo moramo plačati za dodatno izraznost modela, pa je že omenjena višja varianca te naključne spremenljivke, kot lahko zgrajenemu modelu rečemo, oz. nižja natančnost modela, ko gre za opisovanje pojava kot celote. V nekaterih primerih lahko zgradimo relativno preprost model z uporabo ustreznega predznanja o določenem pojavu. V takih primerih je »netočnost« neškodljiva, saj je model usmerjen k pravi funkciji f(P). Točnost se da kasneje povečati, ne da bi pri tem zmanjševali natančnost modela.

### 2.7.1 Prilagajanje modela

Kadar je model preveč prilagojen (angl. overfitted) izbrani množici vrednosti pojava, lahko rečemo, da je tip modela preveč podroben v skladu z zgoraj opisanim kompromisom med točnostjo in natančnostjo. Ravno obratno pa je pri premalo preprostem modelu, ki je preveč splošen in nenatančno opisuje pojav. Kadar je izbrana množica vrednosti pojava učna množica, rečemo, da je model preveč naučen (angl. overtrained) (Witten, Eibe, 2000, str. 38).

Kadar je model preveč prilagojen učni množici podatkov, govorimo o prenaučenosti modela, kar ni preveč koristno, saj model izgubi zmožnost generalizacije pri napovedovanju pojava. Problem običajno nastane, kadar dovolimo učnem algoritmu, da izvaja iteracije tako dolgo, dokler ne doseže minimalnega odstopanja od učnih podatkov. Ko napaka napovedi na modelu, zgrajenem iz učnih podatkov, doseže točko pod mejo, ki se nanaša na lastnosti problema v proučevanem pojavu, se algoritem začne prilagajati lastnostim, ki niso splošno pomembne za pojav, pač pa jih v smislu pojava lahko obravnavamo kot šum (Hellstrom, 1998). Sposobnost generalizacije modela se nato izgubi in napake napovedi na podatkih, ki v učnem procesu niso bili predstavljeni algoritmu, se povečajo. Ta fenomen se načeloma pojavi pri šibkem modeliranju, kjer gre za modele, ki so zelo splošni in sposobni prilagajanja širokemu spektru podatkovnih variacij. Prav tako je pomembna kompleksnost modela, od katere je odvisna njegova izraznost. Relativno preprost model (npr. premica) je manj občutljiv na probleme prenaučenosti. Tudi same lastnosti oz. procesi, ki jih vsebujejo prenaučeni oz. učni podatki lahko povzročajo probleme ob predolgem prilagajanju modela tem podatkom. Podatki, v katerih je delež šuma neuporabnih nerealnih vrednosti relativno visok, lahko povečajo tveganje, da bo prišlo do pretiranega prilagajanja modela, saj algoritem ob učenju šum v podatkih interpretira kot prave pomembne vrednosti. Tudi količina podatkov pri učnem procesu ni nepomembna, saj več kot je učnih podatkov, manjša je možnost, da bo model določene kompleksnosti interpretiral šum v podatkih kot izvirne podatke. Problem pretiranega prilagajanja učnim podatkom se lahko reši le s posredno uporabo preprostejših modelov ali pa neposredno s posegom v učni algoritem, kot je uravnavanje odločitvenih dreves, prednastavitev uteži itd.

### 2.7.2 Splošnost modela

Pri uporabi algoritmov za strojno učenje je bistvenega pomena možnost čim bolj točnega in natančnega napovedovanja na množici testnih podatkov, ki v učnem postopku še niso bili predstavljeni algoritmu. Razlika v uspešnosti napovedovanja med učnimi podatki, na katerih je algoritem zgradil model, ter preizkušenimi podatki se bistveno poveča z uporabo šibkih modelov, kot so nevronske mreže, ki v svoji zasnovi ne vsebujejo veliko vnaprejšnjih predpostavk glede pojavov, ki naj bi jih pojasnjevali (Hellstrom, 1998). Njihova sposobnost generalizacije je tako bistveno večja od močnih modelov, torej modelov z vnaprejšnjimi prilagoditvami glede na podatke. S tem je sposobnost generalizacije pomembna ne samo kot mera učinkovitosti napovedovanja, pač pa tudi kot kriterij za izbiro tipa modela. Obstajajo številne metode za oceno splošnosti napovedovanja modela, ki so v nadaljevanju tudi podrobneje opisane.

### 2.7.3 Preizkus na testnih podatkih

Standardni postopek preizkušanja modela pri večini tehnik strojnega učenja je razdelitev podatkov, ki jih imamo na voljo, v dve skupini, in sicer na učno ter na testno skupino (Witten, Eibe, 2000, str. 56). Včasih se učno množico podatkov razdeli še naprej na manjše množice (za določanje točke, ko je potrebno algoritem ustaviti, da se izognemo pretiranemu prilagajanju podatkom). Testne možice se uporabi samo za končno oceno učinkovitosti napovedovanja, ki je največkrat neposredno odvisen od sposobnosti generalizacije modela.

Ta pristop je zelo potraten, kar se tiče podatkov, saj niso vsi podatki, ki so v danem trenutku na voljo, uporabljeni za učenje oz. izgradnjo modela. Prednost pa je v tem, da nimamo nobenih predsodkov oziroma pričakovanj glede doseganja minimalnih odstopanj od bodočih vrednosti, ki naj bi jih model dosegel, ko mu predstavimo bodoče podatke.

### 2.7.4 Križno preverjanje (angl. cross-validation)

Koncept križnega preverjanja je oblika preizkusa na testnih podatkih. Če celotna množica podatkov vsebuje $N$ podatkovnih primerov, izločimo vsakič en podatkovni primer ter uporabimo preostale za učenje modela. Učinkovitost modela je ocenjena s kvadratom napake izločenega podatkovnega primera. Postopek nato ponovimo za vseh $N$ podatkovnih primerov v množici podatkov, kar nam da $N$ modelov ter z njimi povezane ocene kvadratne napake vsakokratno izločenih podatkovnih primerov. Aritmetična sredina teh ocen nam da skupno oceno napake predvidevanja za model (Witten, Eibe, 2000, str 65).

Seveda je opisana metoda, kadar gre za veliko število podatkovnih primerov (velik $N$) računsko zelo potratna, zato so možne tudi variante, ko je namesto enega izločenega podatkovnega primera izločenih več primerov. Metoda, razvita posebej za nevronske mreže v postopku učenja, predvideva namesto vsakokratnega ponovnega učenja od začetka pri vsaki novi učni množici izmed $N$ množic ohranjanje uteži iz prejšnjega postopka kot začetne vrednosti za naslednji model.

Vidimo, da nam dosežki na področju strojnega učenja lahko ponudijo različna orodja s katerimi lahko povzamemo informacije v izbranih podatkih, ter kako lahko kvaliteto naučenega znanje ocenimo. Seveda je od danih podatkov in njihove »priprave« odvisno, kako učinkovito se lahko iz njih naučimo informacije vsebovane v teh podatkih. Če želimo uporabiti strojno učenje za napovedovanje gibanj na kapitalskem trgu pa je ravno izbira podatkov ter njihova priprava tisto, kar zahteva zelo dobro poznavanje kapitalskega trga.

# 3 GIBANJE CEN DELNIC NA BORZI

Če pogledamo časovne vrste cen oz. tečajev delnic, opazimo kar nekaj značilnosti, pri čemer vsaka posebej ne predstavlja nekaj nepremostljivega, v kombinaciji pa predstavljajo težavno situacijo, ko poskušamo predvideti bodoča tečajna gibanja. Proces gibanja tečajev se obnaša zelo podobno kot proces naključnega sprehoda oz. cene na trgu se spreminjajo skoraj naključno. Verjetnost, da se v določenem dnevu cene vrednostnih papirjev zvišajo, je prav tako enaka verjetnosti, da se znižajo, ne glede na gibanja v preteklosti. Tudi avtokorelacija med dnevnimi spremembami je načeloma zelo nizka, kar pomeni, da sprememba tečaja danes nima velikega vpliva na to, kako se bo spremenila cena delnice jutri. Kapitalski trgi skozi čas prehajajo iz mirnih v bolj turbulentna obdobja in obratno. Raven šuma v podatkih ter njegova nestanovitnost se močno spreminja med posameznimi obdobji. (Hellstrom, 1998, str. 3). Vse to povzroča velike težave pri napovedovanju časovnih vrst, kadar uporabljamo tradicionalne postopke analize.

Tovrstna naključna gibanja tečajev delnic sicer kažejo, da trg dobro deluje oz. je zelo učinkovit, vendar pa je v takih razmerah ugotavljanje ponavljajočih se vzorcev gibanj tečajev delnic praktično nemogoče, s tem pa je onemogočeno napovedovanje cen delnic v prihodnosti. To povzroča težave vlagateljem, ki želijo doseči čim večji in zanesljivejši donos na vložena sredstva na trgu. Podrobnejša preučitev hipoteze učinkovitih trgov (angl. efficient market hypothesis, EMH) je zato ključna za ugotovitev, ali je predvidljivost cenovnih in drugih sprememb na nekem trgu sploh možna.

## 3.1 Hipoteza učinkovitih trgov

Analize časovnih vrst gibanja vrednosti delnic na kapitalskih trgih največkrat pokažejo, da vrednosti delnic sledijo gibanju naključnih sprememb, v literaturi definirane kot naključni sprehod. Gre za koncept, ki naj bi imel analogijo s pijancem, ki se opoteka in katerega spremembe poti levo-desno so povsem nepredvidljive. Koncept se je uporabljal že v šestnajstem stoletju kot model za verjetnostni račun, definiran pa je kot:

$$y(t) = y(t-1) + \alpha(t), \qquad (14)$$

kjer $y(t)$ predstavlja vrednost časovne vrste v danem trenutku $t$, $\alpha$ pa je naključna napaka, katere aritmetična sredina ima vrednost 0 in katere vrednosti so med seboj neodvisne. Sprememba $\Delta y(t) = y(t) - y(t-1)$ je zato enaka $\alpha(t)$ in je neodvisna od

predhodnih sprememb $\Delta y(t-1), \Delta y(t-2), \ldots$ Tudi druge značilnosti $\Delta y$, kot je npr. varianca, so med seboj neodvisne (Hellstrom, 1998).

Naključne spremembe cen na trgu niso posledica neracionalnega obnašanja tržnih udeležencev, pač pa ravno nasprotno kažejo na **zelo učinkovit trg,** kjer cene odražajo vse informacije, ki so v danem trenutku na voljo. Takoj, ko je na voljo kakršna koli informacija, ki bi nakazovala, da je določena delnica na trgu podcenjena in s tem omogoča ustvariti določen dobiček pri transakciji, se vlagatelji množično odločajo za nakup te delnice, kar povzroči, da se cena delnice zaradi večjega povpraševanja zviša na pravilno raven, na kateri je možno ustvariti običajne mere donosa. Z običajna mero donosa imamo v mislih tisto, ki je primerljiva s stopnjo tveganja za to delnico (Mramor, 2000, str. 315).

Če se cene delnic na trgu oblikujejo glede na informacije, dane v sedanjem trenutku, se njihove cene spreminjajo le ob nastanku novih informacij, ki dosegajo trg. Nove informacije morajo biti nepredvidljive, kajti če bi jih bilo možno predvideti, bi bilo to predvidevanje vsebovano v informacijah, danem v sedanjem trenutku. Tako se cene delnic, ki se spreminjajo glede na nove (nepredvidljive) informacije, prav tako spreminjajo nepredvidljivo.

To je osnovni razlog, da gibanje cen delnic na trgu sledi konceptu naključnega hoda, torej naključno in nepredvidljivo, kar kaže na to, da je trg vrednostnih papirjev učinkovit.

Glede na to, kaj mislimo z besedno zvezo »vse razpoložljive informacije«, ločimo **tri oblike hipoteze učinkovitosti trga** (Mramor, 2000, str. 315):

- **Šibka oblika** hipoteze upošteva samo podatke o preteklih cenah. Ta oblika hipoteze izključuje kakršno koli možnost predvidevanja in upošteva pretekle cenovne podatke, saj so cenovne spremembe povsem naključne. To pomeni, da med zaporednimi spremembami cen ni korelacijskih odvisnosti. Posledica tega je, da uporaba tehnične analize ne more voditi do realiziranja nadpovprečnih donosnosti. S tehnično analizo je mišljeno iskanje ponavljajočih se vzorcev v gibanju cen. Čeprav obstajajo taki vzorci, so se jih vsi investitorji naučili izrabiti.

- **Srednje močna** oblika učinkovitega trga upošteva vse javno dostopne informacije. To vključuje poleg cenovnih še druge informacije o trgovanju, kot je obseg trgovanja (npr. število prodanih delnic) ter podatke, uporabljene za temeljno analizo, kot so pričakovana višina prodaje in prihodnjih dobičkov. Posledica tega je, da s pomočjo temeljne analize ni moč doseči

nadpovprečnih donosnosti, saj je pravilna t. i. “notranja” vrednost delnic že enaka trenutni ceni.

- **Močna oblika** učinkovitosti upošteva vse javne in privatne informacije.

Hipoteza o učinkovitem trgu v svojem izhodišču trdi, da zastarele (razpoložljive) informacije ne ponujajo nikakršnih vrednosti za doseganje dobičkov. Dobički so sicer možni, a doseganje le-teh zahteva določeno tveganje in dobički so torej le poštena kompenzacija za izpostavljanje tveganju. S tega stališča je tako priporočljiva pasivna oblika naložbenja, recimo v obliki dovolj diverzificiranega portfelja vrednostnih papirjev (angl. buy & hold strategy). S tem bo dosežena pravična mera donosa glede na zmanjšano raven tveganja, vlagatelj pa se s tem izogne transakcijskim stroškom – provizijam.

Ni presenetljivo, da hipoteza učinkovitih trgov ne vzbuja ravno veliko navdušenja med poklicnimi upravljavci delniških skladov, saj trdi, da je iskanje podcenjenih delnic s ciljem ustvarjanja dobička bolj ali manj stran vržen čas in denar.

Veljavnost šibke in srednje močne učinkovitosti trga je bila potrjena v številnih študijah in raziskavah in je s tem predstavljala prevladujoče stališče v akademski sferi. Kmalu so se pojavile različne teorije in empirični preizkusi, ki so poskušali izpodbijati hipotezo o učinkovitih trgih.

## 3.2. Nepopolna učinkovitost kapitalskega trga

Empirične študije v financah ponujajo vse več dokazov, ki govorijo proti hipotezi učinkovitih trgov. Močna oblika hipoteze je zaradi pomanjkanja dostopnih tako javnih kot privatnih informacij težko statistično preverljiva, zato so zgodnje študije učinkovitosti trgov testirale predvsem šibko obliko hipoteze, in sicer, ali lahko špekulanti izluščijo določena trendovska gibanja v podatkih o gibanju cen delnic. Eden izmed načinov, kako izločiti trendovska gibanja v časovnih vrstah, je merjenje avtokorelacije tržnih donosov. Avtokorelacija se nanaša na težnjo sedanjih donosov k povezavi s preteklimi donosi posamezne delnice. Izražena pozitivna avtokorelacija pomeni, da pozitivnim donosom posamezne delnice sledijo negativni donosi, negativna avtokorelacija pa izraža negativne donose, ki sledijo pozitivnim (angl. reversal). Tudi za slovenski trg kapitala je bilo izvedenih nekaj študij, med katerimi bi omenil predvsem delo Deželanove (1996), kjer je empirično testiranje temeljilo predvsem na preučevanju avtokorelacije finančne časovne vrste podatkov za obdobje od januarja 1994 do junija 1996. Tako v tej, kot tudi v podobni študiji za obdobje 1996–1999 (Mramor, 2000, str. 301) so raziskovalci prišli do zaključka, da

za slovenski trg kapitala ne morejo potrditi hipoteze učinkovitosti trga delnic niti v svoji šibki obliki.

Kljub tradicionalni podpori hipoteze učinkovitih trgov, ki je vladala v akademskih krogih, je veliko tržnih udeležencev, ki verjamejo, da lahko predvidijo cenovna gibanja, tako da lahko dosežejo dobiček. Večina poklicnih borznih trgovcev se namreč do določene mere zanaša na bodisi tehnično bodisi temeljno analizo preteklih informacij. Kupujejo, ko na trgu vlada »bikov trend«, in prodajajo ob »medvedjem trendu«, tako privzamejo direktno korelacijo med trenutnim trendom in bodočimi cenami.

Teoretične razlage, zakaj trgi ne delujejo povsem učinkovito, temeljijo predvsem na izpodbijanju trditve o racionalnem obnašanju vlagateljev, na kateri gradi hipoteza učinkovitih trgov. Obnašanje posameznika je zelo nepredvidljivo in ga je nemogoče opisati ali predvideti. Pri obnašanju udeležencev trga pa je zanimivo to, da se posamezne skupine udeležencev obnašajo na podoben način. Pri obnašanju takih množic pa gre za bistveno primitivnejše vzorce obnašanja, ki jih je kot take moč opazovati, opisati in delno tudi napovedovati.

Neracionalne reakcije vlagateljev najočitneje predstavljajo sledeči primeri (Bodie, 1999, str 367):

- **Napake pri napovedovanju.** Vrsta eksperimentov je pokazala, da ljudje zaradi nedavnih izkušenj hitro spremenijo svoja pretekla prepričanja, tisto, v kar so verjeli pred tem, kadar gre za napovedovanje dogodkov v prihodnosti. To se kaže na primeru vrednotenja delnic podjetij, za katera se pričakuje visok donos, zato ljudje pričakujejo še višjega. Nakar se ob spoznanju, da te delnice ne bodo prinesle tako visokega donosa, njihova vrednost na trgu zaradi povečane prodaje prekomerno zniža pod njihovo »realno« vrednost.

- **Preveliko samozaupanje.** Ljudje dostikrat precenijo svoje sposobnosti, tako tudi sposobnosti razumevanja določenih procesov in gibanj na trgu kapitala ter napovedovanje le-teh. V neki študiji na primer je 90 odstotkov šoferjev na Švedskem označilo svoje šoferske sposobnosti kot nadpovprečne. Tovrstno preveliko zaupanje v svoje sposobnosti bi lahko povzročilo prevlado aktivne nad pasivno obliko naložbenja, kar kaže na to, da se ljudje ne sprijaznijo kar tako s hipotezo o učinkovitih trgih in poskušajo s svojo lastno strategijo doseči dobiček na borzi. Kljub povečanju naložb v indeksirane portfelje delnic v zadnjih letih, torej v naložbe skladov, katerih portfelj se oblikuje skladno kakšnim od borznih indeksov, je manj kot deset odstotkov premoženja vzajemnih skladov v ZDA naloženih v tovrstnih indeksiranih portfeljih.

- **Izogibanje obžalovanju.** Psihologi so ugotovili, da ljudje, ki pri odločitvi izberejo neobičajno izbiro in se le-ta izkaže za napačno, to veliko bolj obžalujejo, kot če bi se napaka zgodila ob običajni izbiri. Primer nakupa delnice enega od Blue chipov, zaradi katerega je ustvarjena izguba (negativen donos), ni tako boleča kot izguba pri nakupu ene od delnic neznanega novoustanovljenega podjetja.

- **Predstavitev problema.** Odločitve posameznikov so dostikrat odvisne tudi od tega, kako je posamezniku problem predstavljen. Posameznik lahko zavrne ponujeno rešitev, če je verjetnost uspeha predstavljena v odstotkih, da ne bo uspeha, in sprejme to isto rešitev, če je verjetnost izkazana v odstotkih, da bo ta rešitev privedla do uspeha. Tako se lahko posamezni investitor s precejšnjo mero tveganja odloča glede ene delnice, na drugi strani pa ne bo tvegal z vrednostnimi papirji, ki jih je namenil za prihodnost svojih otrok. Tovrstno obnašanje naj bi povzročilo tudi preferenčne odločitve posameznih vlagateljev glede nakupa delnic, katerih dividende so letno višje, kot pa tistim z nižjimi dividendami, oziroma tistim, katerih dividende se sploh ne izplačujejo. Vlagatelji se tako raje odločijo trošiti dohodke od premoženja, kakor pa »načenjati« svoj kapital s prodajo nekaj delnic, kar pa bi lahko prav tako pripeljalo do istega dohodka. Tovrstno obnašanje povzroča tudi fenomen, ko vlagatelj nikakor noče prodati delnic, katerih vrednost pada, in si priznati izgubo, upajoč da se bo trend vsak čas obrnil.

Pomembna postavka v kritikah hipoteze učinkovitih trgov ima podlago v tem, da veliko tržnih udeležencev vrednoti več faktorjev in ne samo ceno, za katero lahko kupijo ali prodajo delnico (Hellstrom, 1998). Najbolj izstopajoč primer je tveganje. Ker različni udeleženci različno vrednotijo nivo tveganja, je povsem mogoče, da en udeleženec vidi določeno ceno delnice kot lepo nakupno priložnost, medtem ko drugi isto ceno vidi kot dobro prodajno priložnost. Drugi primer je različna likvidnost, ki jo različni tržni udeleženci potrebujejo. Nekateri udeleženci so pripravljeni plačati višjo ceno v zameno za to, da je prodaja takoj izvršena. Pa vendar za nobenega od teh udeležencev ne moremo trditi, da deluje neracionalno. Preprosto uporabljajo drugačne funkcije koristnosti, ko “preračunavajo” vrednosti različnih rezultatov v različnih razmerah.

Upoštevamo lahko tudi argument proti hipotezi, ki opredeljuje različne časovne odzive tržnih udeležencev pri poslovanju. Na primer, večina malih delničarjev reagira različno v primerjavi s poklicnim trgovcem, ko cena delnice nenadoma pade (Mramor, 1993, str. 142). Te razlike v časovni reakciji povzročajo anomalije v tržnih cenah, tudi če nobene nove informacije ne dosežejo trga. Možno bi bilo identificirati tovrstne situacije in na nek način predvideti prihodnje spremembe.

## 3.3 Tržne anomalije in dokazi o predvidljivosti

Namesto vedenjskih analiz posameznikov za dokazovanje, ali hipoteza učinkovitih trgov drži ali ne, je veliko raziskovalcev začelo statistično analizirati delovanje trga kapitala in ugotavljati odstopanja, ki bi bila lahko uporabna za doseganje presežnih dobičkov. Tovrstne anomalije so lahko neposreden dokaz za neveljavnost hipoteze učinkovitih trgov. Empirično so ugotovljeni naslednji učinki oz. anomalije (Mramor, 2000, str. 314):

- **Učinek majhnega podjetja.** Ta učinek je zaznaven, ko opazujemo podjetja z relativno nizko kapitalizacijo skozi daljše časovno obdobje, ki dajejo višje stopnje donosa kot večja podjetja. To naj bi podrazumevalo višje stopnje tveganja, povezane z manjšimi podjetji, vendar pa naj bi kljub upoštevanju večjega tveganja donos pri manjših podjetjih presegal tistega pri večjih. Še posebej naj bi to bilo očitno v januarju. Razlika med donosnostjo delnic majhnih in velikih podjetij v ZDA je januarja v obdobju 1936 – 1979 v povprečju znašala 714 odstotnih točk na dan. Ta koledarski učinek majhnih podjetij poskušajo razložiti s prodajo delnic decembra zaradi davčnih dolgov, ki jih investitorji pokupijo nazaj v januarju. Prav tako naj bi o majhnih podjetjih krožilo razmeroma malo informacij, saj naj bi jih institucionalni investitorji bolj zapostavljali. To in manjša likvidnost zapostavljenih podjetij naj bi bila vzroka za višjo donosnost delnic takih podjetij, ki bi lahko pomenila eno vrsto premije za tveganje.

- **Razmerje med tržno in knjigovodsko ceno delnice (angl. market-to-book ratio)**. Ugotovljeno je bilo, da je omenjeno razmerje dober kazalec prihodnje donosnosti delnic. Donosnost delnic podjetij z nižjim razmerjem je višja od donosnosti delnic podjetij z višjim razmerjem.

- **Učinek preobratov.** V raziskavah se je pokazalo, da delnice, ki so se v določenem časovnem obdobju slabo odrezale, dosegajo v naslednjem obdobju značilno višje donosnosti. Ravno nasprotno se zgodi z delnicami, ki v določenem obdobju dosegajo nadpovprečne rezultate. Videti je, da se trg delnic odziva na informacije preveč impulzivno.

- **Učinek nizkega multiplikatorja čistega dobička (angl. price earning ratio P/E).** Učinek nizkega P/E temelji na študijah, ki kažejo, da premoženje, ki vsebuje delnice z nizkim koeficientom med ceno, ki jo plačamo za nakup vrednostnega papirja ter čistim dobičkom na delnico, prekaša v donosnosti tista premoženja (portfelje), ki vsebujejo vrednostne papirje z visokim koeficientom. Razlaga za ta efekt naj bi temeljila na tem, da se vrednostni papirji trgujejo pri nizkih P/E, ker začasno niso priljubljeni pri udeležencih na trgu. Družbe, ki trenutno niso priljubljene, pa bodo priljubljene v prihodnosti.

- **Medsebojne odvisnosti.** V raziskavah so bile potrjene pozitivne ugotovitve glede avtokovarianc med različnimi vrednostnimi papirji. Ti učinki, ki potrjujejo povezavo med gibanji cen posameznih delnic, se kažejo s pozitivnim predznakom ter določenim strukturiranim časovnim zamikom. Izkaže se, da donosnosti velikih podjetij sledi donosnost manjših podjetij.

Opisane anomalije, ugotovljene z empiričnimi testi ter rezultati analiz obnašanja množic udeležencev na kapitalskem trgu, pravzaprav predstavljajo nepopolno učinkovitost kapitalskega trga in s tem ponujajo možnost za doseganje dobička. Saj v primeru, da nam uspe dovolj dobro izločiti določen pojav iz podatkov o cenovnih in drugih gibanjih vrednostnih papirjev na kapitalskem trgu, lahko na tej podlagi z »znosnim« tveganjem le ustvarimo določen dobiček.

## 3.4 Metode za napovedovanje gibanja cen delnic

Ne glede na to, ali je hipoteza o učinkovitih trgih sprejeta ali ne, borzni analitiki uporabljajo različne metode za analiziranje dogajanj na kapitalskem trgu, s katerimi poskušajo zaslužiti pri trgovanju z vrednostnimi papirji. V tem podpoglavju so poleg temeljne analize predstavljene predvsem tiste, za katere je potrebna določena matematična oz. statistična analiza. Tukaj bi uvrstil tehnično analizo delnic s svojim instrumentarijem grafikonov in indikatorjev, kot tudi analizo časovnih vrst, saj cenovna gibanja delnic najenostavneje povzamemo v obliki časovne vrste. Še posebej bi izpostavil najnovejše metode, ki si čedalje bolj utirajo pot na analitsko prizorišče; to so metode, ki pri analizi uporabljajo strojno učenje. Tukaj so najbolj zastopane predvsem nevronske mreže in metode najbližjih sosedov. Slednji dve sta tudi uporabljeni pri konstrukciji algoritma za napovedovanje, ki je predstavljen v nadaljevanju magistrskega dela.

### 3.4.1 Temeljna analiza

Za določanje »prave« cene delnice določene delniške družbe temeljna analiza uporablja pričakovane dobičke in dividende te družbe kakor tudi bodoče obrestne mere na denarnem trgu. V končni fazi ta analiza predstavlja poskus določanja sedanje diskontirane vrednosti vseh donosov, ki jih bo lastnik delnice prejel. Če ta diskontirana vrednost presega ceno delnice na trgu, bi na osnovi takega rezultata temeljne analize lahko sklepali, da je tako delnico priporočljivo kupiti. (Bodie, 1999, str. 348). Osnovni principi temeljne analize zahtevajo preučitev preteklih dobičkov ter bilanc podjetja kakor tudi sposobnost managementa. Poleg tega je navadno

potrebno preučiti tudi panogo, v katero analizirano podjetje sodi. Cilj take analize je pridobitev vpogleda v bodoče poslovanje podjetja in doseganje bodočih dobičkov, po možnosti prej in bolje kot ostali udeleženci na trgu.

EMH predpostavlja neuspešnost take analize s ciljem doseganja dobička, kajti če se analitik zanaša na javno dostopne informacije o poslovanju podjetij, bo težko bolj natančen od drugih udeležencev na trgu, ki imajo ravno tako dostop do teh javnih informacij.

### 3.4.2 Tehnična analiza

Tehnična analiza v svoji osnovi predstavlja iskanje ponavljajočih se vzorcev v cenovnih gibanjih delnic, na podlagi katerih analitiki poskušajo napovedati njihova gibanja v prihodnosti. Pri tem se za proučevanje preteklih cen vrednostnih papirjev (npr. delnic) največkrat uporabljajo grafikoni kot osnovno orodje. Čeprav »tehnični analitiki« priznavajo vrednost informacij, ki zadeva bodoče poslovanje podjetja, ki so temelj obravnave v temeljni analizi delnic, verjamejo, da uporaba tovrstnih informacij ni nujna za uspešno trgovanje z delnicami (Achelis, 2003). Ne glede na to, kateri temeljni razlog stoji za spremembo cene same delnice na trgu, lahko namreč v primeru njenega dovolj počasnega odziva analitik identificira trend gibanja in to znanje uporabi v svojo korist v času, ko na trgu traja prilagajanja cene delnice njeni pravi vrednosti. Ključni element uspeha tehnične analize je torej počasen odziv cene delnic na njene »temeljne« dejavnike ponudbe in povpraševanja. Ta predpostavka je sicer v nasprotju s hipotezo učinkovitih trgov, kar pa številnim analitikom po svetu ne preprečuje uporabe tovrstnih metod pri analizi vrednostnih papirjev.

Koncept tehnične analize izhaja iz teorije Charlesa Dowa, razvite v začetku 20. stoletja, resnično pa je tehnična analiza zaživela šele z razvojem in splošno dostopnostjo osebnih računalnikov, s katerimi je mogoče izvesti večje število zahtevnih izračunov v kratkem času.

Glede na to, da s tehnično analizo poskušamo iskati ponavljajočih se vzorcev v cenovnih gibanjih delnic, so za to vsekakor najuporabnejše orodje grafi, ki ponujajo možnosti za vizualno razpoznavanje značilnih vzorcev, trendi, drseče sredine ter indikatorji, ki bodo predstavljeni v nadaljevanju. Ne glede na izbiro orodja pa je treba vedeti, da je tako prikaz grafa kot tudi interpretacija le-tega odvisna od samega analitika, kar vnaša precejšnjo mero subjektivnosti v analizo in s tem pomeni veliko odvisnost uspeha tehnične analize od posameznega analitika.

#### *3.4.2.1 Grafi in vzorci*

Tehnični analitiki si pri vizualnem iskanju raznih vzorcev v cenovnih gibanjih delnic pomagajo z različnimi vrstami grafikonov, med katerimi se najpogosteje uporabljajo linijski in palični grafikoni. Slednji poleg zaključnih tečajev vsebujejo tudi podatke o najnižjih in najvišjih cenah delnic, doseženih tekom dneva in s tem omogočajo tudi kratkoročnejšo analizo, saj so začetni in zaključni tečaji delnic pomembni pri določanju optimizma in pesimizma pri trgovanju posameznega dne, kar je uporabno za določanje gibanja tečajev za nekaj dni vnaprej.

Linijski grafikoni, kjer gre za preprosto povezavo med točkami cen skozi čas, so primernejši za dolgoročne analize. Z njihovo pomočjo je možno razpoznavati vzorce v gibanjih cen kot so linije odpora in podpore (angl. resistance and support), »glava in ramena«, dvojni vrhovi in dvojna dna (Edwards, 1998, str. 69).

Eden izmed najpomembnejših vzorcev, ki ga je koristno identificirati v grafični analizi, je vsekakor trend gibanja tečajev. Trend načeloma pomeni konsistentno spreminjanje tečajev, torej naraščanje ali padanje kljub kratkoročnim nasprotnim gibanjem. Z drugimi besedami, če je v zaporedju valov vsako naslednje dno višje kot prejšnje, lahko rečemo, da gre za rast tečajev (bikovski trend) in nasprotno, kadar je vsak naslednji vrh v valovitem gibanju tečajev nižji od prejšnjega, lahko to označimo za padanje tečajev (medvedji trend). Glede na ročnost je mogoče, v skladu s teorijo Charlesa Dowa, ločiti primarne, sekundarne in terciarne. Prvi lahko trajajo od nekaj mesecev do nekaj let, drugi od enega do treh mesecev in zadnji so terciarni ali dnevni trendi (Bodie, 1997, str. 345). Prepoznavanje dolgoročnejših trendnih gibanj tečajev delnic, predvsem tistih naraščajočih, omogoča investitorjem sodelovanje v teh gibanjih in doseči zaslužke ob relativno nizkem tveganju.

Poleg ugotavljanja trendnih gibanj je pri analizi gibanja tečajev delnic potrebno iz podatkov izločiti nepomembne dogodke in se osredotočiti na pravo vrednost tečajev v nekem obdobju. Eno najstarejših orodij za ta namen predstavlja izračunavanje drsečih sredin. Drseča sredina predstavlja povprečno vrednost tečajev v nekem določenem časovnem obdobju. V najenostavnejši obliki jo lahko ponazorimo s sledečo enačbo:

$$MA = \frac{1}{N}\sum_{i=1}^{N} P_c(i)\,, \qquad (14)$$

kjer je $N$ dolžina časovnega okna, $P_c$ pa zadnji tečaj. $MA$ se izračuna tako, da $N$ zaporednih vzorcev seštejemo in vsoto delimo s številom vzorcev (Edwards, 1998, str. 337). S pomikanjem časovnega okna po časovni premici se izračuna povprečje za vsako časovno periodo. Take povprečne cene delnic v nekem obdobju lahko

jemljemo kot indikator »prave cene« delnice. Če je trenutna cena delnice nad povprečno potem lahko v prihodnosti pričakujemo njen padec. Vrednosti drseče sredine za določeno obdobje lahko jemljemo tudi kot dolgoročni trend. V primeru padajoče smeri takega trenda ter trenutne vrednosti delnice pod vrednostjo drseče sredine, lahko jemljemo porast cene delnice nad vrednostjo takega trenda (preboj) kot znak za obrat dosedanjega padajočega trenda delnice.

#### *3.4.2.2 Indikatorji*

Tehnični indikatorji ponazarjajo gibanje določenega parametra določene delnice skozi čas. Kot parameter se največkrat spremlja cena, lahko tudi obseg trgovanja ali pa kakšen drug indikator. Več kot sto tehničnih indikatorjev za predvidevanje cen delnic je bilo razvitih in se še vedno razvijajo na osnovi teh konceptov.

V literaturi se indikatorji po tipu največkrat delijo v dve skupini. V prvo spadajo indikatorji trenda, kjer indikatorji sledijo trendu. Najbolj znani med njimi so že omenjene drseče sredine in indikator MACD (angl. Moving Average Convergence-Divergence). Slednji je sestavljen iz večih drsečih sredin, in sicer se uporablja krivulja razlike med dvema drsečima sredinama s časovnim oknom različne dolžine, (linija MACD) največkrat so to 26- in 12-dnevne sredine. Tako kreirana krivulja se spremlja z drsečo sredino z oknom še manjše dolžine, največkrat 9 dni (signalna linija). Pregled presečišč omogoča signale za nakup oziroma prodajo delnice. Kadar linija MACD pade pod signalno, je to signal za prodajo in nasprotno (Achelis, 2003).

Drugo večja skupina indikatorjev sestavljajo t. i. oscilatorji ali vodilni (angl. leading) indikatorji. Ti indikatorji za razliko od indikatorjev trenda poskušajo predvideti, kaj se bo s tečaji zgodilo v bližnji prihodnosti. Na podlagi trenutne precenjenosti ali podcenjenosti tečajev poskušajo predvideti, ali se bo gibanje tečajev obrnilo navzgor ali navzdol. Predvsem dajejo informacijo o hitrosti spreminjanja cen in s tem prikazujejo, ali obstoječi trend pridobiva ali izgublja zagon. Obrat tečajev se tako poskuša predvideti, z določeno verjetnostjo seveda, še preden do njega dejansko pride. Popularnejši indikatorji v tej skupini so (Achelis, 2003):

- RSI (angl. relative strength index) je relacija med povprečnimi pozitivnimi in negativnimi spremembami cen v časovnem intervalu določene dolžine. Pogosto se uporablja interval za 14 dni nazaj.

- Drseče povprečje (angl. moving average) je dvig cene nad drsečim povprečjem cen in se razume kot signal za nakup in padec pod njim, kot signal za prodajo.

- MACD – uporabljeni sta dve drseči povprečji z različnimi časovnimi intervali. Signali za nakup in prodajo se ustvarijo na podlagi križanj iz izbranih cenovnih signalov.

### 3.4.3 Analiza časovnih vrst

Tradicionalne tehnike analize časovnih vrst datirajo iz konca dvajsetih let prejšnjega stoletja, ko je Yule (1927) izumil avtoregresivno metodo (AR) za predvidevanje sončevih peg. Splošni AR model izraža prihodnje vrednosti časovne vrste $y(t)$ kot linearno kombinacijo prejšnjih vrednosti $y(t-m)$, plus naključno komponento $e(t)$, ki predstavlja t. i. šum (angl. noise):

$$y(t) = \sum_{m=1}^{d} a_m y(t-m) + e(t) \qquad (16)$$

Drugi običajno sprejet model za analizo časovnih vrst je model drsečih sredin (angl. moving average, MA):

$$y(t) = \sum_{n=1}^{M} b_n e(t-n)\,. \qquad (17)$$

Zgornja enačba opisuje situacijo, kjer je časovna vrsta $y$ kontrolirana z drugo zunanjo časovno vrsto $e$ v linearni zvezi. S kombinacijo prejšnjih enačb dobimo avtoregresivni model drsečih povprečij (angl. autoregressive moving average model, ARMA), ki dominira na področju analize časovnih vrst več kot 50 let (Hellstrom, 1998):

$$y(t) = \sum_{m=1}^{d} a_m y(t-m) + \sum_{n=0}^{M} b_n e(t-n).$$

### 3.4.3 Nevronske mreže

Nevronske mreže se pogosto uporabljajo kot oblika nelinearne regresije, kjer imamo nalogo izbrati povezano aproksimacijo vrednosti med večdimenzionalnimi točkami. Pri učenju nevronskih mrež je potrebno vhodne in izhodne podatki mreži predstaviti hkrati. V primeru predvidevanja časovnih vrst je situacija malo drugačna, saj vsaj na začetku ni na voljo nobenih vzorcev z nekimi lastnostmi. Namesto tega morajo biti prihodnje vrednosti časovne vrste modelirane kot funkcija prejšnjih vrednosti. Ta začasna časovna dimenzija je lahko vključena v sistem nevronske mreže na več načinov (Hellstrom, 1998):

- Zakasnjene vrednosti $y(t)$, kot npr. $y(t), y(t-1), y(t-2),...$, se lahko uporabijo za konstrukcijo prihodnjega vektorja, ki služi kot vhod v nevronsko mrežo. Na ta način lahko gledamo kot na nelinearno posplošitev avtoregresijske metode. Izhod iz nevronske mreže lahko opišemo kot :

$$O(t) = g[y(t-1), y(t-2),..., y(t-d)],$$

kjer je $g$ nelinearna funkcija, ustvarjena v učnem procesu nevronske mreže. Kot poseben primer $g(t) = \sum_{m=1}^{d} amy(t-m) + e(t)$ lahko spoznamo kot model AR. Smiselnost te nelinearne posplošitve temelji na dveh ugotovitvah: Za nevronske mreže kot tudi za večino deterministčnih sistemov, lahko trdimo, da pri delovanju sistema skozi čas obstaja preslikava tipa 1:1 med preteklimi vrednostmi $y(t-1), y(t-2),..., y(t-d)$ časovne vrste ter stanjem sistema v času t. Na podlagi tega lahko sklepamo, da obstaja funkcija $g$, tako da velja $y(t) = g[y(t-1), y(t-2),..., y(t-d)]$. Dokazano je tudi, da je nevronska mreža z najmanj eno vmesno plastjo nevronov sposobna aproksimirati katero koli nelinearno zvezno funkcijo.

- Povratne nevronske mreže imajo vrsto arhitekture, kjer izhod vmesnih nevronov predstavlja povratno informacijo vhodnim nevronom. Na ta način bo mreža ohranila spomin prejšnjih vrednosti vhodnih podatkov. Posamezna vrednost $y(t-1)$ je uporabljena kot vhod in posamezna vrednost $O(t)$ je ustvarjena kot izhod iz nevronske mreže. Začasne odvisnosti so modelirane s pomočjo uteži v povratni zanki nevronske mreže.

Poleg tega da se nevronske mreže uporabljajo kot nelinearno obliko regresije, se nevronske mreže pogosto uporabljajo za reševanje klasifikacijskih problemov, kjer so večdimenzionalni vhodni vektorji mapirani večdimenzionalnim izhodnim vektorjem (pri tem pa število dimenzij na izhodu in vhodu ni nujno enako). Na izhodu je natanko en element enak 1, ostali pa 0. Pozicija elementa, ki je 1, predstavlja razred, povezan s posledičnim vhodnim vektorjem. Na ta način so lahko primeri vhodnih vektorjev in posledičnih izhodnih vektorjev uporabljeni v nevronski mreži, ki določi razred vhodnemu vektorju (Hellstrom, 1998). Največkrat se implementira kot večslojni perceptron s $p$-vhodi in $m$-izhodi. Izhodni sloj pogosto uporablja sigmoidno funkcijo kot aktivacijsko funkcijo, ki ustvari m-dimenzionalni izhodni vektor z vrednostmi v zaprtem intervalu [0,1]. Dokazano je, da so izhodne vrednosti iz uspešno naučene nevronske mreže asimptotični približek razrednih vrednosti, ki jih lahko zasede izhodni vektor. To pomeni, da je optimalna klasifikacija vhodnih vektorjev pozicija v

izhodnem vektorju, ki ima največjo vrednost. Zato se običajno zamaknejo ciljne izhodne vrednosti za majhno vrednost $E$ in element 1 iz $1-E$.

### 3.4.4 Metode najbližjih sosedov

Metoda $k$-najbližjih sosedov je splošna tehnika za klasificiranje, ki od vseh še najmanj temelji na kakršnih koli predpostavkah glede funkcije, ki jo želimo modelirati. Če želimo klasificirati določeno točko $p$, preprosto najdemo množico $k$ najbližjih točk iz celotne učne množice primerov (Mitchell, 1997, str. 48). V primeru časovnih vrst so vhodne točke $p_t$ tipično sestavljene z izbiro zaporednih vrednosti iz časovne vrste.

V splošnem primeru so lahko vhodne točke $p$ kateri koli tip vektorja, ki naj bi imel zmožnosti predvidevanja. V primeru predvidevanja delnic je klasifikacija $C_k(t)$ za točke tipično predznak $k$–dnevnih donosov, izračunanih za $k$-dni vnapre (Hellstrom, 1998)j:

$$C_k(t)\begin{cases} +1 : \check{c}e\ y(t+k) > y(t) \\ -1 : \check{c}e\ y(t+k) < y(t) \\ 0 : \check{c}e\ y(t+k) = y(t) \end{cases}.$$

Bližina se največkrat izračuna kot evklidska distanca v vhodnem prostoru. Aritmetična sredina ali mediana klasifikacija $k$-najbližjih točk se potem vzame kot približek klasifikacije za točko $p$ (Witten, Eibe, 2000, str. 44). Na ta način lahko takoj naredimo klasifikacijo ob dani množici učnih primerov. Čeprav je ta metoda računsko zelo zahtevna v aplikativni fazi, je zelo privlačna v začetni analizi podatkov, kjer so vprašanja glede predvidljivosti ter izbiri vhodnih spremenljivk pomembna zadeva. Lahko bi celo rekli, da neuspeh v primeru uporabe metode $k$-najbližjih sosedov na določenem problemu pogojuje kot neuspeh tudi uporabo katere koli druge induktivne metode. Osnovna predpostavka te metode je, da so bližnji vhodni podatki oziroma točke sorazmerne bližnjim izhodnim podatkom. Težko bi lahko našli primer, da ob zadovoljivi količini podatkov neke zvezne funkcije ne bi mogli klasificirati novih primerov. V takem neuspešnem primeru lahko zaključimo, da funkcijske odvisnosti med izbranimi vhodnimi in izhodnimi podatki ne moremo izraziti, ne da bi dodatno omejili funkcijsko odvisnost. Vendar pa so lahko druge metode, ki uporabljajo "močnejše" modele, uspešnejše kot metoda k-najbližjih sosedov, ki vnaprej ne predvideva nobenih odvisnosti v podatkih. Treba je tudi upoštevati, da je metoda k-najbližjih sosedov v normalni implementaciji globalna metoda. V iskanju najbližjih sosednjih točk algoritem preišče bodisi celotno učno množico bodisi vse prejšnje točke v učni množici. Ta druga metoda je uporabna, ker s tem preprečujemo pogled

v prihodnje vrednosti, kadar s tem poskušamo predvidevati obnašanje določene časovne vrste. V vsakem primeru so sosedne točke izbrane iz časovnega obdobja, ki je lahko zelo dolgo, kar v primeru nestacionarnih funkcij pomeni, da izberemo slabe sosede. Načini uteževanja ali pa uporaba časovnega okna lahko bistveno izboljša rezultate predvidevanja.

# 4 OBLIKOVANJE ALGORITMA ZA NAPOVEDOVANJE CEN DELNIC

Strojno učenje v današnji dobi »zmogljivih« računalnikov ponuja možnost za nove pristope k iskanju možnosti za doseganje dobička na borzi vrednostnih papirjev. Seveda računalniške metode zahtevajo algoritem v obliki računalniškega programa. Algoritem kot niz navodil predstavlja postopek za reševanje določene naloge. Razvoj ustreznega algoritma seveda zahteva svoj proces. Ta proces razvoja nekega algoritma lahko razdelimo na več faz, in sicer (Damij, 1994):

- definicija problema,
- groba zamisel algoritma,
- izdelava algoritma.

V tem magistrskem delu je poudarek predvsem na prvih dveh fazah, ki sta zelo pomembni za razumevanje problema napovedovanja delnic. Tretja faza, faza izdelave algoritma, je nadomeščena z opisom že izdelanega programa.

## 4.1 Definicija problema

**Definicija problema** je v bistvu najtežja faza v procesu izdelave nekega algoritma. Cilj te faze je natančno določiti, kaj je osnovni namen algoritma, kakšne podatke uporablja in kakšne rezultate naj kreira (Damij, 1994).

Osnovni namen algoritma za napovedovanje delnic je uspešno predvidevanje bodočega dogajanja na borzi. Zato je potrebno preučiti sam koncept in metode napovedovanja, ki jih lahko uporabljamo pri izgradnji nekega algoritma. Finančni programi, s katerimi si vlagatelji pomagajo pri investicijah, največkrat uporabljajo borzne podatke, torej podatke, ki so relevantni za tehnično analizo. Ti podatki, ki se nanašajo na trgovanje z vrednostnimi papirji, kot tudi ostali finančni podatki glede inflacije, rasti bruto domačega proizvoda ipd., v svoji izvirni obliki dostikrat niso neposredno uporabni za obdelavo s pomočjo metod strojnega učenja. Zato jih je smiselno prej ustrezno transformirati. Uspešnost napovedovanja cen delnic je tisti rezultat, ki ga od našega algoritma pričakujemo. Kriteriji za ocenjevanje uspešnosti bodo prav tako predstavljeni v tem poglavju.

### 4.1.1 Podatki

Algoritem za predvidevanje uporablja množico znanih entitet podatkov, kot sta tečaj delnice ali obseg prometa z namenom ustvariti predvidene vrednosti istovrstnih lahko pa tudi drugačnih entitet podatkov. V primeru predvidevanja delnic lahko entitete podatkov, ki jih je smiselno uporabiti, razdelimo v dve kategoriji: čisti tehnični podatki in temeljni podatki. Tehnični podatki se uporabljajo pri tehnični analizi delnic, katerih prognoze tečajev temeljijo izključno na preteklih podatkih. Temeljna analiza pa vključuje tudi podatke, povezane s podjetji, katerih delnice analiziramo, njihovo panogo ter stanje na trgu. Poleg teh dveh kategorij podatkov lahko s pomočjo transformacij ustvarimo različne izvedene vrste podatkov, ki so lahko bolj primerne za uporabo v analizi in predvidevanju delniških tečajev.

#### 4.1.1.1 Podatki tehnične analize

Podatki o dnevnem trgovanju s posamezno delnico se oblikujejo v večih časovnih vrstah (Achelis, 2003):

- $y_c$ ali $y$ : zaključni tečaj, po katerem je bil sklenjen zadnji posel,
- $y_h$ : maksimalni, tj. najvišji dnevni tečaj,
- $y_l$ : minimalni, tj. najnižji dnevni tečaj,
- $V$ : promet (angl. volume).

Tovrstni podatki se redkeje uporabljajo v podatkovnem modeliranju delniških tečajev, pač pa se pogosto uporabljajo kot odločitveni razlog za sprožitev nakupa oziroma prodaje. Kot najbolj očitna izbira entitete za predvidevanje se izkaže časovna vrsta $y_c$, ki pa ima nekaj pomanjkljivosti:

- tečaji $y_c$ ponavadi precej variirajo, kar otežuje izdelavo pravilnega modela za daljše časovno obdobje.
- $y_c$ za posamezne delnice se lahko razlikuje v veliki meri skozi več let in je zato neuporaben kot vhodni podatek pri modeliranju.

Zaradi navedenih pomanjkljivosti se pri podatkovnem modeliranju največkrat uporabljajo podatki o donosnosti posamezne delnice v določenem dnevu.

#### 4.1.1.2 Podatki temeljne analize

Poleg prej opisanih dnevno vzorčenih podatkov obstaja še veliko informacij v zvezi z aktivnostjo oziroma finančno situacijo posameznega podjetja. Večina podjetij, ki

kotirajo na borzi, je predmet analize poklicnih borznih analitikov in finančnih institucij. Analize so dostikrat prezentirane v številčni obliki, ki naj bi nakazale “pravo” t. i. notranjo vrednost delnice obravnavanega podjetja. Dostikrat so podana tudi priporočila o nakupu oziroma prodaji, bodisi intuitivno bodisi s pomočjo matematičnega tehtanja analitičnih rezultatov oziroma razpona “pravih” vrednosti. Temeljna analiza podjetja se tipično osredotoča na sledeče faktorje (Bodie, 1997):

1. Gospodarstvo:
- inflacija,
- obrestne mere,
- plačilna bilanca itd.
2. Stanje v panogi podjetja:
- indeksi (tehtane srednje vrednosti), kot so SBI 20, PUBIX itd.,
- cene povezanih dobrin, kot so nafta in naftni derivati, tuje valute,
- vrednosti delnic konkurenčnih podjetij.
3. Stanje podjetja, dobljeno iz objavljenih letnih finančnih izkazov. Iz teh lahko preračunamo različne spremenljivke:
- P/E količnik. Razmerje med ceno delnice in dobičkom na delnico v zadnjih 12 mesecih.
- knjižna vrednost delnice; kapital podjetja deljen s številom izdanih delnic podjetja.

Večina analiz poklicnih analitikov ni javno dostopnih, vendar pa lahko precej kvalitetnih podatkov najdemo v finančnih časopisih, v poslovnih izidih prek borze in v elektronskih bazah podatkov, kot je IBON. Vendar pa je dostopnost večjih količin podatkov, ki jih lahko uporabimo pri računalniški obdelavi, lažja za tehnične kot pa za temeljne vrste podatkov.

#### 4.1.1.3 Izvedeni podatki

Iz podatkov, ki se uporabljajo pri temeljni, predvsem pa tehnični analizi delnic lahko izvedemo drugačne vrste količin, ki so dostikrat uporabnejše za računalniško analizo in napovedovanje dogajanja na kapitalskem trgu. Dostikrat lahko zasledimo uporabo podatkov, kot so donosnost, točke preobratov ali pa trend. Tovrstne podatke lahko pridobimo na sledeč način:

- **Donosnost**

  Enostopenjsko donosnost $R(t)$ lahko definiramo kot relativno povečanje cene delnice glede na prejšnjo točko v časovni seriji (navadno prejšnji dan) (Hellstrom, 1998):

$$R(t) = 100 * \frac{y(t) - y(t-1)}{y(t-1)}. \quad (15)$$

Poznamo tudi pogosto uporabljeno varianto logaritemske donosnosti:

$$R(t) = \log \frac{v(t)}{y(t-1)}. \quad (16)$$

Logaritmirana donosnost se pogosteje uporablja v raziskovalne namene, medtem ko prejšnjo različico pogosteje srečamo v praktični uporabi. Če uporabimo naravni logaritem, se obe meri pri majhnih spremembah časovne vrste ne razlikujeta dosti. $R(t)$ lahko posplošimo tako, da pokriva več kot enodnevni interval. Donosnost $k$-te stopnje lahko tako definiramo kot:

$$R_k(t) = 100 * \frac{y(t) - y(t-k)}{y(t-k)}. \quad (17)$$

- **Trend**

Donos $R_k(t)$ lahko vzamemo kot $k$-dnevni cenovni trend za določeno delnico. Prav tako je primerno deliti $k$ s številom dni, če hočemo dobiti dnevno povišanje cene. Vrednosti trenda za različne vrednosti $k$ lahko analiziramo na enaki osnovi. Tako lahko oblikujemo naslednjo formulacijo za $k$-trend $T_k(t)$:

$$T_k(t) = \frac{100}{k} \times \frac{y(t) - y(t-k)}{y(t-k)}. \quad (18)$$

- **Količina prodaje**

Povečanje količine prodanih delnic na borzi je dostikrat interpretirano kot indikator, da so prišle na trg neke nove informacije (Achelis, 2003). Zato lahko tudi stopnjo spremembe v številu delnic, ki so določenega dne zamenjale lastnika pri trgovanju, koristno uporabimo kot dodatno vhodno spremenljivko.

- **Točke preobratov**

Na točke preobratov med naraščanjem in padanjem na grafu, ki prikazuje gibanje cene neke delnice, lahko gledamo kot na pozicije, kjer je prišlo do izenačitve med ponudbo in povpraševanjem po neki delnici (Achelis, 2003).

Zato lahko te točke jemljemo kot pomembnejše od tistih vmes med njimi. Tako bi lahko vzeli razdalje v času in ceni kot vhodne podatke za predvidevanje naslednjih preobratov.

- **Umetni podatki**

  Z generiranjem umetnih podatkov (z dodajanjem šuma k ceni $y$ ali pa z zamikom podatkov) dosežemo določeno stopnjo neodvisnosti od časa. Na ta način povemo algoritmu za modeliranje, naj se osredotoči na temeljne lastnosti podatkov, namesto da modelira tisto, kar je v bistvu šum v podatkih.

- **Relativna uspešnost delnic**

  Namesto predvidevanja bodočih cen ali pa donosnosti posamezne delnice se lahko osredotočimo na problem, kako primerjati rezultate, ki jih posamezne delnice dosegajo, med večimi delnicami hkrati. Ena izmed metod, ki pri analizi obravnava več delnic hkrati, je zasnovana na principu rangiranja delnic glede na njihovo donosnost. Pri tem pa algoritem za modeliranje kot vhodne podatke uporabi tako grajeno rangirno listo. Koncept rangiranja je podrobneje pojasnjen v poglavju 4.2.3.

### 4.1.2 Transformacija podatkov

V poglavju 4.1.1 opisane vhodne podatke je pogosto potrebno preoblikovati, preden začnemo z modeliranjem. Glede na »obliko« podatkov in s tem namen potrebne transformacije je največkrat potrebno zmanjšanje dimenzij podatkov, njihova normalizacija ali pa jih je potrebno linearizirati.

#### *4.1.2.1 Zmanjšanje dimenzij*

Kadar modeliramo podatke o ceni delnic, je zelo koristno uporabiti čim manjše število vhodnih spremenljivk, saj je izgradnja modela oziroma model tako uporabnejši. Da zmanjšamo število vhodnih spremenljivk, je tako koristno iz obstoječih spremenljivk izvesti agregatne spremenljivke, ki odražajo več točk v časovni vrsti z eno vrednostjo. Primeri takih agregatnih spremenljivk so drseča povprečja ali drseče sredine spremenljivk ( $y$ , $y_h$ , $y_l$ in $V$ ).

### *4.1.2.2 Normalizacija*

Namen normalizacije podatkov je zmanjšanje razlik v razponu vrednosti vhodnih spremenljivk. Razlikujemo dva tipa normalizacije (Hellstrom, 1998):

- Zmanjšanje velikostnih razlik med različnimi vhodnimi spremenljivkami, s čimer dosežemo podobno obnašanje pri algoritmu za modeliranje. Primer: pred vnosom spremenljivk v nevronsko mrežo se vse spremenljivke preračunajo v vrednosti v razponu od 0 do 1. To preračunavanje se običajno izvaja avtomatično v paketih z nevronskimi mrežami.

- Zmanjšanje časovnih razlik v časovni vrsti, kar zmanjša efekte nestacionarnosti. Na primer normalizacija vrednosti spremenljivk $y, y_h, y_l$: izmerjene vrednosti časovnih vrst $y, y_h, y_l$, pogosto variirajo za več sto odstotkov, če jih merimo v obdobju več let. Tako je pogosto potrebno normalizirati podatke, da bi zmanjšali učinke nestacionarnosti. Obstaja več metod, kot na primer:

    1. transformacija v relativne spremembe (donose),
    2. transformacija v logaritmirane donose,
    3. normalizacija z uporabo aritmetične sredine,
    4. normalizacija z uporabo aritmetične sredine in standardnega odklona.

Kot je pojasnjeno v poglavju 2.5.3., sta prvi dve meri precej podobni, ko gre za majhne spremembe v časovni vrsti. Razlog za uporabo logaritmiranih donosov je statistične narave. Varianca se stabilizira in reziduali (angl. outliers) so manj vplivni. Obe transformaciji se zelo pogosto uporabljata pri napovedovanju časovnih vrst finančnih podatkov. Normalizacija z uporabo aritmetične sredine je definirana kot:

$$y_n(t) = \frac{y(t)}{y_{sred}(t)}, \tag{19}$$

kjer se srednja vrednost izračunava z uporabo premičnega okna fiksne dolžine, npr. za 30 dni nazaj. Normalizacija z uporabo aritmetične sredine in standardnega odklona pa je definirana kot:

$$y_n(t) = \frac{y(t) - y_{sred}(t)}{\sigma_y(t)}, \tag{20}$$

kjer je ocenjena vrednost standardnega odklona $\sigma_y(t)$ kot tudi $y_{sred}(t)$ izračunana skozi premično okno dolžine $n$. Spremenljivka $y_n(t)$ izraža število standardnih

odklonov, za katere se lahko vrednosti spremenljivke *y* razlikujejo od njene tekoče mediane. Podobno lahko normalizirano količino (volume) $V_n(t)$ izračunamo kot:

$$V_n(t) = \frac{V(t) - V_{sred}(t)}{\sigma_V(t)}, \tag{20}$$

kjer sta aritmetična sredina $V_{med}(t)$ in standardni odklon $\sigma_V(t)$ obsega trgovanja izračunana skozi tekoče okno fiksne dolžine $n_V$. Običajno okno obsega 30 preteklih dni. $V_n(t)$ izraža število standardnih odklonov, za katero se vrednost količine (angl. volume) razlikuje od tekoče aritmetične sredine.

#### *4.1.2.3 Linearizacija*

Dostikrat je zaželeno izločiti očitne nelinearne faktorje, ki bi jih bilo potrebno drugače upoštevati pri modeliranju. Primer je časovna vrsta podatkov $V(t)$, promet z določeno delnico v določenem dnevu $t$, ki je pogosto popačena zaradi ekstremno visokih vrednosti kot posledica posameznih trgovanj s strani institucionalnih investitorjev, kot so vzajemni skladi. Linearizacijo lahko v tem primeru izvedemo s pomočjo sploščujoče funkcije (angl. squashing function), ki zmanjšuje vrednosti podatkov, ki presegajo določeno mejo. Na primer vhodne podatke $V(t)$ lahko preoblikujemo v $V'(t)$ skladno s funkcijo oblike (Morton, 2003):

$$V'(t) = \frac{1}{1 + e^{-V(t)}}. \tag{21}$$

Šele ko imamo podatke pripravljene v obliki, ki je uporabna za gradnjo verodostojnega modela za iskanje zakonitosti v teh podatkih, se lahko lotimo napovedovanja gibanj na kapitalskem trgu. V naslednjem poglavju bodo opisani koncepti napovedovanja, ki so primerni za kapitalski trg, kaj je sploh možno napovedovati in v kakšni obliki.

## 4.2 Zamisel algoritma

Zamisel algoritma pomeni razgradnjo obravnavanega problema na niz manjših podproblemov ter izbiro matematične metode, ki najbolje ustreza reševanju tega problema (Damij, 1994).

Pri definiciji osnovnega namena našega algoritma smo nalogo algoritma opredelili, kot predvidevanje bodočega dogajanja na borzi. To pomeni da bo algoritem moral na osnovi preteklih podatkov zgraditi model, ki bo poskušal predvideti vrednosti preučevanih količin v nekem končnem številu dni vnaprej. Kot proučevane količine se v praksi, kot tudi v akademskih raziskavah, največkrat izbere cene ali donosnosti posamezne delnice. Kadar se lotimo neposredno napovedovanja cen delnic lahko uporabimo različne pristope. Pri napovedovanju se lahko zgledujemo po analizi časovnih vrst, ali pa problem rešujemo skozi tehnično analizo delnic, kjer je poudarek na oblikovanju določenih trgovalnih pravil (Hellstrom, 1998).

V tem magistrskem delu pa je poudarek na malo drugačnem načinu predvidevanja delnic in sicer predvidevanju ranga delnice v skupini izbranih delnic. Ta način analize lahko zasledimo pri raziskavi predvidevanja delnic na švedskem kapitalskem trgu, kjer je dal zelo dobre rezultate (Hellstrom, 2000). Prednosti tega analitičnega pristopa lahko vidimo ob primerjavi z do sedaj bolj uveljavljenimi analizami, kot sta analiza časovnih vrst in tehnična analiza. Zato bodo v nadaljevanju predstavljene vse tri možne zamisli.

Napovedovanje vrednosti v časovni vrsti in oblikovanje trgovalnih pravil lahko jemljemo kot vrsto induktivnega učenja. Skladno s principom induktivnega učenja, kot je to opisano v drugem poglavju tega dela, se izgradnja modela (koncepta) oblikuje iz dane množice primerov. Tukaj lahko kot primere uporabimo zgodovinske podatke o donosnosti v obliki časovne serije $X(t)$ torej donosnost $X$ v posameznem dnevu $t$. Tako kot se v drugem poglavju izoblikuje koncept »ptica«, je naša naloga izoblikovati model gibanja donosnosti delnic na podlagi preteklih donosnosti po posameznih dnevih. Koncept bi najprimerneje koristil v obliki funkcije $f(X(t))$. Taka funkcija bi nam zagotavljala vrednosti o donosnosti v bodočem času, premaknjenim za $h$ dni, torej v času $t+h$.

K gradnji funkcije najlažje pristopimo tako, da v definiranem časovnem obdobju izberemo posamezne primere iz časovne serije tako, da izberemo dvojice sedanjih in bodočih donosnosti:

$$\left\{\left(X(t), z(t+h)\right), t=1,\ldots N\right\},$$

Pri tem predvidevamo, da velja funkcijska odvisnost:

$$z(t+h) = f(X(t)), \forall t, \qquad (22)$$

Tako kot pri oblikovanju koncepta ptica poskušamo za vsak $t$ ustvariti funkcijo $g(X(t))$, ki je seveda samo približek funkcije $f(X(t))$ v tem smislu, da je izpolnjen kriterij najmanjšega odstopanja od izvirne funkcije, ki se ji poskušamo približati, torej

najmanjše napake. Napako med izvirno funkcijo in funkcijo, ki jo aproksimiramo, lahko zapišemo kot:

$$e_t = e(g(t), z(t+h)). \tag{23}$$

Pri tem je $e$ neka funkcija napake, ki jo poskušamo minimizirati. Zaradi poenostavitve lahko $g(X(t))$ označimo kot $g(t)$.

Definirati moramo torej sledeče entitete (Hellstrom, 1998):

- vhod: $X(t)$ vektor,
- izhod $z(t+h)$ vektor,
- funkcija napake , $e(g,z)$,
- kriterij napake, to je največkrat minimum funkcije $e$,
- predhodno znanje o aproksimacijski funkciji $g$, če ga imamo na voljo.

Zanimivo je, da je funkcija $g$, ki se jo na koncu naučimo, odvisna od vseh teh entitet.

### 4.2.1 Časovne vrste

Vhodna komponenta $X$ in izhodna komponenta iz vsakega podatkovnega primera $((X(t), z(t+h))$ ima v pristopu časovnih vrst naslednjo obliko (Hellstrom, 1998):

$$X(t) = (y(t), y(t-1),..., y(t-k+1)) \tag{24}$$

in

$$z(t+h) = y(t+h). \tag{25}$$

Vhodne vrednosti so na ta način oblikovane kot zamaknjene vrednosti vhodne časovne vrste $y(t)$, za katero želimo predvideti vrednosti $h$ korakov vnaprej. V primeru delniških časovnih vrst so njene vrednosti največkrat $h$-dnevni donosi ali pa cene ob zaključku trgovanja na borzi. Napaka pri predvidevanju e pri vsakem koraku predstavlja razliko med tistim, kar nam da funkcija $g$, in tistim, kar nam podatkovni primer predlaga. To je:

$$e_t = g(t) - (t+h) \tag{26}$$

Kot kriterij napake uporabimo vsoto kvadratov odklonov:

$$\|E\| = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e_t{}^2} \tag{27}$$

Učna naloga v tem primeru je najti tako funkcijo $g$, pri kateri bo $\|E\|$ najmanjši. Običajna izbira je lahko:

- Linearni avtoregresijski (AR) model:

$$g(t) = \sum_{m=1}^{k} a_m y(t-m-1) + a_0 . \qquad (28)$$

- Nelinearni model, implementiran s pomočjo večnivojske nevronske mreže. Z enim skritim nivojem in linearnim izhodom definira funkcijo $g$ kot:

$$g(t) = \sum_{j} \left( w_j h\left( \sum_{m=1}^{k} w_{m,j} y(t-m-1) + w_{0,j} \right) + w_0 \right), \qquad (29)$$

kjer je $h$ nelinearna funkcija, kot na primer sigmoidna funkcija $h(x) = \frac{1}{1+e^{-x}}$.

Učenje predstavlja postopek iskanja parametrov, pri katerih je kriterij napake najmanjši. Običajno je minimizirana vsota kvadratov odklonov končna ocena uspešnosti pri predvidevanju. Če hočemo uporabiti tako dobljeno funkcijo, je potrebno oblikovati trgovalna pravila, kdaj kupiti in kdaj prodati na osnovi predvidevanj. Preprosto trgovalno pravilo je moč oblikovati na sledeč način:

$$D(t) = \begin{cases} kupi & : \check{c}e\ g(t) > \alpha \\ prodaj & : \check{c}e\ g(t) < \alpha \end{cases}.$$

Pri tem imamo določen prag vrednosti $\alpha$, ki predstavlja razmejitveno vrednost modela podatkov $g(t)$, pri katerem se odločamo, ali bomo delnico prodali ali kupili. V praksi si lahko postavimo določeno rezervo, saj je za inicializacijo nakupa ali prodaje potrebna malo večja sprememba v ceni, zato lahko z uvedbo spodnjega in zgornjega praga prestavimo mejo, pri kateri se sproži trgovalna akcija, vmes pa počakamo. Tako lahko prejšnje pravilo preoblikujemo v sledečo obliko:

$$D(t) = \begin{cases} kupi & : \check{c}e\ g(t) > \alpha \\ prodaj & : \check{c}e\ g(t) < -\beta \\ nespremenjeno & : drugace \end{cases}.$$

Pri tem sta $\alpha$ in $\beta$ pragovni vrednosti za izvedbo prodaje ali nakupa, ko je vrednost našega modela $g(t)$ oziroma približka originalne funkcije $X(t)$ presegla ali pa padla pod določen prag.

Koren srednje vsote kvadratov odklonov kot mera, ki kaže, kako dobra aproksimacija originalne funkcije je naša funkcija $g(t)$, ni vedno idealna za predvidevanja časovnih vrst, zato ker:

- Koncept korena srednje vsote kvadratov odklonov od prvotne funkcije obravnava vse odklone enako, tako majhne kot tiste velike. To ni vedno najbolj primerno, ko gre za gibanja delniških tečajev. Točke pri predvidevanju, ki za trgovanje ne bi bile nikoli uporabljene (zaradi premajhnih sprememb cene bi bile nezanimive), lahko povzročijo večje reziduale na drugih točkah, ki bi bile bolj zanimive zaradi zmanjševanja celotne vsote kvadratov odklonov.
- Majhna predvidena sprememba v ceni delnice, ki pa je v resnici rezultirala v veliko večji vrednosti, čeprav v isti smeri, bo z enakim obravnavanjem odklonov precej spregledana v primeru mere korena srednje vsote kvadratov odklonov. Kljub temu bo vlagatelj v tem primeru verjetno zadovoljen, če bo majhna predvidena sprememba cene dovolj velika, da sproži nakupni signal.

### 4.2.2 Oblikovanje trgovalnih pravil

Namesto da poskušamo predvideti cene delnic za vse točke od 1 do $N$ in se na podlagi konkretnih predvidevanj cene odločamo za nakup in prodajo, lahko skonstruiramo algoritem, katerega zahteva je, da spozna situacije, na podlagi katerih je potrebno kupiti ali prodati vrednostni papir. Pri tem ni potrebno, da funkcija $g(t)$ odraža časovno vrsto, torej cene same, pač pa je to lahko model v drugačni obliki. Nalogo predvidevanja lahko definiramo kot klasifikacijski problem s tremi razredi: »Kupi«, »Prodaj«, »Nespremenjeno«. Ker pa v bistvu potrebujemo klasifikacijo nekako v celotnem časovnem obdobju preučevanja, je najbolje klasificirati vse točke od 1 do $N$. Tako še vedno potrebujemo ogrodje časovnih vrst. Trgovalno pravilo lahko opišemo kot časovno vrsto $T(t)$, ki jo definiramo na sledeč način (Hellstrom 1998):

$$T(t) = \begin{cases} kupi & : \check{c}e\ g(t) = 1 \\ prodaj & : \check{c}e\ g(t) = -1 \\ nespremenjeno & : \check{c}e\ g(t) = 0 \end{cases}.$$

Vhodni del vsakega podatkovnega primera $((X(t), z(t+h))$ ima isto obliko kot pri načinu napovedovanja vrednosti iz časovne vrste:

$$X(t) = (y(t), y(t-1), ..., y(t-k+1)), \qquad (30)$$

s tem, da $y(t)$ ne predstavlja ceno delnice v določenem časovnem trenutku t, pač pa drugo posredno vrednost.

Tipično se uporablja $h$-dnevni donos. Izhodni del z vsakega podatkovnega primerka ima tako sledečo obliko:

$$z(t+h) = R_h(t+h). \tag{31}$$

Tukaj za razliko od zamaknjene časovne vrste cen delnic formuliramo zamaknjeno časovno vrsto donosov s $h$-dnevnim zamikom.

Napako v predvidevanju $e_t$ pri vsakem koraku $t$ je tako potrebno definirati tudi nekoliko drugače kot pri navadni časovni vrsti, in sicer je smotrno namesto vrednosti odklonov v merah vrednosti izvirne časovne vrste uporabiti drugo formulacijo, kot je npr. stopnja pravilnih zadetkov (angl. hit rate), ki naj bo seveda čim višja in jo zato poskušamo maksimizirati:

$$e_t = \begin{cases} 1: \ g(t) = 1 \quad IN \quad z(t+h) < 0 \\ 1: \ g(t) = -1 \quad IN \quad z(t+h) > 0 \\ 0: \ drugace \end{cases}.$$

Na ta način se z napako kaznuje predviden donos vsakič, ko se razlikuje od dejanskega donosa $z(t+h)$. Druga možnost, katere cilj je maksimiziranje teoretičnega profita za trgovalno pravilo, je sledeča oblika napake:

$$e_t = \begin{cases} -z(t+h): \ g(t) = 1 \quad IN \quad z(t+h) < 0 \\ \ \ z(t+h): \ g(t) = -1 \quad IN \quad z(t+h) > 0 \\ -z(t+h): g(t) = 1 \quad IN \quad z(t+h) > 0 \\ \ \ z(t+h): g(t) = -1 \quad IN \quad z(t+h) < 0 \\ \qquad 0: \ drugače \end{cases}.$$

Na tak način napaka odraža višino izgube pri trgovanju na način konkretnega trgovalnega pravila. Srednja vrednost napake za primere, ko pri trgovanju dosežemo rezultat različen od nič:

$$\|E\| = \frac{\sum et}{\sum at}, \tag{32}$$

kjer $a_t$ predstavlja:

$$a_t = \begin{cases} 1: če \quad et \neq 0 \\ 0: drugace \end{cases}.$$

Podobno kot pri neposrednem predvidevanju cene delnice na osnovi časovne vrste cen delnice, je tudi pri tem načinu naloga predvidevanja najti tako funkcijo, pri kateri je napaka $\|E\|$ minimalna.

**Primer:**
*Vhod:* vrednosti cen delnic z zamikom :

$$X(t) = (y(t), y(t-1),..., y(t-k+1)).$$

*Izhod:* 5-dnevni donos, izračunan pri $t+5$:

$$z(t+5) = R(t+5).$$

Funkcijo napake $e_t$ lahko izberemo po vzoru ene izmed zgornjih enačb. Kriterij za napako pa po vzoru enačbe za računanje $\|E\|$. Funkcija $g$, ki poskuša povzeti gibanje donosov, je oblikovana s pomočjo drsečih povprečij na sledeč način:

$$g(t) = \begin{cases} 1: & če\ povp_K(t) > povp_D(t)\ IN\ povp_K(t-1) \le povp_D(t-1) \\ -1: & če\ povp_K(t) < povp_D(t)\ IN\ povp_K(t-1) \ge povp_D(t-1) \\ 0: & drugace \end{cases},$$

kjer je drseče povprečje $povp(t)$ za določeno število časovnih enot $k$ izračunano po formuli:

$$povp_K(t) = \frac{1}{k}\sum_{m=0}^{k-1} y(t-m).$$

Učenje pri tem primeru je sestavljeno iz iskanja optimalnih vrednosti, s katerimi bo opredeljena funkcija $g$, torej število dni, za katere računamo krajša drseča povprečja $S$ (angl. short) in daljša $L$ (long). Recimo, da so tipične vrednosti za krajše drseče povprečje 2 dni ($S$=2) in za daljše povprečje 10 dni (L=10). Trgovalno pravilo signalizira nakup, če krajše drseče povprečje $mav$ prečka daljše drseče povprečje od spodaj navzgor. Signal za prodajo pa je sprožen v obratnem primeru, ko krajše drseče povprečje prečka daljše povprečje $mav_l$ od zgoraj navzdol (slika 15). Optimalne nastavitve za $S$ in $L$ so torej določene v učnem procesu.

Taka formulacija pokriva tako imenovane tehnične indikatorje, ki so pogosti pri analizi delnic in trgovanju v praksi. Čeprav redko srečamo tak formalen opis, se jih da pogosto formulirati v obliki trgovalnih pravil, kot je to opisano zgoraj. Kreiranje trgovalnih pravil je na splošno lahko zelo dober pristop k razvoju algoritmov za predvidevanje obnašanja vrednostnih papirjev na trgu.

**Slika 15: Trgovalno pravilo, oblikovano na podlagi dveh drsečih povprečij**


Vir: Program Napovedovalec

Dejstvo, da je napaka pri predvidevanju $g(t)-z(t+h)$ za situacije, ko ne ukrepamo »Nespremenjeno« postavljena na 0, kaže na to, da lahko razvijamo modele, ki lahko upoštevajo komponento šuma, torej podatke v časovni vrsti. Ti za nas niso zanimivi oziroma je predvidljivost zaradi nezadostne vrednosti sprememb premajhna in nas te točke pri trgovanju ne zanimajo. V tem primeru predeli funkcije z nizko predvidljivostjo sprožijo več razredov z vrednostjo »Nespremenjeno«.

### 4.2.3 Rangiranje delnic

Ta način ima kar nekaj prednosti pred prej opisanimi pristopi, saj se pri rangiranju delnic avtomatično upošteva večja količina informacij v primerjavi s poskusom napovedi cene posamezne delnice v času. Informacija o rangu ene delnice odseva, kje med izbranimi delnicami je le-ta uvrščena in je tako povezana z ostalimi delnicami. To dejstvo nam pomaga zajeti tržno anomalijo medsebojnih odvisnosti med delnicami, ki kaže na določeno povezavo med gibanji tečajev delnic.

Pri samem rangiranju posamezne delnice v skupini delnic lahko izberemo različne kriterije na podlagi katerih razvrstimo delnice. Najprimerneje je razvrščati delnice na podlagi donosnosti delnic, doseženih v preteklosti saj lahko na podlagi tržne anomalije učinka preobratov upoštevamo povezavo uspeha delnice v prihodnosti z preteklo uspešnostjo. Donosnost posamezne delnice izračunamo na sledeč način:

$$R_k^m(t)=\frac{y_m(t)-y_m(t-k)}{y_m(t-k)}. \qquad (36)$$

$R_k^m(t)$ = $k$-dnevni donos za delnico $m$

$y_m(t)$ = cena delnice $m$ na dan $t$

$y_m(t-k)$ = cena delnice $m$ prejšnjega dne

Na podlagi $k$-dnevne donosnosti $R_k$ za posamezno delnico po posameznih dneh v določenem obdobju potem izračunamo rang posamezne delnice $A_k^m(t)$ v primerjavi z ostalimi delnicami za isti dan. S tem delnico, ki je dosegla največji donos, postavimo na prvo mesto, tisto, ki je dosegla najslabši donos, pa ima zadnje mesto.

Tako rangirane vrednosti je smiselno normalizirati v tako obliko, da da dobimo za delnico ki je uvrščena na sredino vrednost ranga 0, maksimalna in minimalna vrednost pa v razponu med +0,5 in -0,5. S tem pridobimo uniformirano porazdelitev vrednosti rangov ter srednjo vrednost ranga 0, kar pomeni, da je delnica s pozitivnim rangom nadpovprečno, tista z negativnim rangom pa podpovprečno donosna v tem dnevu. Vrednost ranga $A_k^m(t)$ za $k$-dnevni donos za posamezno delnico $m$ pridobimo na sledeč način (Hellstrom, 2000):

$$A_k^m(t) = \frac{F\left\{ R_{ik}(t) \mid R_{mk}(t) \geq R_{ik}(t), 1 \leq i \leq N \right\} - 1}{N-1} - 0{,}5 . \qquad (37)$$

Funkcija $F$ vrne število elementov iz množice donosov delnic za določen dan $R_i(t)$, in sicer tistih delnic, ki so imele tega dne večjo donosnost od naše delnice $m$. Ko dobljeno število delimo s številom vseh delnic manj 1 (naša trenutna delnica) ter od tega odštejemo eno polovico, dobimo rang posamezne delnice med izbranimi delnicami na določen dan po kriteriju donosnosti. Rangirane vrednosti so istočasno porazdeljene med vrednostjo +0,5 in -0,5.

Razvrščanje okoli vrednosti nič je primerno, ko poskušamo napovedati rang posamezne delnice v prihodnosti. Če nam namreč v trenutku $t$ za delnico $m$, za katero ugotovimo $A_k^m(t+k) > 0$, kjer je $k > 0$ z napovedjo uspe ustvariti dobiček, je to podobno kot, če uspemo identificirati delnico, za katero ugotovimo donosnost $R_k(t+k) > 0$.

Ko poskušamo predvideti $k$-dnevni rang delnice za $k$ dni vnaprej moramo formulirati funkcijo (model) $g_{m,}$ tako da je:

$$A_h^m(t+h) = g_m(I_t), \qquad (38)$$

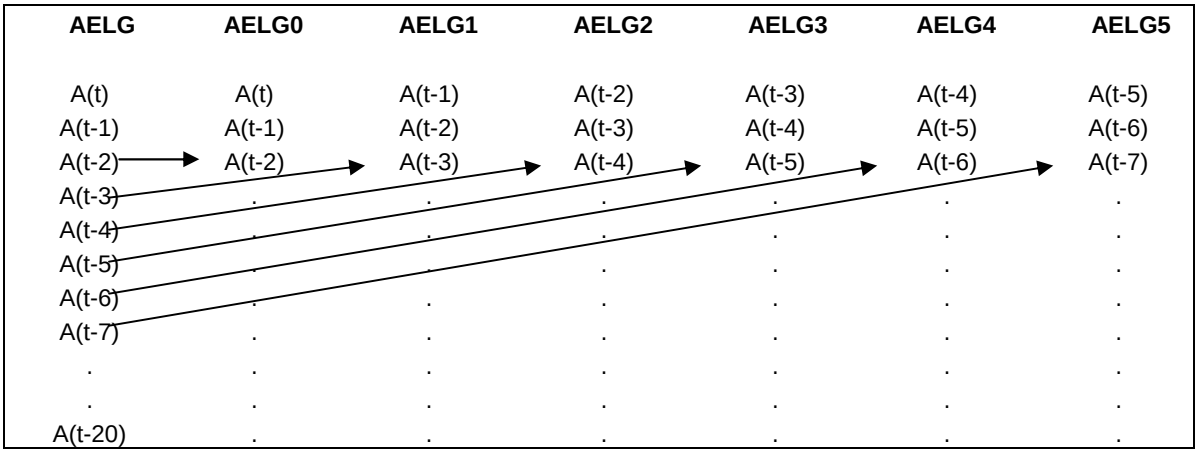
kjer je $I_t$ vrsta informacij, ki so dostopni v času $t$. Kot informacijo lahko uporabimo donosnost delnic $R_k^m(t)$, rang $A_k^m(t)$, obseg prodaje itd. Za povzemanje učinka preobratov je potrebno »povezati« prihodnje gibanje delnice z njeno preteklo uspešnostjo. Tako je najprimerneje upoštevati odvisnost med bodočim rangom $A_h^m(t+h)$ in trenutnim rangom z donosom za več dni nazaj $A_1^m(t)$, $A_2^m(t)$, $A_3^m(t)$, $A_4^m(t)$, $A_{20}^m(t)$. Torej formuliramo zgornjo funkcijo na sledeč način:

$$A_h^m(t+h) = g(\ A_1^m(t)\ ,\ A_2^m(t)\ ,\ A_3^m(t)\ ,\ A_4^m(t)\ ,...,\ A_{20}^m(t)\ )\ . \qquad (39)$$

Za funkcijo $g$ lahko izberemo nevronsko mrežo, odločitvena drevesa ali pa kakšno preprostejšo funkcijo (model). Za več izbranih delnic moramo za vsako posebej zgraditi poseben model, ki jih ocenimo in nato primerjamo med seboj. Izberemo tisto delnico, katere napovedani rang je najvišji in za katero lahko rečemo, da je napoved najbolj zanesljiva.

Da bi zgradili modele, je potrebno pred tem za vsako posamezno delnico v skladu s formulo št. 39 oblikovati podatke v obliki časovne vrste po sledečem postopku. V prvem koraku iz ene časovne vrste rangov dnevnih rangov za eno delnico ustvarimo večje število časovnih vrst s potrebnim zamikom (slika 16).

**Slika 16: Rangirne vrednosti posameznih delnic v časovnih vrstah**




Vir: Lastni prikaz

Pri tem rang delnice, v tem primeru gre za delnico Aerodroma Ljubljana (AELG), razvrstimo v zamaknjene časovne vrste z zamikom enega dneva. Spremenljivko $A_h^m(t+h)$, torej bodoči rang, ki predstavlja razred, je smiselno poenostaviti, in sicer oblikovati samo dve možni vrednosti, ki naj jih ta spremenljivka zavzame (Slika 17).

Glede na to, ali se rangi nahajajo pod ali pa nad vrednostjo 0 (predznak ranga), jih uvrstimo med podpovprečne (POD) oziroma nadpovprečne (NAD).

**Slika 17: Oblikovanje razreda v časovni vrsti podatkov posamezne delnice**

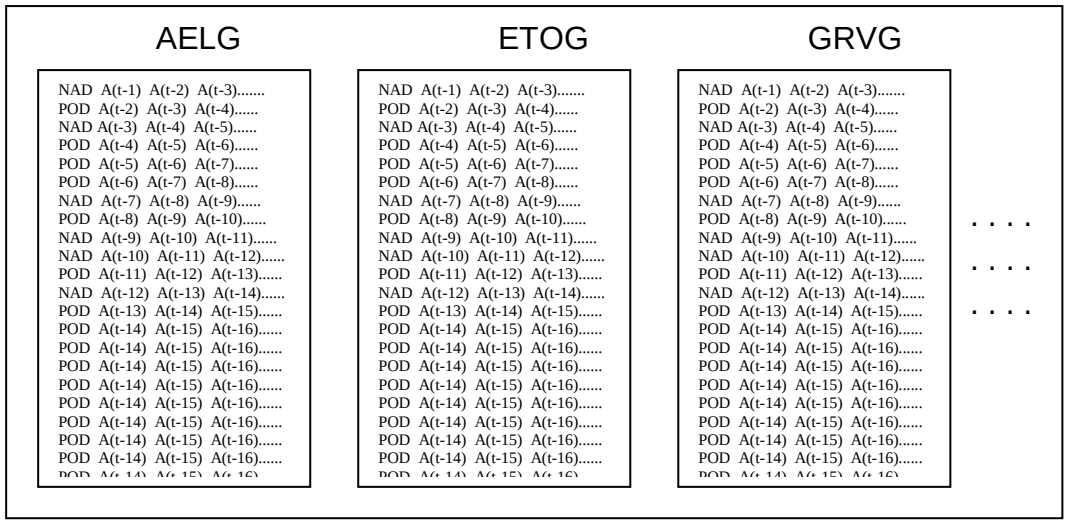
| AELG | AELG0 | AELG1 | AELG2 | AELG3 | AELG4 | AELG5 |
|---|---|---|---|---|---|---|
| A(t) | NAD | A(t-1) | A(t-2) | A(t-3) | A(t-4) | A(t-5) |
| A(t-1) | POD | A(t-2) | A(t-3) | A(t-4) | A(t-5) | A(t-6) |
| A(t-2) | NAD | A(t-3) | A(t-4) | A(t-5) | A(t-6) | A(t-7) |
| A(t-3) | . | . | . | . | . | . |
| A(t-4) | . | . | . | . | . | . |
| A(t-5) | . | . | . | . | . | . |
| A(t-6) | . | . | . | . | . | . |
| A(t-7) | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| A(t-20) | . | . | . | . | . | . |

Vir: Lastni prikaz

Na ta način omogočimo povezavo med preteklimi rangi delnice in potencialnimi nadpovprečnimi rangi, ki jih tako lahko napovedujemo. Obenem pa smo tako omogočili iskanje vzorcev v rangih, katerih pogostost in s tem verjetnost ponovitve je večja, kot če bi poskusili na primer iskati delnice z najvišjim rangom. To pomeni, da je pomembneje najti delnico, za katero lahko z veliko verjetnostjo trdimo, da bo jutri dosegla nadpovprečen donos, kot pa najti delnico, za katero lahko z relativno majhno verjetnostjo trdimo, da bo jutri dosegla največji donos (torej bo najboljša). V primeru manjše verjetnosti imamo namreč večjo možnost, da se zmotimo in se tako lahko zelo hitro znajdemo na negativni strani rangirne lestvice.

Po drugi strani pa je združitev razredne spremenljivke $A_h^m(t+h)$ v dve vrednosti smiselna tudi zato, ker je v primeru ljubljanskega kapitalskega trga količina podatkov, ki je uporabna za analizo, relativno majhna glede na to, da sega zgodovina Ljubljanske borze šele dobro desetletje nazaj. To je zelo malo v primerjavi z zahodnimi borzami, ki dajejo na razpolago desetletja stare zgodovinske podatke. Pravzaprav bi v primeru Ljubljanske borze lahko kot kvalitetne podatke upoštevali šele objavljene dnevne tržne cene za štiri do pet let nazaj glede na to, da se je precej velikih podjetij dokončno olastninilo in vpisalo na borzo šele v tem času. Za število dni, za katere vnaprej poskušamo napovedovati, katera delnica je potencialno uspešna $(h)$, je privzet samo en dan, ker se odvisnosti med rangi s povečevanjem časovne razdalje drastično zmanjšujejo. Podatke o rangiranju delnic, kot so prikazani na sliki 17, je potrebno pripraviti za vse izbrane delnice (glej sliko 18).

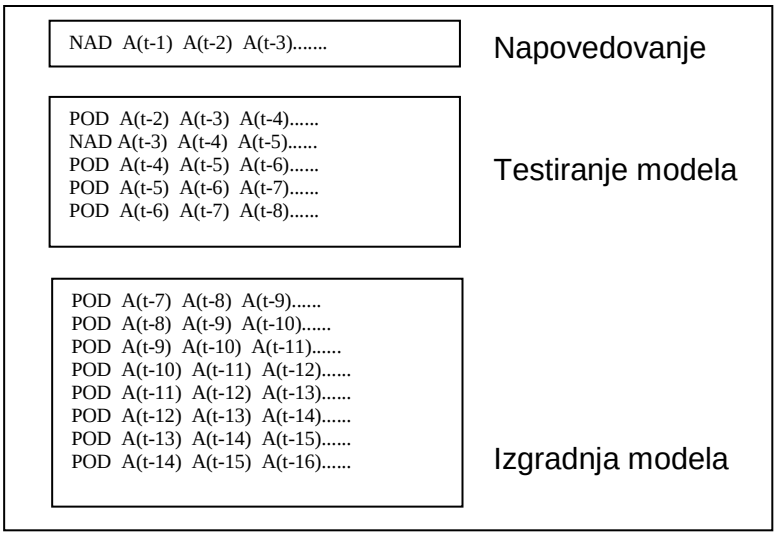
**Slika 18: Oblikovanje časovne vrste za vsako delnico posebej**




Vir: Lastni prikaz

Ko imamo tako urejene podatke, jih lahko obdelamo s pomočjo algoritmov za strojno učenje, in sicer je smiselno podatke razdeliti v tri dele (slika 19).

**Slika 19: Razdelitev podatkov na segmente za testiranje in predvidevanje**




Vir: Lastni prikaz

Če primerjamo opisane tri koncepte napovedovanja, koncept časovnih vrst, izgradnja trgovalnih pravil in pa rangiranje delnic, sta slednji dve boljši zaradi uporabe donosnosti. Če primerjamo donosnost delnic s ceno oziroma poskušamo napovedovati direktno cene delnic, se izkaže, da ima uporaba donosnosti več prednosti:

- donosnosti delnic imajo sorazmerno konstanten razpon, tudi ko se podatki raztezajo skozi več let. Cene lahko očitno variirajo veliko bolj in s tem otežujejo kreiranje pravilnega modela za daljše časovno obdobje,
- donosnosti delnic lahko primerjamo med seboj,
- lažje je oceniti algoritem za predvidevanje donosnosti z izračunom točnosti predvidevanja predznaka donosnosti.

Dodatno, pa je način napovedovanja s pomočjo rangiranja delnic še posebno zanimiv zato, ker lahko bolj natančno povzamemo (modeliramo) medseboje odvisnosti med gibanjem cen različnih delnic.

## 4.3 Učinkovitost napovedovanja

Preden se lotimo optimiranja parametrov algoritmov za predvidevanje, je potrebno razrešiti tri dileme:

- Kakšno učinkovitost oziroma rezultate želimo doseči in kakšne mere učinkovitosti so za nas relevantne?
- Kateri način je najboljši za oceno učinkovitosti napovedovanja?
- S čim naj primerjamo učinkovitost našega sistema (algoritma), da lahko rečemo, da zadostuje in da je dovolj dober?

Splošen problem, ki ga srečamo pri finančnem napovedovanju, je nestacionarna narava finančnih procesov, saj tako učinkovitost tržne strategije vlagatelja variira skozi čas zaradi spremenljivih pogojev, ki zadevajo finančni trg. Neka strategija je lahko uspešna v času, ko se trg nahaja v trendnem gibanju, medtem ko je druga bolj uspešna v netrendnem času. Četudi rešujemo tovrstne probleme v času modeliranja, pa se problemi pojavijo v času preizkušanja na zgodovinskih podatkih.

Merjenje učinkovitosti napovedovanja nekega zgrajenega modela, je potreben v dveh fazah razvojnega cikla (Witten, Eibe, 2000, str. 122).

1. faza: modeliranje algoritma za napovedovanje, kjer gre za izbiro modela oziroma za optimalne nastavitve parametrov v modelu.
2. faza: preizkus zgrajenega algoritma na zgodovinskih podatkih, s katerim ugotavljamo, ali algoritem služi zastavljenemu namenu.

V prvi fazi merimo učinkovitost napovedovanja z namenom reševanja težav s preveliko prilagoditvijo modela učnim podatkom. Prilagajanje parametrov učnega algoritma z namenom doseganja najboljše učinkovitosti napovedovanja nas privede do situacije, ko algoritem preveč prilagodimo učnim podatkom.

V drugi fazi, ko merimo učinkovitost algoritmov za predvidevanje vrednosti v časovni vrsti, pravzaprav primerjamo vrednosti, ki smo jih predvideli s tistimi, ki so se dejansko zgodile (testni podatki). Predvidene vrednosti delniških tečajev v časovni vrsti lahko predstavimo kot $\hat{y}(t)$, pri čemer velja $t = 1,...,N$, dejanske vrednosti cen delnic pa $y(t)$. Prav tako lahko za namene predvidevanja donosov delnic časovno vrsto njihovih predvidenih vrednosti izrazimo kot $\hat{R}(t)$, pri čemer prav tako velja $t = 1,...,N$, dejanske vrednosti donosov delnic pa $R(t)$. Pri predvidevanju vrednosti, ki se bodo zgodile v prihodnosti, moramo vedno vzeti določen časovni korak $h$, za katerega predvidevamo vrednosti vnaprej. Tako predvidene vrednosti $y'(t)$ in $R'(t)$ ustvarimo v časovnem trenutku $t - h$.

Za merjenje učinkovitosti prej omenjenih metod napovedovanja lahko uporabimo različna merila za časovne vrste in tehnično analizo se najprimerneje izkažejo meritve vsote odklonov od pravih vrednosti. Pri tem je ocenjevanje algoritmov, ki temeljijo na trgovalnih pravilih, v tem, da je število proizvedenih signalov za trgovanje razmeroma majhno, kar ne omogoča ravno najboljše statistične osnove za ocenjevanje učinkovitosti trgovanja. Stopnja pravilno napovedanega predznaka spremembe je morda zanimivejši kriterij, še posebno pri napovedovanju s pomočjo rangiranja delnic, kjer lahko dobro ocenimo učinkovitost izbiranja delnic s pozitivno donosnostjo. Vedno pa lahko seveda uporabimo kot kriterij dobiček, ki ga s posamezno metodo uspemo ustvariti v nekem obdobju ali pa primerjamo učinkovitost algoritma za napovedovanje z donosnostjo borznega indeksa.

### 4.3.1 Vsota kvadratov odklonov

Vsota kvadratov odklonov vrednosti, ki jih napove algoritem pri testiranju od testnih dejanskih vrednosti, je kriterij, ki se v statistiki pogosto uporablja za oceno učinkovitosti nekega matematičnega modela. V našem primeru lahko ta kriterij učinkovitosti uporabimo pri napovedovanju cen delnic $y$, pri čemer lahko kriterij označimo kot koren vsote kvadratov napak (angl. Root Mean Square Error, RMSE) in ga lahko opredelimo na sledeč način:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=h}^{N}\left(y(t) - \hat{y}(t)\right)^2} \quad . \tag{33}$$

Prav tako lahko merilo najmanjše vsote kvadratov napak uporabimo za napovedovanje donosov delnic, ki ga lahko definiramo kot:

$$RMSE_R = \sqrt{\frac{1}{N}\sum_{t=h+1}^{N}\left(R(t) - \hat{R}(t)\right)^2} \tag{34}$$

### 4.3.2 Stopnja pozitivnih donosnosti

Dostikrat se lahko pri predvidevanju namesto natančne vrednosti gibanja zadovoljimo že s pravilnimi smermi gibanja časovne vrste. To pomeni, da nas zanima samo dejstvo, ali se je časovna vrsta dvignila ali padla. Takrat je najbolje predvideti kar predznak donosa (Hellstrom, 1998). Stopnja zadetkov nam tako pove, kako pogosto smo pravilno predvideli predznak donosa. Izračunamo ga kot razmerje med številom pravilno predvidenih vrednosti in številom vseh sprememb v časovni vrsti. Pri tem je zelo pomembno, da ne upoštevamo vrednosti, ko ni bilo sprememb, torej ko je cena v času ostala nespremenjena.

Razlog za izogibanje predvidevanju nespremenjenih vrednosti, torej vrednosti, ko je bil donos $R(t)$ nič, je ta, da se moramo v primeru vključitve ničtih vrednosti odločiti, kako bomo obravnavali sledeče kombinacije predvidenih in dejanskih vrednosti. Ali jih bomo obravnavali kot zadetke (angl. hits) pri napovedovanju ali ne.

**Tabela 2: Razpored kombinacij zadetkov z ničtimi vrednostmi**

| | |
|---|---|
| $\hat{R}(t) = 0$ | $R(t) > 0$ |
| $\hat{R}(t) = 0$ | $R(t) < 0$ |
| $\hat{R}(t) = 0$ | $R(t) = 0$ |
| $\hat{R}(t) > 0$ | $R(t) = 0$ |
| $\hat{R}(t) < 0$ | $R(t) = 0$ |

Vir: Hellstrom, 1998

Ne glede na izbor, s katerim bomo obravnavali omenjene kombinacije vrednosti, na katere lahko naletimo, bomo donose vedno obravnavali asimetrično. Ker pa vrednosti enodnevnih donosov lahko predstavljajo tudi do 20 odstotkov primerov v tipični časovni vrsti delnic, bi lahko povzročili, da delež cenovnih dvigov zavzame več ali pa manj kot 50 %. Tak rezultat lahko bistveno vpliva na naravo naključnosti časovne vrste, torej na koncept naključnega hoda. Z odstranitvijo ničtih vrednosti donosov, tako iz predvidenih kot tudi dejanskih vrednosti, se delež primerov, ko je donos v porastu, res približa deležu 50 odstotkov. Takrat lahko v primeru enodnevnih donosov (h=1), ko je stopnja zadetkov R(t) bistveno višja od 0,5, res rečemo, da smo uspeli pri predvidevanju predznaka donosov.

### 4.3.4 Donosnost strategije

Na podlagi predvidevanj algoritma lahko spremljamo tudi dejanske donose v smislu simulacije trgovanja. V takem primeru je pametno, da spremljamo celoten donos,

dosežen na podlagi imaginarnega trgovanja skozi določeno podatkovno obdobje. Tako ustvarjena vrednost se lahko pogosto predstavi v obliki t. i. diagrama sredstev. Vendar pa se pri tovrstnem kriteriju pojavlja določena anomalija, saj rezultati začetnega trgovanja bolj vplivajo na končni donos, kot pa trgovanje ob koncu obdobja. Razlog tiči v kumulativni naravi trgovanja, kajti po tej metodi se začetni dobički reinvestirajo in se tako večkrat pojavljajo v skupaj nakopičenem »bogastvu«, ki je rezultat trgovanja v celotnem obdobju. Osnovna oblika trgovanja predstavlja samo enkraten nakup na začetku in prodajo na koncu trgovalnega obdobja. Taka »kupi in zadrži« (angl. buy & hold) strategija je ena od najstarejših primerjalnih metod učinkovitosti napovedovanja delnic.

Da bi se izognili kumulativnemu efektu, ki se pojavlja pri celotnem donosu, doseženem na podlagi simulacije trgovanja, bi lahko izračunavali povprečen donos, dosežen na dnevni ravni. Ta donos lahko ocenimo z metodo analize časovnih vrst tako, da upoštevamo trgovanje pri vsakem časovnem koraku časovne vrste v smeri predvidene spremembe (Hellstrom, 1998).

$$P = 100 * \frac{1}{N} * \sum_{t=h+1}^{N} \frac{predznak(y(t) - y(t-h))(y(t) - y(t-h))}{y(t-h)}, \qquad (35)$$

kjer je:

$$predznak(x)\begin{cases} 1 & če\ je\ \ x > 0 \\ -1 & če\ je\ \ x < 0 \\ 0 & če\ je\ \ x = 0 \end{cases}.$$

Na ta način seštejemo vse dobičke in izgube med pravilno in nepravilno predvidenimi vrednostmi delnice.

### 4.3.5 Primerjava uspešnosti modela z borznim indeksom

Kadar simuliramo trgovanje na podlagi napovedi algoritma, se kot normalna ocena lahko uporablja tudi primerjava s kakšnim od borznih indeksov, ki se izračunavajo na podlagi večih delnic. Tovrstne primerjave se najpogosteje uporabljajo kot kriterij uspešnosti tudi pri institucionalnih investitorjih. Upravljavec premoženja, ki doseže kapitalske donose, ki presegajo donosnost borznega indeksa, je ponavadi tudi nagrajen. Vzajemni skladi so še posebej podvrženi tovrstnim kriterijem, saj lahko izgubijo stranke, če njihova uspešnost pade pod dosežek borznega indeksa. To je tudi eden od razlogov, zakaj lahko več vzajemnih skladov, ki trgujejo na istem trgu, dosega zelo podobne rezultate.

## 4.4 Zgradba programa Napovedovalec

Iz opisa algoritma za rangiranje delnic, ki smo ga zbrali za praktično izvedbo, lahko vidimo, da gre za zelo obsežen postopek. Potrebna je priprava zbranih podatkov, izgradnja večih modelov, ter izbira najboljše napovedi na podlagi zgrajenih modelov. Vse to je potrebno izvesti za več dni in pri tem uporabiti različne algoritme strojnega učenja. Najustreznejša rešitev tega obsežnega problema je izdelava programske rešitve, s katero bi lahko opisane postopke čim bolj avtomatizirali in istočasno omogočili oceno uspešnosti trgovanja na podlagi opisane teorije predvidevanja rangov delnic. Ker je poudarek magistrskega dela na sami izdelavi algoritma za razvrščanje ne pa tudi na zahtevni izgradnji samih algoritmov za strojno učenje smo se odločili uporabiti zunanjo knjižnico že implementiranih metod strojnega učenja.

Računalniški program Napovedovalec, s katerim bi želeli preizkusiti teorijo predvidevanja ranga delnice, je imel pri načrtovanju zastavljene sledeče kriterije:

- vnos podatkov v sistem naj bi bil zaradi velike količine podatkov kar najbolj avtomatiziran,
- sistem naj bi sam zgradil klasifikacijsko funkcijo oziroma model,
- klasifikacijo zahtevanega podatkovnega primera naj bi izvedel sistem avtomatično,
- število klasifikacij za preizkus naj bi bilo poljubno in vnaprej dano s strani uporabnika,
- na podlagi odločitve naj bi sistem tudi sam izračunal učinkovitost odločitve, kar pomeni izračun donosnosti,
- sistem naj bi podrobno predstavil razloge za odločitev, kar pomeni, da naj bi predstavil porazdelitve verjetnosti za vsako delnico posebej in po možnosti prikazal zgrajen model,
- prikazal naj bi tudi povzetek porazdelitev verjetnosti, iz katerega bi se uporabnik lahko prepričal o pravilnem delovanju sistema,
- sistem naj bi tudi grafično prikazal gibanje vrednosti delnic, in sicer za vsako posebej, kar uporabniku med drugim omogoča oceno uspešnosti napovedovanja.
- Glede na to, da je knjižnica WEKA, ki nosi breme strojnega učenja, zasnovana za izvajanje v okolju Java, bi bilo najprimerneje sistem za napovedovanje programirati v istem programskem jeziku.

### 4.4.1 Programski paket WEKA

WEKA[3] (angl. Waikato Environment for Knowledge Analysis) je programsko okolje, ki združuje knjižnice za analizo podatkov z metodami za rudarjenje podatkov. Razvit je bil na univerzi Waikato na Novi Zelandiji. Gre za projekt s prosto dostopno kodo, napisan v programskem jeziku Java. Zaradi svoje zasnove je programski paket WEKA moč uporabljati na več načinov, in sicer kot končen produkt, torej programski paket, ki je namenjen analizi podatkov ali pa lahko zaradi njegove odprte kode in dobro dokumentiranih razredov uporabimo WEKO kot knjižnico razredov v programu, ki ga sami napišemo.

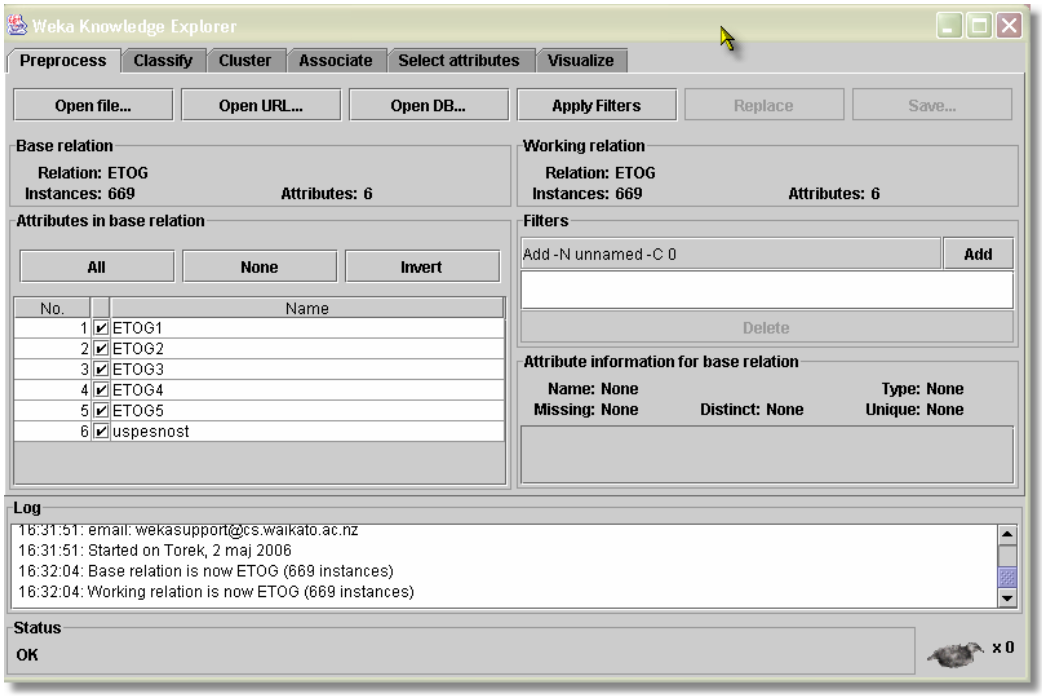
Ta drugi način je za rešitev našega problema, torej konstrukcije algoritma za napovedovanje, še posebej uporaben. WEKA ima namreč vse potrebne metode strojnega učenja že implementirane in pripravljene za uporabo v našem programu Napovedovalec. Gre za lepo strukturirano knjižnico, ki zaradi urejene strukture razredov in njihovih metod omogoča njeno enostavno uporabo. Metode, kot je npr. `distributionForInstance()`, vrnejo vedno enako strukturiran rezultat, ne glede na to iz katerega objekta jih pokličemo. To nam omogoča vedno enak sistem odločanja glede na verjetnosti za posamezno izmed vrednosti, ki jih razred lahko zavzame (podpovprečen, nadpovprečen). Zato lahko tudi naše razrede oblikujemo tako, da sprejemajo to poenoteno strukturo knjižnice.

Kadar uporabljamo WEKO kot programski paket, imamo v novejših različicah na voljo uporabniški vmesnik, pri katerem je možno učne metode, torej metode strojnega učenja, uporabiti na podatkih, iz katerih na ta način izvlečemo informacije, ki opisujejo te podatke. Pri tem je možno uporabiti več učnih metod hkrati in tako z eksperimentiranjem na tak način zgraditi več različnih modelov, s katerimi v nadaljnjem testiranju pridemo do rezultata, kateri model najbolje opisuje podatke in nam s tem omogoča predvideti dogajanja v bodoče. Pred analizo je podatke možno ustrezno pripraviti, transformirati in prečistiti s pomočjo t. i. filtrov. Primarna oblika podatkov, potrebna za njihovo obdelavo, je t. i. format ARFF, ki zahteva ločeno predstavitev atributov od podatkov samih. Atribute je sistemu potrebno predstaviti z njihovimi imeni ter vrednostmi, katere lahko posamezni atribut zavzame. Podatki morajo slediti vrstnemu redu pri predstavitvi atributov in biti pri tem ločeni z vejico (glej prilogo G).

Pri uporabi posameznega modela imamo na voljo Explorer, pri eksperimentiranju z večimi modeli hkrati pa je možno izbrati Experimenter. V uporabniškem vmesniku Explorer je mogoče izbrati več opcij (glej sliko 20):

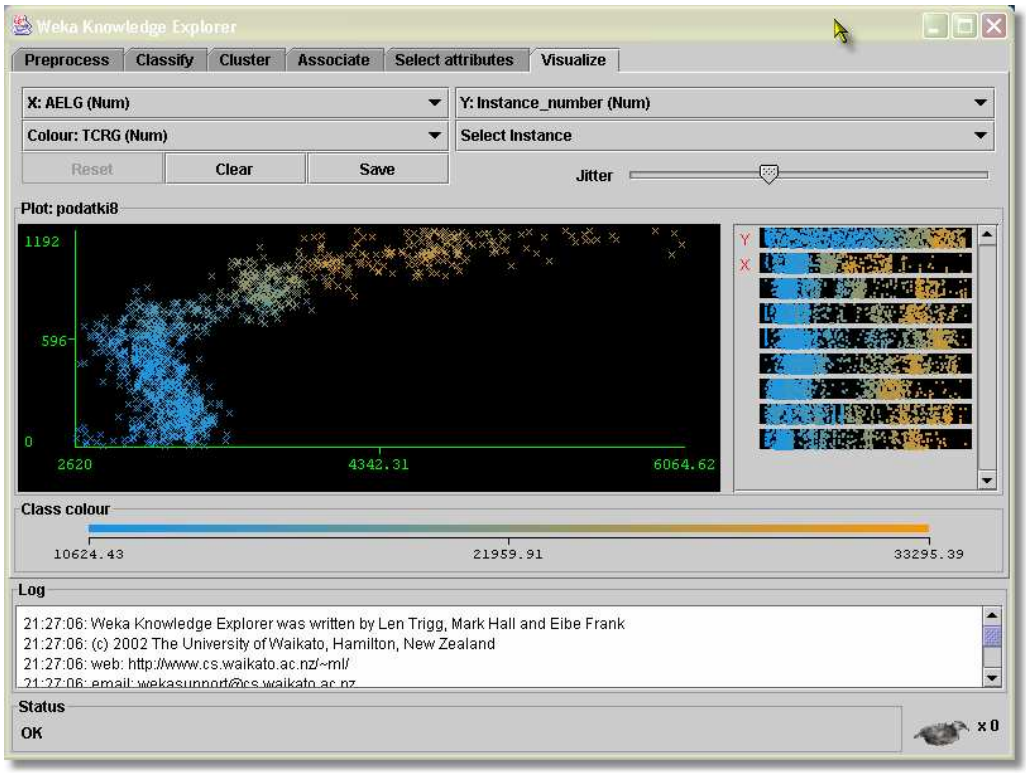
[3] Podrobnejši opis programa WEKA je moč najti v knjigi Wittna in Franka (2000).

**Slika 20: Okno za predprocesiranje podatkov**




Vir: Programski paket WEKA

**Slika 21: Okno za predogled podatkov za obdelavo**




Vir: Programski paket WEKA

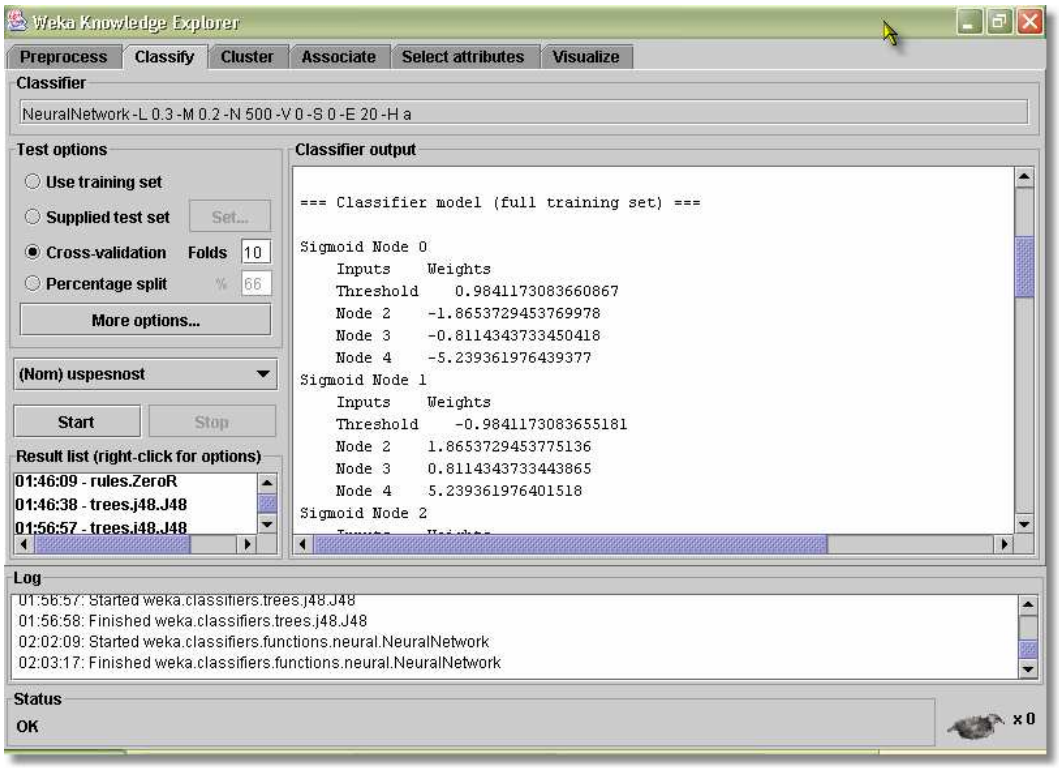
predprocesiranje podatkov (zavihek Preprocess), izgradnja modelov (zavihek Classify), razvrščanje objektov v skupine (zavihek Cluster), iskanje povezovalnih pravil (zavihek Associate), izbor spremenljivk (zavihek Select attributes) in prikaz

podatkov (zavihek Visualize). Z izbiro zavihka preprocess se nam prikaže panel, ki predstavlja izhodišče za rudarjenje podatkov (glej sliko 20).

Tukaj lahko izberemo podatke za obdelavo (angl. dataset) in si ogledamo karakteristike posameznih spremenljivk (atributov). Te spremenljivke je mogoče vključiti v analizo ali izključiti iz nje s filtri (gumb Apply Filters). Grafični prikaz podatkov v oknu Visualize (glej sliko 21) nam omogoča vizuelni pogled na podatke, predvsem njihovo grupiranost okoli določenih vrednosti.

V osrednjem panelu Classify lahko izberemo različne algoritme strojnega učenja, s katerimi program zgradi model za napovedovanje. Želeni algoritem izberemo v okvirju Classifier. Različnim algoritmom ustrezajo različni parametri, ki jih nastavimo v podoknu za nastavitev parametrov. Za testiranje modela lahko izberemo metodo medsebojnega primerjanja (angl. cross-validation) ali pa izberemo za testiranje modela posebno skupino podatkov. Zgrajeni model in parametri uspešnosti napovedovanja se izpišejo v oknu Classifier output (glej sliko 22).

**Slika 22: Okno za prikaz zgrajenega modela v programskem paketu WEKA**




Vir: Programski paket WEKA.

Drugi način uporabe sistema WEKA je v obliki knjižnice razredov, katere lahko uporabimo v okviru svojega programa. WEKA je, kot je bilo omenjeno že zgoraj, napisana v objektno orientiranem programskem jeziku Java. Vsak Java program je implementiran kot razred (angl. class). V objektno-orientiranem programiranju razred

predstavlja zbirko spremenljivk (angl. variables) skupaj z metodami (angl. methods), ki operirajo s temi spremenljivkami. Skupaj definirajo objekt (angl. object), ki je ustvarjen na podlagi tega razreda. Objekt je preprosto primer razreda, ki ima spremenljivke postavljene na konkretne vrednosti.

V Javi se razredi zaradi organiziranosti največkrat po funkcionalnosti združujejo v pakete (angl. packages). Tovrstna organiziranost poleg urejenosti omogoča tudi izogibanje nesporazumom pri uporabi razredov, ki bi lahko imeli enako ime.

WEKA se distribuira skupaj s svojo izvorno kodo in, kar je še pomembneje, z odlično dokumentacijo. Iz dokumentacije je razvidno, da je vsak učni algoritem, ki jih je iz verzije v verzijo vedno več, implementiran kot razred. Vendar pa je zaradi razvojne naravnanosti sama funkcionalnost algoritmov poleg algoritemskega razreda razdeljena na več razredov v skladu z objektno orientiranostjo. Funkcionalnost, ki je skupna večini algoritmom, je namreč implementirana v enem razredu ali vmesniku (angl. interface) in ta razred nato uporabljajo različni drugi razredi, s čimer se je razvojni tim izognil večkratni implementaciji iste funkcionalnosti na različnih koncih izvorne kode. Ta prednost, ki se kaže v možnosti lažjega nadaljnjega razvoja in je pomembna pridobitev objektne orientiranosti, pa po drugi strani povzroča nekaj težav, ko poskušamo razumeti delovanje sistema s pomočjo dokumentacije, saj je potrebno iskano funkcionalnost, implementirano v posameznih metodah posameznega razreda, iskati med vsemi razdrobljenimi razredi celotnega sistema. Zaradi te možnosti je smiselno uporabiti že izdelane algoritme strojnega učenja za analizo podatkov delniških tečajev ter poskus predvidevanja rangov delnic v bodočnosti. Vendar je potrebno preučiti tudi sestavo sistema, njegove razrede, polja in metode, če jih želimo v našem programu uporabiti.

Osnovna funkcionalnost sistema WEKA, ki jo uporabljajo vsi algoritmi strojnega učenja, je implementirana v osrednjem paketu (`weka.core`). Predvsem so pomembni razredi `Attribute`, `Instance` in `Instances`. Objekt razreda `Attribute` predstavlja atribut in vsebuje ime atributa, njegov tip ter v primeru nominalnega atributa njegove možne vrednosti (velik, majhen, rdeč, zelen ...). Objekt razreda `Instance` pa vsebuje vrednosti po atributih in tako predstavlja posamezen podatkovni primer. V primeru razreda `Instances` pa gre za skupek podatkov, katere lahko pripravljene po instancah (primerkih) podamo algoritmom v obdelavo. Razred `Instances` ima med drugim tudi zelo pripravno metodo `Instances(Reader)`, ki je v bistvu konstrukcijska metoda, s katero naredimo nov objekt tipa `Instances`. Pri tem uporabimo ukaz `new` in lahko že pri kreiranju objekta tipa `Instances` preberemo datoteko s podatki iz trajnega spomina s pomočjo standardnega Java razreda `FileReader`. Tako imamo na razpolago

pripraven način za hiter vnos podatkov v sistem, pri čemer so lahko podatki organizirani v formatu ARFF (glej prilogo G).

Ko imamo ustvarjene podatkovne primerke v okviru objekta razreda `Instances`, jih lahko obdelujemo s pomočjo učnih algoritmov. Algoritmi so implementirani kot razredi, in sicer v paketu `Classifiers`. Vsaka skupina algoritmov je implementirana v svojem podpaketu. Najbolj pomemben razred v tem paketu je razred `Classifier`, katerega najpomembnejša lastnost je ta, da pravzaprav definira splošno strukturo za klasificiranje. Tako lahko dobimo enako shemo rezultatov iz različnih klasifikatorjev in jih lahko zato med seboj primerjamo. Vsebuje dve metodi, `BuildClassifier()` in `classifyInstance()`, ki pa ju morajo vsi učni algoritmi implementirati. Slednje poteka tako, da so vsi klasifikatorji podrazredi tega razreda in s tem podedujejo obe metodi. Pri dedovanju razreda Classifier ostali učni algoritmi v določeni meri »povozijo« metodi in jih s tem prilagodijo konkretnemu učnemu mehanizmu. Vendarle pa ostane enak vmesnik, preko katerega kličemo omenjeni metodi iz naše izvorne kode, kar nam močno poenostavi in uredi programsko kodo. Drugi pomemben razred je tudi `DistributionClassifier`, ki je podrazred razreda `Classifier` in ima med drugim implementirano metodo `distributionForInstance()`, katera na zahtevo po določeni instanci vrne verjetnostno porazdelitev za možne vrednosti razredov.

Večina razredov iz paketa `Classifier`, torej učnih algoritmov, nam preko te metode, ki jo podedujejo z dedovanjem razreda `Distribution Classifier`, omogoča izračunavanje verjetnostnih porazdelitev za dano instanco, torej podatkovni primer. WEKA verzije 3.3.6, ki je uporabljena tudi v nadaljnji analizi delnic, ima tako implementirane sledeče skupine učnih metod (algoritmov):

- odločitvena drevesa (weka.classifiers.trees),
- odločitvena pravila (weka.classifiers.rules),
- metode najbližjih sosedov (weka.classifiers.lazy),
- nevronske mreže (weka.classifiers.functions.neural),
- verjetnostne porazdelitve – bayesovi klasifikatorji (weka.classifiers.bayes).

Med odločitvenimi drevesi sem za predvidevanje najbolj donosnih delnic uporabljal Odločitveno drevo C 4.5 oziroma njegovo dodelano različico J 48, ki se nahaja v razredu `weka.classifiers.trees.j48`. Objekti, kreirani na podlagi tega razreda, imajo precejšnje število metod, ki jih programer lahko uporablja za izvedbo analize v okviru svojega programa. Omenil bi sledeče:

- `public void buildClassifier(Instances instances)`: Gre za metodo, podedovano iz skupnega vrhnjega razreda `Classifier`, ko pri

objektu tipa J48, ki smo ga ustvarili z njegovo istoimensko konstrukcijsko metodo `J48()`, zgradimo klasifikator na podlagi podatkov, ki smo jih že pripravili v obliki razreda `Instances`, in jih posredujemo metodi pri njenem klicu. Klasifikator predstavlja zgrajeno odločitveno drevo, ki ga z metodo `toString()`, lahko tudi izpišemo oziroma nam je v obliki črkovnega niza na razpolago za izpisovanje po naši želji.

- `public void evaluateModel(Classifier classifier,Instances data):` Ta metoda, ki je sestavni del razreda `weka.classifiers.Evaluation`, je shranjena v podpaketu `Classifiers`. Njena naloga je ovrednotiti določen model, zgrajen po prejšnji metodi. Da bi to metodo sploh lahko uporabili, je potrebno poprej ustvariti objekt razreda `Evaluation`, kar storimo z eno od konstrukcijskih metod. Za vrednotenje določenega modela po tej metodi, ki sem jo tudi uporabljal, je potrebno priložiti tudi testne podatke, na katerih metoda testira zahtevani model (klasifikator).

- `public final double pctCorrect():` Ko imamo enkrat zgrajen objekt, katerega namen je ovrednotenje določenega modela, lahko povzamemo parametre kvalitete klasifikatorja pri klasificiranju testnih podatkov. Najprimernejši pokazatelj je vsekakor delež pravilno klasificiranih podatkov, ki ga s to metodo izrazimo in lahko uporabimo pri odločanju. Metoda nam odstotkovni rezultat vrne v obliki števila.

- `public final double[] distributionForInstance(Instance instance):` Ta metoda nam za obravnavano instanco (podatkovni primer) vrne porazdelitev verjetnosti za vsako možno vrednost razreda. Vrednosti porazdelitev nam, kot je iz opisa metode razvidno, le-ta vrne v obliki enodimenzionalnega polja, kjer je za vsako vrednost, ki jo lahko zavzame razred, priložena še verjetnost za njen nastanek. Tako lahko s pomočjo klasifikatorja, ki smo ga zgradili po zgornji metodi, avtomatično klasificiramo bodoči podatkovni primer.

Tudi za nevronske mreže, ki se nahajajo v paketu `weka.classifiers.functions.neural`, kot tudi za algoritem k-najbližjih sosedov `(Ibk)`velja, da so oblikovane v razrede, ki podpirajo istoimenske metode. Način obdelave podatkov oziroma učenje in izgradnja modela je seveda drugačen, pač v skladu s teoretičnimi osnovami. Pomembno pa je, da so rezultati učenja izraženi vedno na enak način, kar omogoča poenostavljeno uporabo v našem programskem projektu.

### 4.4.1 Opis programa

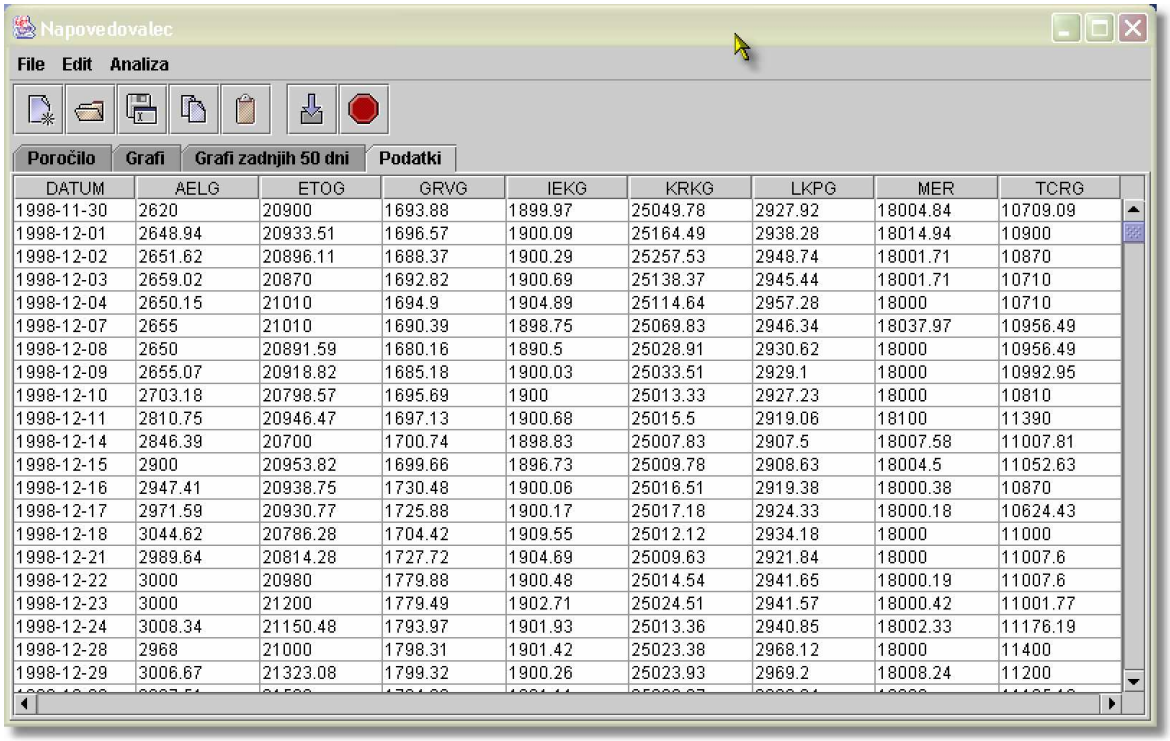
Program, katerega celotna programska koda je v prilogi F, je sestavljen iz dveh razredov, in sicer razreda `Napovedovalec`, ki je okvir programa, ter razreda `InstanceHolder`, ki predstavlja hranilca podatkov. Razred `Napovedovalec` je pravzaprav implementiran kot panel, torej podeduje vse lastnosti razreda `Jpanel`, kar je običajno za okenske programe. Ob zagonu programa je vstopna funkcija »main« zadolžena za kreacijo okvirja (angl. frame) ter kreacijo in umestitev objekta `Napovedovalec` v okvir. Pri tem pa ta funkcija ustvari še hranilca podatkov, torej objekt na podlagi razreda `InstanceHolder`. Pri kreaciji objekta `Napovedovalec` je potrebno prikazati klasične elemente, ki jih uporabnik pričakuje v okenskem grafičnem okolju, to so menijska vrstica, orodna vrstica z gumbi ter delovna površina, kjer poteka prikaz in obdelava podatkov.

Vse to se zgodi ob kreaciji objekta `Napovedovalec`. Prav tako se istočasno ustvarijo razredi, ki opravljajo namenske funkcije na podlagi ukazov, ki jih uporabnik sproža iz menijske vrstice z izbiro menijev ali pa s klikom na gumbe iz orodne vrstice. Ti razredi (angl. listeners), ki »poslušajo« menijske ali orodne ukaze, se ustvarijo takoj ob kreaciji menijev in gumbov. Ti razredi so sledeči:

- `OpenL` – omogoča odpiranje dokumenta, in sicer poročila v obliki tekstovne datoteke,
- `SaveL` – omogoča shranjevanje dokumenta. Namenjen je shranjevanju poročila v obliki tekstovne datoteke,
- `ExitL` – omogoča izhod iz programa,
- `CopyL` – omogoča kopiranje dela ali celote besedila v odložišče za kasnejše lepljenje iz odložišča,
- `CutL` – omogoča izrezovanje dela besedila v odložišče in kasnejše lepljenje tega besedila,
- `PasteL` – omogoča lepljenje besedila, ki je v odložišču.

Dodana sta tudi dva razreda za poslušanje ukazov ter aktivacijo razredov, ki so zadolženi za vnos (angl. Import) podatkov v program ter za zagon analize, torej obdelave teh podatkov z algoritmi strojnega učenja. To sta razreda `ImportL`, ki je zadolžen za kreiranje objekta import ter `AnalizaL,` ki je zadolžen za kreiranje objekta analiza. Razred Import je namenjen uvozu podatkov iz trajnega spomina ter prikazu teh podatkov v grafični obliki (Slika 24). Vnos podatkov se izvede s pomočjo razreda `FileReader`, ki prebrano tekstovno datoteko v formatu ARFF prenese Wekinemu razredu `Instances` oz. je s kreacijsko metodo ustvarjanju objekta na podlagi tega razreda prenesen objekt tipa `FileReader`. Ko imamo tako podatke v obliki instanc enkrat v sistemu, lahko sledi njihova nadaljnja obdelava. V prvi vrsti je sistem zadolžen za prikaz grafov časovnih vrst cen izbranih delnic (slika 25).

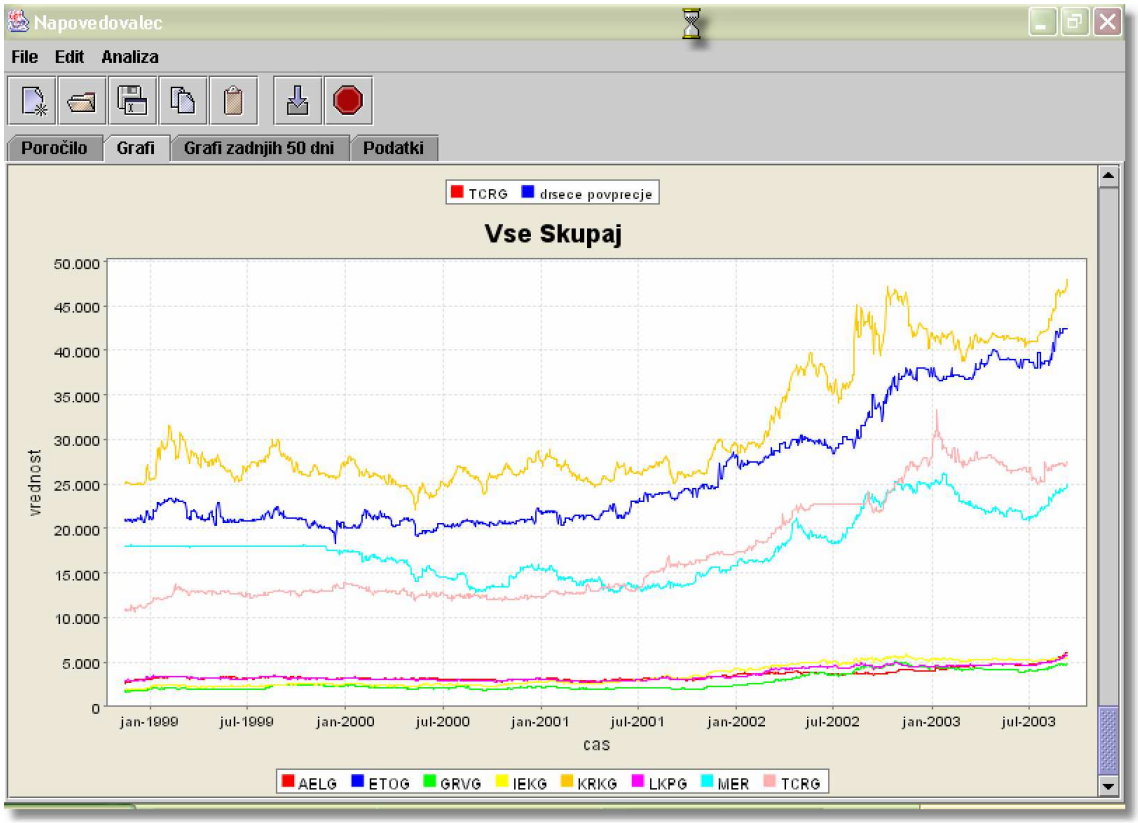
**Slika 24: Tabelarični prikaz podatkov o kotacijah analiziranih delnic**




| DATUM | AELG | ETOG | GRVG | IEKG | KRKG | LKPG | MER | TCRG |
|---|---|---|---|---|---|---|---|---|
| 1998-11-30 | 2620 | 20900 | 1693.88 | 1899.97 | 25049.78 | 2927.92 | 18004.84 | 10709.09 |
| 1998-12-01 | 2648.94 | 20933.51 | 1696.57 | 1900.09 | 25164.49 | 2938.28 | 18014.94 | 10900 |
| 1998-12-02 | 2651.62 | 20896.11 | 1688.37 | 1900.29 | 25257.53 | 2948.74 | 18001.71 | 10870 |
| 1998-12-03 | 2659.02 | 20870 | 1692.82 | 1900.69 | 25138.37 | 2945.44 | 18001.71 | 10710 |
| 1998-12-04 | 2650.15 | 21010 | 1694.9 | 1904.89 | 25114.64 | 2957.28 | 18000 | 10710 |
| 1998-12-07 | 2655 | 21010 | 1690.39 | 1898.75 | 25069.83 | 2946.34 | 18037.97 | 10956.49 |
| 1998-12-08 | 2650 | 20891.59 | 1680.16 | 1890.5 | 25028.91 | 2930.62 | 18000 | 10956.49 |
| 1998-12-09 | 2655.07 | 20918.82 | 1685.18 | 1900.03 | 25033.51 | 2929.1 | 18000 | 10992.95 |
| 1998-12-10 | 2703.18 | 20798.57 | 1695.69 | 1900 | 25013.33 | 2927.23 | 18000 | 10810 |
| 1998-12-11 | 2810.75 | 20946.47 | 1697.13 | 1900.68 | 25015.5 | 2919.06 | 18100 | 11390 |
| 1998-12-14 | 2846.39 | 20700 | 1700.74 | 1898.83 | 25007.83 | 2907.5 | 18007.58 | 11007.81 |
| 1998-12-15 | 2900 | 20953.82 | 1699.66 | 1896.73 | 25009.78 | 2908.63 | 18004.5 | 11052.63 |
| 1998-12-16 | 2947.41 | 20938.75 | 1730.48 | 1900.06 | 25016.51 | 2919.38 | 18000.38 | 10870 |
| 1998-12-17 | 2971.59 | 20930.77 | 1725.88 | 1900.17 | 25017.18 | 2924.33 | 18000.18 | 10624.43 |
| 1998-12-18 | 3044.62 | 20786.28 | 1704.42 | 1909.55 | 25012.12 | 2934.18 | 18000 | 11000 |
| 1998-12-21 | 2989.64 | 20814.28 | 1727.72 | 1904.69 | 25009.63 | 2921.84 | 18000 | 11007.6 |
| 1998-12-22 | 3000 | 20980 | 1779.88 | 1900.48 | 25014.54 | 2941.65 | 18000.19 | 11007.6 |
| 1998-12-23 | 3000 | 21200 | 1779.49 | 1902.71 | 25024.51 | 2941.57 | 18000.42 | 11001.77 |
| 1998-12-24 | 3008.34 | 21150.48 | 1793.97 | 1901.93 | 25013.36 | 2940.85 | 18002.33 | 11176.19 |
| 1998-12-28 | 2968 | 21000 | 1798.31 | 1901.42 | 25023.38 | 2968.12 | 18000 | 11400 |
| 1998-12-29 | 3006.67 | 21323.08 | 1799.32 | 1900.26 | 25023.93 | 2969.2 | 18008.24 | 11200 |

Vir: Programski paket Napovedovalec

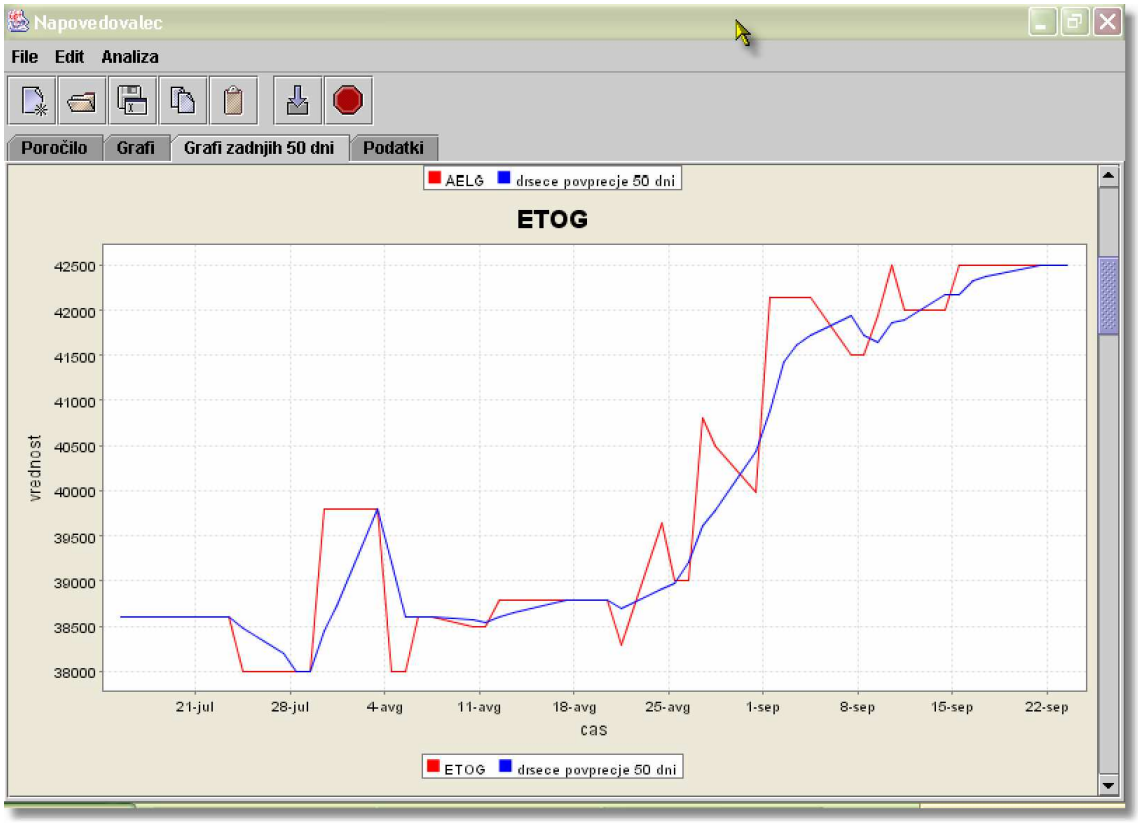
**Slika 25: Grafični prikaz tečajev posameznih delnic za obdobje od 22.6.-23.9.2003**




Vir: Programski paket Napovedovalec

Iz grafov lahko kasneje po analizi primerjamo, kako uspešno se je algoritem odločil za investicijo v določenem dnevu. Poleg časovne vrste cen, ki jo dosega delnica na trgu, program izračuna in pokaže tudi drseča povprečja, s katerimi si lahko pomagamo pri odločitvi za nakup oziroma prodajo delnice. Obe časovni vrsti za posamezno delnico in drseča povprečja so prikazana tudi na skupnem grafikonu (slika 26).

**Slika 26: Gibanja tečaja delnice ETOG in njegove 50-dnevne drseče sredine**




Vir: Programski paket Napovedovalec

Grafi se ustvarijo s pomočjo grafične knjižnice Jfree. Knjižnica med drugim omogoča kreiranje posameznih časovnih vrst ter drsečih sredin na njihovi podlagi. Ker je program zasnovan za napovedovanje poljubnega števila delnic, je potrebno za vsako delnico posebej kreirati časovno vrsto. Ker je časovnih vrst lahko več, je sistem zasnovan tako, da vse časovne vrste spravi v enodimenzionalno polje. Prav tako je potrebno urediti tudi grafe kot objekte v polju.

Razred `Analiza` je osrednji gradnik programa, kjer se izvaja strojno učenje ter predvidevanje na osnovi tako zgrajenih modelov. Program pridobi instančne podatke iz hranilnika podatkov v obliki posebne niti, da ne zasedemo celotnega procesorskega časa in tako za določen čas »zamrznemo« računalnik. Poleg tako pridobljenih podatkov program istočasno pridobi od uporabnika informacije o želenem algoritmu ter številu dni, za katere je potrebno preizkusiti napovedovanje.

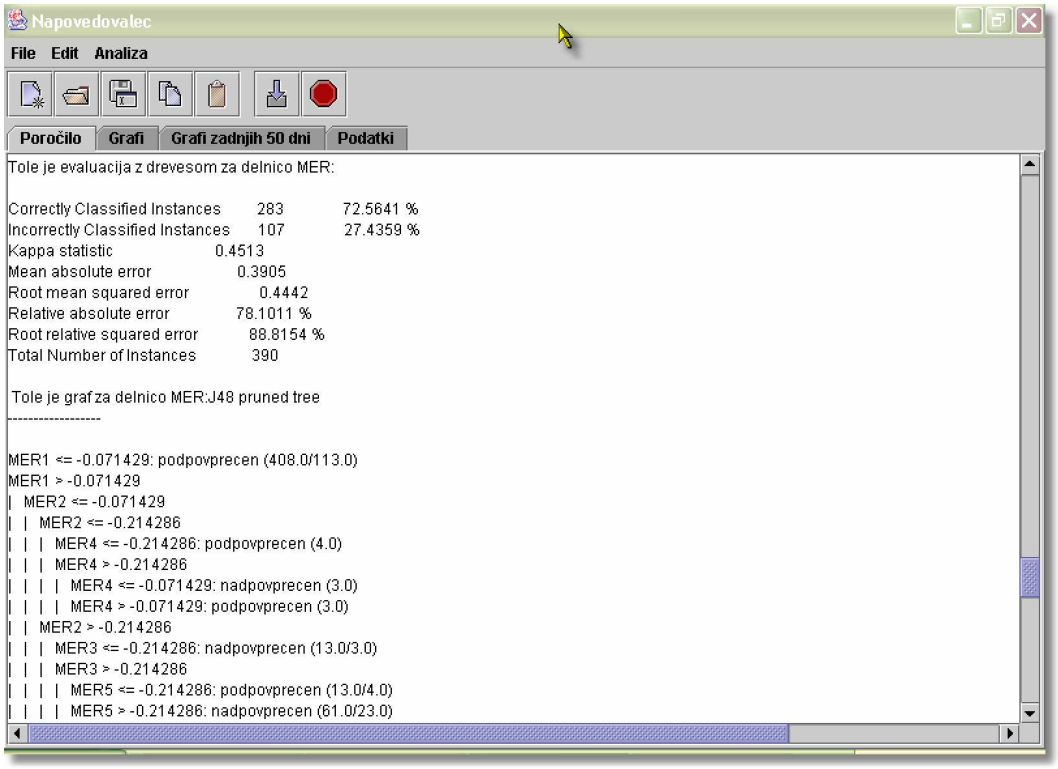
Na tej podlagi je razred `Analiza` pripravljen razdeliti podatke na ustrezne deleže: na tiste za učenje, delež za testiranje ter na delež za napovedovanje. Pri tem je za vsako delnico potrebno kreirati vrednosti donosov ter jih na podlagi donosov za vsak dan posebej rangirati. Poleg tega je potrebno ustvariti tudi zamaknjene časovne vrste tako donosov kot tudi rangov za vsako delnico posebej. Donose, range ter imena delnic je potrebno spraviti v polja, ki so med seboj strukturirano urejena tako, da so med seboj povezane vrednosti na istih mestih v poljih. To je najbolje opraviti znotraj ene zanke, da se pri razvrščanju v polja ne zmotimo. Nato se na podlagi uporabnikovega izbora algoritma izvede strojno učenje.

Za učenje iz knjižnice WEKA uporabimo podrazrede razreda `Classifier`:

- Neural Network – nevronske mreže,
- J48 – verzija odločitvenih dreves C-45,
- Ibk – k-najbližjih sosedov.

Vsak izmed klasifikatorjev se ustvari za vsako delnico posebej, vsi skupaj pa so shranjeni v enem polju. Celotno analizo je potrebno izvesti za tolikšno število dni, kot ga je izbral uporabnik. Postopek analize je zato umeščen znotraj ene zanke, ki se izvede tolikokrat, kot znaša želeno število dni. Rezultat analize je potrebno pripraviti v obliki poročila. Le-to se sestavi tako, da je iz njega razvidno, kako so se zgrajeni učni modeli obnesli pri testiranju, saj se je na tej podlagi program odločil za trgovanje.

**Slika 29: Poročilo analize z uporabo odločitvenih dreves**




Vir: Programski paket Napovedovalec in lastni izračuni

Rezultat obdelave za posamezen dan tako zajema prikaz modela, ki je bil zgrajen na podlagi učnih podatkov za vsako delnico posebej. Tako je iz primera izpisa (slika 30) razvidno odločitveno drevo za delnico podjetja Merkur, d.d. (MER), kjer vidimo, da bo delnica jutri rangirana nadpovprečno dobro, če je bila prejšnji dan podpovprečna, in sicer pod ali pa enako vrednosti –0,071429.

**Slika 30: Izpis odločitvenega drevesa za delnico Merkurja (MER)**

```
=====================================================================
Tole je evaluacija z drevesom za delnico MER:

Correctly Classified Instances         283               72.5641 %
Incorrectly Classified Instances       107               27.4359 %
Kappa statistic                          0.4513
Mean absolute error                      0.3905
Root mean squared error                  0.4442
Relative absolute error                 78.1011 %
Root relative squared error             88.8154 %
Total Number of Instances              390

 Tole je graf za delnico MER:J48 pruned tree
------------------

MER1 <= -0.071429: podpovprecen (408.0/113.0)
MER1 > -0.071429
|   MER2 <= -0.071429
|   |   MER2 <= -0.214286
|   |   |   MER4 <= -0.214286: podpovprecen (4.0)
|   |   |   MER4 > -0.214286
|   |   |   |   MER4 <= -0.071429: nadpovprecen (3.0)
|   |   |   |   MER4 > -0.071429: podpovprecen (3.0)
|   |   MER2 > -0.214286
|   |   |   MER3 <= -0.214286: nadpovprecen (13.0/3.0)
|   |   |   MER3 > -0.214286
|   |   |   |   MER5 <= -0.214286: podpovprecen (13.0/4.0)
|   |   |   |   MER5 > -0.214286: nadpovprecen (61.0/23.0)
|   MER2 > -0.071429: nadpovprecen (275.0/63.0)

Number of Leaves  :     8

Size of the tree :      15

Delnica MER je klasificirana   podpovprecen

verjetnost za delnico MER za podpovprecen znasa 72,3%

verjetnost za delnico MER za nadpovprecen znasa 27,7%
=====================================================================
```

Vir: Programski paket Napovedovalec

Pri tem upoštevamo, da se rangirne vrednosti lahko gibljejo v razponu med +0,5 in -0,5. V primeru, da bo rangirna vrednost prejšnjega dne nad to vrednostjo in da bo rangirna vrednost predprejšnjega dne (MER2 – dva dni nazaj) večja od vrednosti –

0,071429 (spodnji list drevesa), bo prav tako kotirala nadpovprečno. Tako lahko po listih navzdol preberemo celotno drevo. Tako zgrajeno odločitveno drevo Napovedovalec preizkusi na testnih podatkih rangiranih vrednosti za vsako delnico posebej. Rezultati preizkusa modela (drevesa) so prikazani nad njim. Iz njih lahko za delnico Merkurja razberemo, da je to drevo pravilno klasificiralo podatke z 72,5-odstotno verjetnostjo, kar je razvidno iz podatka `Correctly Classified Instances`. Pravilno je klasificiranih 283 testnih primerov, medtem ko je nepravilno klasificiranih 107, kar predstavlja preostalih 27,4 odstotka od 390 testnih primerov (Instances). Poleg teh najpomembnejših testnih podatkov so prikazane še druge opisne statistične vrednosti.

Najpomembnejši so podatki o verjetnosti za klasifikacijo konkretnega »današnjega« podatkovnega primera, torej zadnjega primera v časovni vrsti, kjer poskušamo predvideti jutrišnjo najbolj obetavno delnico izmed izbranih. Pod drevesom na dnu izseka izpisa lahko vidimo, da je verjetnost za delnico MER za podpovprečen 72,3 odstotka in za nadpovprečen preostalih 27,7 odstotka. To pomeni, da lahko za jutrišnji dan na podlagi odločitvenega drevesa ocenimo, da bo delnica Merkurja kotirala podpovprečno in to lahko trdimo s 72,5-odstotno verjetnostjo, kolikor znaša pogostost pravilnega napovedovanja drevesa. Na osnovi teh podatkov se ni modro odločati za nakup te delnice, saj bo najverjetneje med izbranimi delnicami slabo, torej podpovprečno kotirala. Napovedovalec je torej moral izbrati med drugimi delnicami, katerih rezultate analize si lahko bralec ogleda v celotnem izpisu v prilogi A.

**Slika 31: Povzetek enodnevnega preizkusa odločitvenih dreves**

```
============================================================

                        POVZETEK ANALIZE

============================================================


DELNICA    INDEX    NAD          POD    ZANESLJIVOST VELIKOST DREVESA    PONDER
AELG         0      0,28         0,72         0            0.0          0
ETOG         1      0,76         0,24         68,97        17.0         22,85
GRVG         2      0,79         0,21         69,74        21.0         23,5
IEKG         3      0,13         0,87         0            0.0          0
KRKG         4      0,32         0,68         0            0.0          0
LKPG         5      0,71         0,29         72,82        7.0          22,97
MER          6      0,28         0,72         0            0.0          0
TCRG         7      0,72         0,28         69,74        3.0          21,65

Potencialno najbolj donosna delnica je GRVG
Donosnost za dan 1 za delnico GRVG bi znasala: 0,90339%

Skupna donosnost za obdobje 1 dni bi znasala:  0,90339%
```

Vir: Programski paket Napovedovalec

V celotnem izpisu, ki ga izdela program Napovedovalec, so prikazani isti parametri za vse obravnavane delnice. Poleg tega imamo na vpogled tudi povzetek obdelave, ki si ga lahko ogledamo tudi na sliki 31. V tem povzetku so prikazane verjetnosti za spremenljivki nadpovprečen in podpovprečen za vsako delnico posebej za obravnavani dan, saj se obdelava izvaja po dnevih. Poleg verjetnosti, da bo delnica kotirala nadpovprečno, so na razpolago tudi vrednosti o zanesljivosti teh porazdelitev. To so odstotkovne vrednosti pravilno napovedanih testnih podatkov za vsako odločitveno drevo, torej za vsako delnico posebej. Podatki so veljavni samo za tiste delnice, katerih porazdelitev verjetnosti za vrednost nadpovprečen znaša več kot nič, saj nas tiste z večjo podpovprečno verjetnostjo ne zanimajo. Prav tako imamo na razpolago tudi podatke o velikosti drevesa, saj bi bilo zanimivo videti, kako velikost odločitvenega drevesa vpliva na napovedovanje. Ponderirana vsota vseh treh kazalcev je prikazana v zadnjem stolpcu povzetka. Izbrani ponderji v tem primeru so bili 0,6 za zanesljivost zadetka, 0,3 za porazdelitev verjetnosti za nadpovprečno vrednost ter 0,1 za velikost drevesa, skupaj torej 1.

Iz izpisa povzetka torej vidimo, da je največjo ponderirano vsoto dosegla delnica GRVG, in sicer 23,9. Če bi kupili tako izbrano delnico, bi v enem dnevu dosegli 0,9 odstotno donosnost. Program je zastavljen tako, da zaradi obsežnosti izpisa pri večjem številu dni (več kot 5) izpisuje samo rezultate oz. donosnosti, ki bi bile dosežene, če bi se trgovalo na podlagi tako dobljenega izbora (glej prilogo A). Pri tem program ne zahaja v podrobnosti na tem poročilu.

# 5 REZULTATI NAPOVEDOVANJA GIBANJA DONOSNOSTI DELNIC

Napovedovanje obnašanja nekega pojava na podlagi preteklih podatkov je nemogoče, če imamo opravka s časovno vrsto podatkov, ki se obnaša po načelu naključnega hoda. V takem primeru vsakršen algoritem za predvidevanje predznaka spremembe v časovni vrsti na daljši rok doseže 50-odstotno stopnjo zadetka. V primeru napovedovanja delnic, ko imamo opravka s časovnimi vrstami, ki se spreminjajo »skoraj« povsem naključno, **so stopnje natančnosti predvidevanja predznaka spremembe v višini 54 % označene kot uspešne** (Hellstrom, 2001, str. 8). Po izjavah razvijalcev novih algoritmov in metod za napovedovanje delnic ni težko razviti algoritme, s katerim bi opravili enkratno ali dvakratno uspešno napoved v daljši časovni vrsti, pač pa je težko narediti veliko število napovedi, od katerih bi bila večina pravilnih, torej takih s pozitivnim predznakom spremembe. Rečemo lahko, da smo skonstruirali uspešen algoritem, šele ko ob velikem številu napovedi ustvarimo več kot petdeset odstotkov pozitivnih sprememb v časovni vrsti oz. več kot 54 %, ki nekako predstavlja mejo, katere večina novo razvitih algoritmov za napovedovanje delnic ne presega.

Glede na to, da je naloga, ki smo si jo zastavili, ustvariti algoritem, ki med danimi delnicami izbere delnico donosnejšo od povprečja danih delnic, je stopnja doseženih pozitivnih sprememb v daljšem časovnem obdobju smiseln kriterij za oceno uspešnosti algoritma, saj na tej podlagi lahko ocenimo, kako pogosto je algoritem izbiral donosne delnice. Doseganje skupnega donosa v nekem daljšem roku ni najbolj smiselna, saj algoritem z rangiranjem delnic poskuša izkoriščati tržne anomalije v obliki učinka preobratov, ki pa je kratkoročni učinek. Skupni donos bi bilo možno ugotavljati s simulacijo trgovanja, kjer bi uvedli dodatna trgovalna pravila s katerimi bi omejili število vstopov in izstopov, saj vsakodnevno trgovanje povzroča ogromne transakcijske stroške, ki izničijo vsak normalno ustvarjen donos. Poleg tega je treba upoštevati, da algoritem za napovedovanje nadpovprečno uspešne delnice, ki napove, katera delnica bo v naslednjem časovnem koraku izmed izbranih donosnejša od povprečja v času splošnega padca cen delnic, rezultira v izbiri prav tako padajoče delnice, predvidoma seveda tiste, ki bo padla manj od povprečja izbranih delnic. Ker pa nas zanima predvsem učinkovitost algoritma, katerega naloga je izbrati donosno delnico, pa stopnja pravilno napovedanega predznaka spremembe v časovni vrsti tudi na daljši rok pokaže njegovo uspešnost.

Na krajši rok lahko poleg stopnje pravilno napovedanih pozitivnih donosov ocenimo uspešnost algoritma tudi z dobičkom doseženim v tem roku. To lahko primerjamo z določenim drugim kriterijem, kot je npr. dobiček dosežen s strategijo »kupi in zadrži«. Na podlagi take primerjave lahko ocenimo vse tri metode: nevronske mreže, odločitvena drevesa in metodo najbližjega soseda.

## 5.1 Določitev obdobja testiranja in izbira vrednostnih papirjev

Pri izbiri obdobja podatkov o trgovanju smo se odločili pogledati, kako se metoda napovedovanja obnaša v času naraščanja in kako v času padanja cen delnic na borzi. Tako sta bili izbrani dve obdobji in sicer od 30. 11. 1998 do 24. 09. 2003 ter od 12.1. 2001 do 11. 07.2005. Za konec prvega obdobja, ko je program Napovedovalec testiral zgrajene modele, je značilna rahla rast tečajev. Za konec drugega obdobja pa je značilen padec tečajev delnic. Program za napovedovanje v skladu z opredelitvami iz četrtega poglavja zgradi model na podlagi vzorcev iz začetka obdobja in simulira trgovanje za zadnje število izbranih dni. Za dolgo obdobje v katerem smo ocenjevali rezultate simuliranega trgovanja so bili uporabljeni podatki zadnjega pol leta v izbranem obdobju, medtem ko je za kratko obdobje izbrana dolžina zadnjih dveh mesecev.

Pri sami izbiri delnic sta bila postavljena dva kriterija, in sicer likvidnost delnice, torej pogostost trgovanja, in dovolj dolgo obdobje pogostega trgovanja z delnico na borzi, ki naj bi znašalo vsaj nekaj let. Podatke o tečajnih vrednostih smo povzeli iz arhiva

na spletni strani poslovnega dnevnika Finance, nanašajo pa se na delnice sledečih delniških družb:

1. obdobje:
AELG: delnica delniške družbe Aerodrom Ljubljana, d.d.
ETOG: delnica delniške družbe Etol, d.d.
GRVG: delnica delniške družbe Gorenje, d.d.
IEKG: delnica delniške družbe Intereuropa, d.d.
KRKG: delnica delniške družbe Krka, d.d.
LKPG: delnica delniške družbe Luka Koper, d.d.
MER: delnica delniške družbe Merkur, d.d.
TCRG: delnica delniške družbe Terme Čatež, d.d.

2. obdobje:
KRKG: delnica delniške družbe Krka, d.d.
MELR: delnica delniške družbe Mercator, d.d.
MER: delnica delniške družbe Merkur, d.d.
GRVG: delnica delniške družbe Gorenje, d.d.
MAJG: delnica delniške družbe Mlinotest, d.d.
SALR: delnica delniške družbe Salus, d.d.
TCRG: delnica delniške družbe Terme Čatež, d.d.
SAVA: delnica delniške družbe Sava, d.d.
ZTOG: delnica delniške družbe Žito, d.d.

Za izbrane delnice je značilno, da se je v izbranem v izbranem obdobju z njimi najpogosteje trgovalo. Obstajajo tudi delnice družb s katerimi se je na Ljubljanski borzi pogosteje trgovalo, kot je Pivovarne Union, d.d. ali pa Lek, d.d., Vendar pa obdobje, v katerem so te delnice kotirale na borzi ni zadovoljivo dolgo. Po drugi strani pa delnice, ki na borzi kotirajo dovolj dolgo, niso dovolj likvidne. Zgoraj navedene delnice pa nekako izpolnjujejo oba kriterija.

## 5.2 Testiranje algoritma na daljše obdobje

Ob testiranju algoritma na podatkih iz prej opisanega prvega izbranega obdobja za obdobje šestih mesecev (150 trgovalnih dni), ko naj bi z algoritmom hipotetično dnevno kupovali in prodajali, so bili doseženi sledeči rezultati (Tabela 3):

**Tabela 3: Stopnja pozitivnih donosnosti, doseženih v 6 mesecih z različnimi metodami strojnega učenja.**

| Uporabljena metoda strojnega učenja | Število pozitivnih donosov | Število negativnih donosov | Stopnja pozitivnih donosov (Hit rate) v % |
|---|---|---|---|
| Nevronske mreže | 72 | 51 | 58 |
| Odločitvena drevesa | 60 | 53 | 53 |
| K-najbližjih sosedov | 84 | 62 | 57 |

Vir: Lastni izračuni

Na tej osnovi ugotavljamo, da je algoritem dosegel najboljši rezultat z uporabo nevronskih mrež s stopnjo pozitivnega donosa v višini 58 %, kar lahko v primerjavi z drugimi študijami štejemo kot zelo uspešno. Iz izpisa programa Napovedovalec, ki implementira algoritem (glej prilogo E), vidimo, da s pomočjo nevronskih mrež algoritem velikokrat spreminja odločitev glede izbrane delnice, v nasprotju s preostalima metodama. Iz tega sledi, da so nevronske mreže bistveno bolj odzivne na spremembe, kar lahko štejemo za zelo uporabno pri trgovanju na kratki rok, za kar je algoritem tudi namenjen.

## 5.3 Testiranje algoritma na krajše obdobje

### 5.3.1 Strategija »kupi in zadrži

Da bi lahko ocenili donosnost algoritma za napovedovanje delnic, je prej potrebno ugotoviti donosnost, ki bi jo dosegli s pomočjo strategije "kupi in zadrži" v krajšem obdobju, recimo dveh mesecev (40 trgovalnih dni). Izračunati je potrebno torej donosnost nakupa izbranih delnic in prodaje po zadnjih 40 dneh vsakega izmed obeh obdobij. Ta strategija bi nam omogočila sledeče donose prikazane v tabeli št. 4. Kot vidimo, je za konec prvega obdobja značilna rast tečajev, za konec drugega obdobja pa padec tečajev. Z rezultati strategije "kupi in zadrži" lahko za isto obdobje dveh mesecev primerjamo rezultate hipotetičnega trgovanja, ki bi ga izvedel Napovedovalec na podlagi rangiranja delnic.

**Tabela 4: Donosnost posamezne delnice pri strategiji »kupi in zadrži« po obdobjih**

| 1. Obdobje | | | | 2. Obdobje | | | |
|---|---|---|---|---|---|---|---|
| **Delnica** | **Začetni tečaj** | **Končni tečaj** | **Donosnost v %** | **Delnica** | **Začetni tečaj** | **Končni tečaj** | **Donosnost v %** |
| AELG | 4.789,02 | 6.064,62 | 26,64 | KRGK | 77.989,98 | 77.358,22 | -0,81 |
| ETOG | 38.600,00 | 42.500,00 | 10,10 | MELR | 37.702,44 | 34.656,40 | -8,08 |
| GRVG | 4.083,53 | 4.741,44 | 16,11 | MER | 39.759,28 | 36.065,29 | -9,29 |
| IEKG | 5.250,00 | 5.455,40 | 3,91 | GRVG | 5.909,43 | 5.343,91 | -9,57 |
| KRKG | 40.787,93 | 47.950,28 | 17,56 | MAJG | 1.700,00 | 1.500,00 | -11,76 |
| LKPG | 4.547,23 | 5.719,18 | 24,77 | SALR | 129.000,00 | 125.003,00 | -3,10 |
| MER | 20.864,16 | 24.936,66 | 19,52 | TCRG | 38.500,00 | 34.500,16 | -10,39 |
| TCRG | 26.890,00 | 27.500,04 | 2,27 | SAVA | 44.623,74 | 43.699,80 | -2,07 |
| | | | | ZTOG | 30.350,00 | 29.951,32 | -1,31 |

Vir: Lastni izračuni

V sklopu ocenjevanja uspešnosti smo se odločili oceniti tudi, kako posamezni faktorji vplivajo na posamezno metodo strojnega učenja. Zanimiva je torej porazdelitev verjetnosti med tem, ali bo posamezna delnica na določen dan v prihodnosti kotirala nadpovprečno ali pa podpovprečno v primerjavi z ostalimi izbranimi delnicami. Prav

tako bi lahko bilo zanimivo videti, kako se je zgrajeni model izkazal pri napovedovanju (klasifikaciji) na testni množici podatkov. Tretji potencialno zanimiv faktor, ki bi ga lahko upoštevali, vendar samo v primeru metode odločitvenih dreves, pa je vpliv same velikosti odločitvenega drevesa. Navedene kriterije lahko v programu Napovedovalec aktiviramo v obliki ponderjev. Skupna vrednost ponderjev, kot je to v poglavju 4 že navedeno, mora znašati 1.

### 5.3.2 Odločitvena drevesa

Rezultati napovedovanja nadpovprečne delnice za en dan vnaprej z uporabo metode odločitvenih dreves, ki bi jih algoritem dosegel v času dveh mesecev, so prikazani v tabeli 5. Uporabljene so bile različne razdelitve ponderjev po kriterijih za odločanje, in sicer glede na kriterij porazdelitve verjetnosti med tem, ali bo delnica naslednjega dne kotirala nadpovprečno ali ne, potem kriterij klasifikacijske učinkovitosti odločitvenega drevesa na testni množici podatkov ter kriterij velikosti zgrajenega odločitvenega drevesa.

**Tabela 5: Kriteriji uspešnosti algoritma z uporabo odločitvenih dreves**

| Ponderji | | | Število negativnih donosnosti | Število pozitivnih donosnosti | Delež pozitivnih donosnosti v % | Donosnost v % |
|---|---|---|---|---|---|---|
| Porazdelitev verjetnosti med spremenljivkama | Učinkovitost modela | Velikost drevesa | | | | |
| 1,0 | 0,0 | 0,0 | 8 | 22 | 73 | 16,47 |
| 0,9 | 0,1 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,8 | 0,2 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,7 | 0,3 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,6 | 0,4 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,5 | 0,5 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,8 | 0,1 | 0,1 | 11 | 23 | 67 | 12,83 |
| 0,7 | 0,2 | 0,1 | 11 | 20 | 64 | 13,41 |
| 0,6 | 0,3 | 0,1 | 12 | 22 | 64 | 11,47 |
| 0,7 | 0,1 | 0,2 | 11 | 23 | 67 | 12,83 |
| 0,6 | 0,2 | 0,2 | 11 | 22 | 66,6 | 12,83 |

Vir: Lastni izračuni

Iz rezultatov lahko razberemo, da algoritem veliko bolje napoveduje, če pri izbiri nadpovprečne delnice upošteva tudi učinkovitost modela, ki ga je dosegel pri testiranju. Torej se algoritem bolje obnese, če kriteriju porazdelitve verjetnosti med spremenljivkama NAD in POD za konkretno enodnevno napoved ne dodelimo celotnega ponderja (1,0), pač pa pri odločanju za nakup del ponderja dodelimo tudi kriteriju učinkovitosti modela. V tem primeru je algoritem dosegal 78 % pozitivih donosov, od katerih je bila tudi dosežena skupna donosnost boljša, kot če ne upoštevamo pretekle uspešnosti zgrajenega modela. Sama velikost zgrajenega drevesa pa ni pomembna, zato je temu kriteriju bolje dodeliti ponder 0,0. V

nasprotnem primeru rezultati tako po stopnji doseženih pozitivnih donosov kot tudi po skupni vrednosti ustvarjenega donosa padejo. V najboljšem primeru z uporabo odločitvenih dreves algoritem za napovedovanje doseže večjo donosnost kot s strategijo »kupi in zadrži« pri večini delnic iz tabele 4. V primeru podatkov iz drugega obdobja, torej v času splošnega padanja cen delnic na borzi, pa je program Napovedovalec s pomočjo algoritmov za gradnjo odločitvenih dreves dosegel donosnost -0,79 %, torej bolje, kot najboljša delnica po strategiji »kupi in zadrži«.

### 5.3.3 Metoda k-najbližjih sosedov

V primeru uporabe metode k-najbližjih sosedov vidimo, da se je najboljše zanesti na klasifikacijo novega primera, ne da bi se ozirali na uspešnost metode v preteklosti, saj v primeru podelitve celotnega ponderja (1,0) porazdelitvi verjetnosti med spremenljivkama, dosežemo najboljšo stopnjo pozitivnih donosov, kakor tudi najvišjo donosnost (glej tabelo 6).

**Tabela 6: Kriteriji uspešnosti algoritma z uporabo k-sosedov**

| Ponderji | | Število negativnih donosnosti | Število pozitivnih donosnosti | Delež pozitivnih donosnosti v % | Donosnosti v % |
|---|---|---|---|---|---|
| Porazdelitev verjetnosti med spremenljivkama | Učinkovitost modela | | | | |
| 1,0 | 0,0 | 11 | 39 | 78 | 23,02 |
| 0,9 | 0,1 | 10 | 9 | 47 | 9,22 |
| 0,8 | 0,2 | 15 | 15 | 50 | 5,75 |
| 0,7 | 0,3 | 14 | 16 | 53 | 14,14 |
| 0,6 | 0,3 | 14 | 16 | 53 | 14,14 |

Vir: Lastni izračuni

V primeru podatkov iz drugega obdobja, torej v času splošnega padanja cen delnic na borzi, pa je program Napovedovalec s pomočjo metode k-najbližjih sosedov prav tako dosegel donosnost -0,79 %, torej tudi bolje, kot najboljša delnica po pasivni strategiji »kupi in zadrži«.

### 5.3.4 Nevronske mreže

Metode nevronskih mrež so se poleg odločitvenih dreves prav tako dobro izkazale. Dosežena je bila tudi nekaj večja maksimalna donosnost v višini 24,93 %. Nevronske mreže so tako kot pri šestmesečnem testu tudi tukaj pokazale veliko večjo odzivnost in so večkrat spreminjale odločitev za nakup določene izmed izbranih delnic. Pri odločitvenih drevesih se sproti z vsakim novim trgovalnim dnem sicer spremenijo vrednosti v listih, ne pa tudi sama struktura drevesa. To vodi v odločitev na podlagi zelo podobnega drevesa (ohranitev spomina) in s tem ne privede do odločitve za drugo delnico. Pri nevronskih mrežah pa sprememba vrednosti uteži bolj vpliva na

spremembo odločitve, katera delnica ima v naslednjem obdobju najvišjo verjetnost, da bo dosegla nadpovprečno donosnost, kar je na kratek rok pomembno.

**Tabela 7: Kriteriji uspešnosti algoritma z uporabo nevronskih mrež**

| Ponderji | | Število negativnih donosnosti | Število pozitivnih donosnosti | Delež pozitivnih donosnosti v % | Donosnost v % |
|---|---|---|---|---|---|
| Porazdelitev verjetnosti med spremenljivkama | Učinkovitost modela | | | | |
| 1,0 | 0,0 | 15 | 30 | 67 | 12 |
| 0,9 | 0,1 | 12 | 30 | 71 | 23,68 |
| 0,8 | 0,2 | 11 | 32 | 74,4 | 24,93 |
| 0,7 | 0,3 | 10 | 32 | 76,2 | 22,15 |
| 0,6 | 0,4 | 11 | 32 | 74,4 | 22,16 |

Vir: Lastni izračuni

## 5.4 Skupna ocena rezultatov

Na podlagi rezultatov napovedovanja, podanih v tem poglavju, predvsem dolgoročnega testiranja, lahko sklepamo, da je s pomočjo rangiranja delnic in uporabo strojnega učenja, dejansko možno napovedati katere delnice bodo v prihodnosti donosne. Rezultat stopnje natančnosti predvidevanja predznaka spremembe v višini 58%, dosežen v daljšem roku, nam namreč pove, da nismo samo ugibali, pač pa smo uspeli dejansko predvideti »precej« več kot petdeset odstotkov pozitivnih premikov v časovni vrsti. Ta rezultat je treba upoštevati ob dejstvu, da gre za zelo učinkovit trg, torej so premiki v časovnih vrstah gibanja cen delnic zelo, zelo blizu petdeset odstotni porazdelitvi med padci in dvigi cen, kjer lahko rezultate v stopnji natančnosti predvidevanja predznaka spremembe nad 54%, ob dokaj velikem številu nakupov, štejemo za uspešne.

Pri preizkusu testiranih metod strojnega učenja pri napovedovanju ranga delnice lahko ugotovimo, da se najbolje obnesejo nevronske mreže, saj jim uspe doseči najvišji končni donos po določenem številu nakupov. To pomeni, da izbirajo donosnejše delnice v primerjavi z ostalimi metodami.

# 6 SKLEP

Napovedovanje dogajanja na kapitalskem trgu je zaradi njegove visoke učinkovitosti precej zahtevna naloga, lahko bi rekli celo nemogoča, saj je gibanje tečajev delnic, po trditvah nekaterih ekonomistov, povsem naključno. Pa vendar obstajajo empirični dokazi, ki temu nasprotujejo, saj kažejo na določene tržne anomalije, katere je možno zaznati kot ponavljajoče se vzorce cenovnih gibanj vrednostnih papirjev. Te

anomalije bi spretni vlagatelj lahko izkoristil ter tako dosegel nadpovprečni donos v primerjavi z ostalimi udeleženci na kapitalskem trgu.

Od pojava računalnikov naprej, s katerimi lažje obdelamo velike količine podatkov, lahko vlagatelj namesto svojega »občutka« uporablja stroj, s katerim lahko natančneje analizira borzne podatke in s tem lažje doseže prednost pred drugimi vlagatelji, kar je lahko nagrajeno z večjim dobičkom. V tej tekmi se vedno pojavljajo nove računske metode, s katerimi lahko dobimo »globlji« vpogled v borzne podatke. Vedno bolj se uveljavljajo tudi metode strojnega učenja, ki izhajajo iz področja umetne inteligence. S svojo sposobnostjo modeliranja podatkov so te metode že številna leta v uporabi na področju medicine, meteorologije, robotike, itd. Z njimi si lahko pomagamo natančneje povzeti določene vzorce tudi v borznih podatkih, kot so vzorci tržnih anomalij, na podlagi česar lahko natančneje napovemo dogajanje na kapitalskem trgu.

V tem magistrskem delu je bila preizkušena metoda napovedovanja, s pomočjo katere smo poizkušali izkoristiti predvsem dve tržni anomaliji: učinek preobratov, ki se kaže v pretirani impulzivnosti pri odzivanju trga delnic na nove informacije, ter učinek medsebojne cenovne odvisnosti med delnicami. Metoda deluje tako, da se določeno število delnic za čas nekega obdobja vsak dan posebej razvrsti. Razvrstitev (rangiranje) se opravi glede na donosnost delnic v preteklosti. Za tem se iz teh časovnih vrst rangov za vsako izmed delnic zgradi model s pomočjo metod strojnega učenja. Na podlagi testiranja modelov ugotovimo, katera izmed delnic bo glede na ostale izbrane delnice dosegla nadpovprečno donosnost v prihodnjem obdobju. Kot metode strojnega učenja so bile uporabljene nevronske mreže, metoda k-najbližjih sosedov in odločitvena drevesa. Opisan postopek smo implementirali v obliki računalniškega programa, imenovanega »Napovedovalec«.

Uspešnost neke metode, s katero napovedujemo gibanje v prihodnosti lahko ocenjujemo po različnih kriterijih. Eden izmed kriterijev je tudi stopnja pravilno napovedanega predznaka spremembe v časovni vrsti, ki bi jih dosegli z neko metodo. Ta stopnja nam lahko pove, kakšna je verjetnost, da na podlagi določene metode dosežemo pozitivno donosnost. Pri tem je treba vedeti, da ni težko doseči pozitivni donos z manjšim številom nakupov in prodaj, pač pa je izziv izvesti veliko število trgovanj in pri tem ustvariti veliko število pozitivnih donosov. Zato smo v okviru testiranja izvedli trgovanje za vsak dan v časovni vrsti. Če tako uporabljeno metodo rangiranja delnic ocenjujemo s stopnjo pravilno napovedanih pozitivnih sprememb v časovni vrsti, vidimo da na daljši rok (šest mesecev) doseže zelo dober rezultat, če ga primerjamo z rezultati drugih raziskovalcev. Razlog za to verjetno leži tudi v slabši učinkovitosti slovenskega kapitalskega trga, iz katerega smo črpali podatke, primerljive raziskave pa so bile opravljene na bolj razvitih zahodnih trgih.

Ker je učinek preobratov, ki ga poskušamo s to metodo izkoristiti, kratkoročne narave, je tudi ta metoda učinkovitejša na kratek rok. Po rezultatih na krajši rok smo metodo ocenjeval po dveh kriterijih, in sicer prav tako s stopnjo doseženih pozitivnih donosov kot tudi z donosnostjo, doseženo v tem obdobju. Po obeh kriterijih sta se zelo dobro izkazali tako metoda nevronskih mrež kot tudi metoda odločitvenih dreves. Prednost bi dali metodi nevronskih mrež, ki je med poskusom pokazala bistveno večjo odzivnost pri odločitvah glede potencialno uspešne delnice, saj se je odločitev večkrat spreminjala, kar menim, da je na kratki rok pomembno.

Tako kot ostale analitične metode, ki jih poznamo s področja tehnične analize ali pa analize časovnih vrst, tudi ta metoda ni idealna in ne more prinesti čarobne obogatitve uporabniku. Ena od pomanjkljivosti je izbira padajoče delnice, saj se z izborom potencialno nadpovprečne delnice izmed izbranih delnic ne moremo izogniti negativnim donosnostim, kadar cene vseh izmed izbranih delnic padejo. Metoda poskuša namreč izmed danih delnic izbrati tisto, ki naj bi bila nadpovprečno donosna. Tako je v primeru, ko vse dane delnice padejo, tudi donosnost izbrane »nadpovprečne« delnica negativna kljub temu, da je izguba morda manjša od povprečne izgube danih delnic. Kljub temu, da padec cen vseh izbranih delnic na isti dan verjetno ni zelo pogost pojav, je omenjeno težavo možno rešiti z uvedbo dodatnih trgovalnih pravil, ki bi omejila takratne nakupe.

Na splošno pa lahko lahko ocenimo, da je analiza delnic s pomočjo metod strojnega učenja in ustrezne priprave podatkov v obliki rangiranja delnic lahko uporabna pri trgovanju, predvsem zaradi svoje natančnosti modeliranja tržnih anomalij.

# 7 LITERATURA IN VIRI

## LITERATURA


1. Achelis B. Steven: Technical analysis from A to Z. [URL: http://www.equis.com/Education/TAAZ/ ], 10. 10. 2003.

2. Armstrong Scott: Principles of forecasting. Boston: Kluwer Academic, 2001. 864 str.

3. Bodie Zvi, Kane Alex, Marcus Alan J.: Investments, International Editions: Irwin/McGraw-Hill, 1997, 997 str.

4. Breiman L. et al.: Classification and regression trees. Wadsforth International Group, 1986.

5. Brigham Eugene F.: Fundamentals of financial management, 7th Edition. Orlando (FL): The Dryden Press, Harcourt Brace College Publishers, 1995. 843 str.

6. Damij Talib: Tehnike razvoja algoritmov. Ljubljana: Univerza v Ljubljani, Ekonomska Fakulteta, 1994, 21 str.

7. Deželan Silva: Učinkovitost trga kapitala. Magistrsko delo. Ljubljana: Univerza v Ljubljani, Ekonomska Fakulteta, 1996. 104 str.

8. Edwards Robert D., Magee John: Technical analysis of stock trends. Seventh edition. New York: St. Lucie Press, 1998. 721 str.

9. Fayyad U.M.: On the induction of decision trees for multiple concept learning. Doktorsko delo, Univerza v Michiganu, 1991.

10. Gams Matjaž: Principi poenostavljanja v sistemih za avtomatsko učenje. Doktorsko delo. Ljubljana: Univerza v Ljubljani, Fakulteta za elektrotehniko in računalništvo, 1988. 135 str.

11. Gams Matjaž: Intelligence: Through the principle and paradox of multiple knowledge. (Advances in computation, vol. 6). Huntington: Nova Science, 2001. 245 str.

12. Good Irving John: Probability and the weighing of evidence. London: Charles Griffin, 1950.

13. Goonatilake Suran, Treleaven Philip: Intelligent systems for finance and business. Chicheter: John Wiley & Sons, 1995. 335 str.

14. Hellstrom Thomas: A Random walk through the stock market. Umea (Švedska): Umea University, 1998, 129 str.

15. Hellstrom Thomas: Optimizing the sharpe ratio for a rank based trading system. [URL: http://www.cs.umu.se/~thomas ], 10. 10. 2000.

16. Jagrič Timotej: Uporaba nevronskih mrež pri napovedovanju ekonomske aktivnosti z vodilnimi indikatorji. Doktorska disertacija. Maribor: T. Jagrič, 2000.

17. Jean Bojan: Uporaba in prilagoditev metod temeljne analize delnic v slovenskih razmerah. Magistrsko delo. Ljubljana: Ekonomska fakulteta, 2000.

18. Kira K. in Rendell L.: A practical aproach to feature selection, Proc. Int. conf. On machine learning. Aberdeen: Morgan Kaufmann, 1992. str. 249–256.

19. Kononenko Igor: Strojno učenje. Ljubljana: Založba FE in FRI, 1997. 303 str.

20. Košmelj Blaženka, Rovan Jože: Statistično sklepanje. Ljubljana: Ekonomska fakulteta, 1997. 312 str.

21. Lozano-Perez T. in Kaelbling L.: Artificial intelligence, Zapiski predavanj, Massachusetts Institute of Technology, 2003.

22. Mantaras R.L.: ID3 revisited: A distance based criterion for attribute selection, Proc. Int. Symp. Methodologies for Intelligent Systems. Charlotte, North Carolina, 1989.

23. Morton Rob: Training the multi-layer perceptron [URL:] http://home.clara. net/robmorton/projects/neural, 10. 10. 2003.

24. Mramor Dušan: Uvod v poslovne finance. Ljubljana: Gospodarski vestnik, 1993. 381 str.

25. Mramor Dušan: Poglavja iz poslovnih financ (zapiski predavanj). Ljubljana: Ekonomska fakulteta, 1997. 125 str.

26. Mramor Dušan et al.: Trg kapitala v Sloveniji. Ljubljana: Gospodarski vestnik, 2000. 471 str.

27. Michalski Ryszard, Bratko Ivan, Kubat M.: Machine learning and data mining. Chichester: John Wiley & Sons, 1998. 456 str.

28. Mitchell Tom: Machine learning. New York: McGraw Hill, 1997. 414 str.

29. Murphy John: Technical analysis of the financial markets. New York: New York Institute of Finance, 1999. 542 str.

30. Press W.H. et al.: Numerical recepies in C: The art of Scientific programing. Cambridge University Press, 1988.

31. Quinlan Ross: Induction of decision trees, Machine learning, št. 1, 1986. str. 81–106.

32. Quinlan Ross: A Case study in machine learning. [ URL: http://www. cse.unsw .edu.au/~quinlan/ ], 10. 10. 2003.

33. Quinlan Ross: Learning with continuous classes. [ URL: http://www .cse.unsw. edu.au/~quinlan/ ], 10. 10. 2003.

34. Quinlan Ross: Bagging, Boosting, and C4.5 [URL: http://www .cse.unsw.edu. au/~quinlan/ ], 10. 10. 2003.

35. Rees Bill: Financial analysis. London: Prentice Hall, 1995. 390 str.

36. Ribnikar Ivan: Monetarna ekonomija I. Ljubljana: Ekonomska fakulteta, 1999. 380 str.

37. Rissanen J.: A universal prior for integers and estimation by minimum description length. The Anndls of Statistics, 11(2). 1993. str. 416–431.

38. Smyth P. in Goodman R.M. (1990): Rule induction using information theory. Knowledge Discovery in Databases, MIT Press, 1990.

39. Šmigič Dragan: Uporaba metod strojnega učenja pri analizi vrednostnih papirjev. Zbornik A 6. mednarodne multikonference Informacijska družba 2003, Ljubljana, str. 139–142.

40. Trančar Vesna: Tehnična analiza delnic in njena uporaba v praksi. Naše gospodarstvo, Ljubljana, 64 (2000), 5/6, str. 742–755.

41. Yule Udny G.: On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. Philosophical transactions of the royal society of london, London,1927. A,226, Harrison and Sons, London, 1927. str 267–297.

42. Witten H. Ian, Eibe Frank: Data mining. San Francisco: Morgan Kaufman Publishers, 2000. 337 str.

**VIRI**

1. Časnik Finance. [URL: http://www. Finance-on.net/arhivtec.php],10. 10. 2003.

2. JfreeChart. [URL: http://www.jfree.org/jfreechart/index.html], Programska knjižnica JfreeChart, 10. 10. 2003.

3. Weka. [URL: http://www.cs.waikatoac.nz/ml/weka/], Programski paket Weka 3.3.6, 10. 10. 2003.

# PRILOGA

Priloga A: Izpis programa Napovedovalec
Priloga B: Izpis hipotetičnega trgovanja za obdobje 50-ih dni z uporabo odločitvenih dreves
Priloga C: Izpis hipotetičnega trgovanja za obdobje 50-ih dni z metodo k - najbližjih sosedov
Priloga D: Izpis hipotetičnega trgovanja za obdobje 50-ih dni z nevronskimi mrežami
Priloga E: Izpis hipotetičnega trgovanja za obdobje 150-ih dni z metodo nevronskih mrež
Priloga F: Programska koda računalniškega programa Napovedovalec
Priloga G: Podatki o kotacijah delnic urejeni v formatu ARFF

**Priloga A**

Izpis programa Napovedovalec za en dan v naprej. Izpisani so učinki potencialne uspešnosti napovedovanja posamezne delnice s posameznim odločitvenim drevesom in povzetkom analize na koncu.

```
=====================================================================
Tole je evaluacija z drevesom za delnico AELG:

Correctly Classified Instances         253               64.8718 %
Incorrectly Classified Instances       137               35.1282 %
Kappa statistic                          0.2931
Mean absolute error                      0.4298
Root mean squared error                  0.4838
Relative absolute error                 86.1454 %
Root relative squared error             96.8414 %
Total Number of Instances              390

 Tole je graf za delnico AELG:J48 pruned tree
------------------

AELG1 <= -0.071429
|   AELG3 <= 0.357143: podpovprecen (363.0/102.0)
|   AELG3 > 0.357143: nadpovprecen (19.0/8.0)
AELG1 > -0.071429: nadpovprecen (398.0/101.0)

Number of Leaves  :     3

Size of the tree :      5

Delnica AELG je klasificirana   podpovprecen

porazdelitev verjetnosti za delnico AELG za podpovprecen znasa 71,9%

porazdelitev verjetnosti za delnico AELG za nadpovprecen znasa 28,1%
=====================================================================
Tole je evaluacija z drevesom za delnico ETOG:

Correctly Classified Instances         269               68.9744 %
Incorrectly Classified Instances       121               31.0256 %
Kappa statistic                          0.3782
Mean absolute error                      0.4014
Root mean squared error                  0.463
Relative absolute error                 80.3319 %
Root relative squared error             92.635  %
Total Number of Instances              390

 Tole je graf za delnico ETOG:J48 pruned tree
------------------

ETOG1 <= 0.071429
|   ETOG1 <= -0.071429
|   |   ETOG4 <= -0.071429: podpovprecen (250.0/57.0)
|   |   ETOG4 > -0.071429
|   |   |   ETOG2 <= -0.357143: nadpovprecen (19.0/6.0)
|   |   |   ETOG2 > -0.357143: podpovprecen (115.0/39.0)
|   ETOG1 > -0.071429
|   |   ETOG5 <= -0.214286: podpovprecen (35.0/11.0)
|   |   ETOG5 > -0.214286
```

```
|   |   |   ETOG4 <= -0.214286
|   |   |   |   ETOG5 <= 0.071429: podpovprecen (8.0)
|   |   |   |   ETOG5 > 0.071429
|   |   |   |   |   ETOG3 <= -0.214286: nadpovprecen (7.0/1.0)
|   |   |   |   |   ETOG3 > -0.214286: podpovprecen (4.0/1.0)
|   |   |   ETOG4 > -0.214286: nadpovprecen (59.0/20.0)
ETOG1 > 0.071429: nadpovprecen (283.0/68.0)

Number of Leaves  :     9

Size of the tree :      17

Delnica ETOG je klasificirana   nadpovprecen

porazdelitev verjetnosti za delnico ETOG za podpovprecen znasa 24,03%

porazdelitev verjetnosti za delnico ETOG za nadpovprecen znasa 75,97%
=====================================================================
Tole je evaluacija z drevesom za delnico GRVG:

Correctly Classified Instances         272               69.7436 %
Incorrectly Classified Instances       118               30.2564 %
Kappa statistic                          0.3939
Mean absolute error                      0.3829
Root mean squared error                  0.4576
Relative absolute error                 76.6309 %
Root relative squared error             91.5279 %
Total Number of Instances              390

 Tole je graf za delnico GRVG:J48 pruned tree
------------------

GRVG1 <= -0.071429
|   GRVG1 <= -0.357143: podpovprecen (129.0/13.0)
|   GRVG1 > -0.357143
|   |   GRVG3 <= 0.214286: podpovprecen (238.0/74.0)
|   |   GRVG3 > 0.214286
|   |   |   GRVG1 <= -0.214286: nadpovprecen (5.0)
|   |   |   GRVG1 > -0.214286
|   |   |   |   GRVG2 <= 0.214286: podpovprecen (9.0/2.0)
|   |   |   |   GRVG2 > 0.214286: nadpovprecen (5.0/1.0)
GRVG1 > -0.071429
|   GRVG1 <= 0.071429
|   |   GRVG3 <= 0.214286: nadpovprecen (119.0/47.0)
|   |   GRVG3 > 0.214286
|   |   |   GRVG3 <= 0.357143
|   |   |   |   GRVG4 <= 0.357143
|   |   |   |   |   GRVG5 <= 0.071429: nadpovprecen (4.0/1.0)
|   |   |   |   |   GRVG5 > 0.071429: podpovprecen (11.0/3.0)
|   |   |   |   GRVG4 > 0.357143: nadpovprecen (2.0)
|   |   |   GRVG3 > 0.357143: podpovprecen (5.0)
|   GRVG1 > 0.071429: nadpovprecen (253.0/52.0)

Number of Leaves  :     11

Size of the tree :      21

Delnica GRVG je klasificirana   nadpovprecen

porazdelitev verjetnosti za delnico GRVG za podpovprecen znasa 20,55%
```

```
porazdelitev verjetnosti za delnico GRVG za nadpovprecen znasa 79,45%
=====================================================================
Tole je evaluacija z drevesom za delnico IEKG:

Correctly Classified Instances         258               66.1538 %
Incorrectly Classified Instances       132               33.8462 %
Kappa statistic                          0.308
Mean absolute error                      0.4003
Root mean squared error                  0.4788
Relative absolute error                 80.4513 %
Root relative squared error             96.09   %
Total Number of Instances              390

 Tole je graf za delnico IEKG:J48 pruned tree
------------------

IEKG1 <= -0.071429
|   IEKG1 <= -0.357143: podpovprecen (83.0/11.0)
|   IEKG1 > -0.357143
|   |   IEKG3 <= 0.071429
|   |   |   IEKG5 <= 0.071429
|   |   |   |   IEKG1 <= -0.214286: podpovprecen (103.0/23.0)
|   |   |   |   IEKG1 > -0.214286
|   |   |   |   |   IEKG3 <= -0.071429: podpovprecen (67.0/20.0)
|   |   |   |   |   IEKG3 > -0.071429
|   |   |   |   |   |   IEKG2 <= -0.214286: nadpovprecen (4.0)
|   |   |   |   |   |   IEKG2 > -0.214286
|   |   |   |   |   |   |   IEKG2 <= 0.071429: podpovprecen (16.0/4.0)
|   |   |   |   |   |   |   IEKG2 > 0.071429: nadpovprecen (3.0)
|   |   |   IEKG5 > 0.071429: nadpovprecen (42.0/18.0)
|   |   IEKG3 > 0.071429
|   |   |   IEKG1 <= -0.214286
|   |   |   |   IEKG5 <= -0.214286: nadpovprecen (2.0)
|   |   |   |   IEKG5 > -0.214286
|   |   |   |   |   IEKG4 <= 0.071429: podpovprecen (5.0/1.0)
|   |   |   |   |   IEKG4 > 0.071429
|   |   |   |   |   |   IEKG2 <= 0.071429: nadpovprecen (4.0)
|   |   |   |   |   |   IEKG2 > 0.071429: podpovprecen (5.0/2.0)
|   |   |   IEKG1 > -0.214286
|   |   |   |   IEKG2 <= 0.071429: nadpovprecen (16.0/4.0)
|   |   |   |   IEKG2 > 0.071429
|   |   |   |   |   IEKG4 <= 0.071429: podpovprecen (3.0)
|   |   |   |   |   IEKG4 > 0.071429: nadpovprecen (4.0/1.0)
IEKG1 > -0.071429
|   IEKG1 <= 0.071429
|   |   IEKG3 <= 0.214286
|   |   |   IEKG2 <= -0.071429
|   |   |   |   IEKG4 <= -0.214286: nadpovprecen (20.0/4.0)
|   |   |   |   IEKG4 > -0.214286
|   |   |   |   |   IEKG3 <= -0.357143: nadpovprecen (2.0)
|   |   |   |   |   IEKG3 > -0.357143: podpovprecen (37.0/12.0)
|   |   |   IEKG2 > -0.071429
|   |   |   |   IEKG4 <= -0.071429
|   |   |   |   |   IEKG4 <= -0.357143: podpovprecen (2.0)
|   |   |   |   |   IEKG4 > -0.357143
|   |   |   |   |   |   IEKG5 <= 0.214286: nadpovprecen (32.0/14.0)
|   |   |   |   |   |   IEKG5 > 0.214286: podpovprecen (6.0/1.0)
|   |   |   |   IEKG4 > -0.071429: nadpovprecen (67.0/18.0)
|   |   IEKG3 > 0.214286: podpovprecen (15.0/5.0)
|   IEKG1 > 0.071429: nadpovprecen (242.0/47.0)
```

```
Number of Leaves  :     23

Size of the tree :      45

Delnica IEKG je klasificirana   podpovprecen

porazdelitev verjetnosti za delnico IEKG za podpovprecen znasa 86,75%

porazdelitev verjetnosti za delnico IEKG za nadpovprecen znasa 13,25%
=====================================================================
Tole je evaluacija z drevesom za delnico KRKG:

Correctly Classified Instances         297               76.1538 %
Incorrectly Classified Instances        93               23.8462 %
Kappa statistic                          0.5201
Mean absolute error                      0.3368
Root mean squared error                  0.4264
Relative absolute error                 67.3049 %
Root relative squared error             85.1967 %
Total Number of Instances              390

 Tole je graf za delnico KRKG:J48 pruned tree
------------------

KRKG1 <= -0.071429
|   KRKG1 <= -0.214286: podpovprecen (198.0/24.0)
|   KRKG1 > -0.214286
|   |   KRKG3 <= -0.071429: podpovprecen (142.0/46.0)
|   |   KRKG3 > -0.071429
|   |   |   KRKG4 <= -0.071429
|   |   |   |   KRKG3 <= 0.071429: nadpovprecen (12.0/3.0)
|   |   |   |   KRKG3 > 0.071429: podpovprecen (3.0)
|   |   |   KRKG4 > -0.071429
|   |   |   |   KRKG5 <= 0.214286: podpovprecen (34.0/9.0)
|   |   |   |   KRKG5 > 0.214286: nadpovprecen (8.0/2.0)
KRKG1 > -0.071429: nadpovprecen (383.0/83.0)

Number of Leaves  :     7

Size of the tree :      13

Delnica KRKG je klasificirana   podpovprecen

porazdelitev verjetnosti za delnico KRKG za podpovprecen znasa 67,61%

porazdelitev verjetnosti za delnico KRKG za nadpovprecen znasa 32,39%
=====================================================================
Tole je evaluacija z drevesom za delnico LKPG:

Correctly Classified Instances         284               72.8205 %
Incorrectly Classified Instances       106               27.1795 %
Kappa statistic                          0.4564
Mean absolute error                      0.3985
Root mean squared error                  0.4443
Relative absolute error                 79.7017 %
Root relative squared error             88.8532 %
Total Number of Instances              390

 Tole je graf za delnico LKPG:J48 pruned tree
------------------
```

```
LKPG1 <= -0.071429
|   LKPG2 <= -0.214286
|   |   LKPG3 <= -0.071429: podpovprecen (81.0/13.0)
|   |   LKPG3 > -0.071429: nadpovprecen (6.0/1.0)
|   LKPG2 > -0.214286: podpovprecen (322.0/104.0)
LKPG1 > -0.071429: nadpovprecen (371.0/106.0)

Number of Leaves  :     4

Size of the tree :      7

Delnica LKPG je klasificirana   nadpovprecen

porazdelitev verjetnosti za delnico LKPG za podpovprecen znasa 28,57%

porazdelitev verjetnosti za delnico LKPG za nadpovprecen znasa 71,43%
=====================================================================
Tole je evaluacija z drevesom za delnico MER:

Correctly Classified Instances         283               72.5641 %
Incorrectly Classified Instances       107               27.4359 %
Kappa statistic                          0.4513
Mean absolute error                      0.3905
Root mean squared error                  0.4442
Relative absolute error                 78.1011 %
Root relative squared error             88.8154 %
Total Number of Instances              390

 Tole je graf za delnico MER:J48 pruned tree
------------------

MER1 <= -0.071429: podpovprecen (408.0/113.0)
MER1 > -0.071429
|   MER2 <= -0.071429
|   |   MER2 <= -0.214286
|   |   |   MER4 <= -0.214286: podpovprecen (4.0)
|   |   |   MER4 > -0.214286
|   |   |   |   MER4 <= -0.071429: nadpovprecen (3.0)
|   |   |   |   MER4 > -0.071429: podpovprecen (3.0)
|   |   MER2 > -0.214286
|   |   |   MER3 <= -0.214286: nadpovprecen (13.0/3.0)
|   |   |   MER3 > -0.214286
|   |   |   |   MER5 <= -0.214286: podpovprecen (13.0/4.0)
|   |   |   |   MER5 > -0.214286: nadpovprecen (61.0/23.0)
|   MER2 > -0.071429: nadpovprecen (275.0/63.0)

Number of Leaves  :     8

Size of the tree :      15

Delnica MER je klasificirana   podpovprecen

porazdelitev verjetnosti za delnico MER za podpovprecen znasa 72,3%

porazdelitev verjetnosti za delnico MER za nadpovprecen znasa 27,7%
=====================================================================
Tole je evaluacija z drevesom za delnico TCRG:

Correctly Classified Instances         272               69.7436 %
Incorrectly Classified Instances       118               30.2564 %
Kappa statistic                          0.3947
```

```
Mean absolute error                      0.4096
Root mean squared error                  0.4606
Relative absolute error                 81.8897 %
Root relative squared error             92.04   %
Total Number of Instances              390

 Tole je graf za delnico TCRG:J48 pruned tree
------------------

TCRG1 <= -0.071429: podpovprecen (397.0/103.0)
TCRG1 > -0.071429: nadpovprecen (383.0/108.0)

Number of Leaves  :     2

Size of the tree :      3

Delnica TCRG je klasificirana   nadpovprecen

porazdelitev verjetnosti za delnico TCRG za podpovprecen znasa 28,2%

porazdelitev verjetnosti za delnico TCRG za nadpovprecen znasa 71,8%



================================================================

                        POVZETEK ANALIZE

================================================================



DELNICA    INDEX  NAD         POD   ZANESLJIVOST      VELIDREVESA PONDER
AELG        0     0,28        0,72        0           0.0         0
ETOG        1     0,76        0,24        68,97       17.0        22,85
GRVG        2     0,79        0,21        69,74       21.0        23,5
IEKG        3     0,13        0,87        0           0.0         0
KRKG        4     0,32        0,68        0           0.0         0
LKPG        5     0,71        0,29        72,82       7.0         22,97
MER         6     0,28        0,72        0           0.0         0
TCRG        7     0,72        0,28        69,74       3.0         21,65

Potencialno najbolj donosna delnica je GRVG
Dosezeni donos za dan 1 za delnico GRVG bi znasal: 0,90339%

Skupni donos za obdobje 1 dni bi znasal:  0,90339%
```

**Priloga B**

Izpis hipotetičnega trgovanja za obdobje 50-ih dni z uporabo **odločitvenih dreves**.

Ponder s katerim upoštevamo verjetnost, da bo delnica kotirala nadpovprečno je izbran v višini 0,7, ponder za upoštevanje učinkovitosti drevesa v višini 0,3 in ponder za upoštevanje velikosti drevesa 0,0.

```
Dosezeni Donos za dan 1 za delnico  LKPG na dan: 2003-07-15 bi znasal: -0,05583% Skupni donos pa    -0,05583%
Dosezeni Donos za dan 2 za delnico  LKPG na dan: 2003-07-16 bi znasal: 0,35516% Skupni donos pa     0,29933%
Dosezeni Donos za dan 3 za delnico  LKPG na dan: 2003-07-17 bi znasal: 0,75536% Skupni donos pa     1,0547%
Dosezeni Donos za dan 4 za delnico  LKPG na dan: 2003-07-18 bi znasal: -0,53416% Skupni donos pa    0,52053%
Dosezeni Donos za dan 5 za delnico  LKPG na dan: 2003-07-21 bi znasal: 1,42436% Skupni donos pa     1,94489%
Dosezeni Donos za dan 6 za delnico  LKPG na dan: 2003-07-22 bi znasal: -0,83627% Skupni donos pa    1,10862%
Dosezeni Donos za dan 7 za delnico  LKPG na dan: 2003-07-23 bi znasal: 0,56526% Skupni donos pa     1,67388%
Dosezeni Donos za dan 8 za delnico  LKPG na dan: 2003-07-24 bi znasal: 0,34806% Skupni donos pa     2,02194%
Dosezeni Donos za dan 9 za delnico  LKPG na dan: 2003-07-25 bi znasal: 0,58642% Skupni donos pa     2,60836%
Dosezeni Donos za dan 10 za delnico  LKPG na dan: 2003-07-28 bi znasal: 0,10323% Skupni donos pa    2,71159%
Dosezeni Donos za dan 11 za delnico  LKPG na dan: 2003-07-29 bi znasal: 1,28761% Skupni donos pa    3,9992%
Dosezeni Donos za dan 12 za delnico  LKPG na dan: 2003-07-30 bi znasal: 1,23955% Skupni donos pa    5,23875%
Dosezeni Donos za dan 13 za delnico  LKPG na dan: 2003-07-31 bi znasal: -0,04882% Skupni donos pa   5,18993%
Dosezeni Donos za dan 14 za delnico  LKPG na dan: 2003-08-01 bi znasal: 1,17112% Skupni donos pa    6,36105%
Dosezeni Donos za dan 15 za delnico  LKPG na dan: 2003-08-04 bi znasal: 0,26411% Skupni donos pa    6,62516%
Dosezeni Donos za dan 16 za delnico  LKPG na dan: 2003-08-05 bi znasal: -0,35026% Skupni donos pa   6,2749%
Dosezeni Donos za dan 17 za delnico  LKPG na dan: 2003-08-06 bi znasal: 1,07968% Skupni donos pa    7,35457%
Dosezeni Donos za dan 18 za delnico  LKPG na dan: 2003-08-07 bi znasal: 0,12586% Skupni donos pa    7,48043%
Dosezeni Donos za dan 19 za delnico  LKPG na dan: 2003-08-08 bi znasal: -0,12141% Skupni donos pa   7,35902%
Dosezeni Donos za dan 20 za delnico  LKPG na dan: 2003-08-11 bi znasal: -0,13116% Skupni donos pa   7,22785%
Dosezeni Donos za dan 21 za delnico  LKPG na dan: 2003-08-12 bi znasal: -0,1925% Skupni donos pa    7,03535%
Dosezeni Donos za dan 22 za delnico  LKPG na dan: 2003-08-13 bi znasal: 0,02193% Skupni donos pa    7,05728%
Dosezeni Donos za dan 23 za delnico  LKPG na dan: 2003-08-14 bi znasal: 0,28771% Skupni donos pa    7,34499%
Dosezeni Donos za dan 24 za delnico  LKPG na dan: 2003-08-18 bi znasal: 0,36719% Skupni donos pa    7,71218%
Dosezeni Donos za dan 25 za delnico  LKPG na dan: 2003-08-19 bi znasal: 1,69608% Skupni donos pa    9,40826%
Dosezeni Donos za dan 26 za delnico  LKPG na dan: 2003-08-20 bi znasal: 0,6332% Skupni donos pa     10,04146%
Dosezeni Donos za dan 27 za delnico  LKPG na dan: 2003-08-21 bi znasal: 0,09449% Skupni donos pa    10,13596%
Dosezeni Donos za dan 28 za delnico  LKPG na dan: 2003-08-22 bi znasal: -0,62823% Skupni donos pa   9,50773%
```

```
Dosezeni Donos za dan 29 za delnico  LKPG na dan: 2003-08-25 bi znasal: 0,311% Skupni donos pa      9,81873%
Dosezeni Donos za dan 30 za delnico  LKPG na dan: 2003-08-26 bi znasal: 0,65317% Skupni donos pa    10,4719%
Dosezeni Donos za dan 31 za delnico  LKPG na dan: 2003-08-27 bi znasal: 0,37775% Skupni donos pa    10,84965%
Dosezeni Donos za dan 32 za delnico  LKPG na dan: 2003-08-28 bi znasal: 0,49848% Skupni donos pa    11,34813%
Dosezeni Donos za dan 33 za delnico  LKPG na dan: 2003-08-29 bi znasal: 0,00805% Skupni donos pa    11,35618%
Dosezeni Donos za dan 34 za delnico  LKPG na dan: 2003-09-01 bi znasal: 0,55232% Skupni donos pa    11,9085%
Dosezeni Donos za dan 35 za delnico  LKPG na dan: 2003-09-02 bi znasal: 1,51938% Skupni donos pa    13,42788%
Dosezeni Donos za dan 36 za delnico  LKPG na dan: 2003-09-03 bi znasal: -0,1508% Skupni donos pa    13,27708%
Dosezeni Donos za dan 37 za delnico  LKPG na dan: 2003-09-04 bi znasal: 0,17048% Skupni donos pa    13,44756%
Dosezeni Donos za dan 38 za delnico  LKPG na dan: 2003-09-05 bi znasal: 0,17635% Skupni donos pa    13,62391%
Dosezeni Donos za dan 39 za delnico  LKPG na dan: 2003-09-08 bi znasal: 0,68879% Skupni donos pa    14,31269%
Dosezeni Donos za dan 40 za delnico  LKPG na dan: 2003-09-09 bi znasal: 0,88655% Skupni donos pa    15,19924%
Dosezeni Donos za dan 41 za delnico  LKPG na dan: 2003-09-10 bi znasal: 1,11536% Skupni donos pa    16,31461%
Dosezeni Donos za dan 42 za delnico  LKPG na dan: 2003-09-11 bi znasal: 0,78758% Skupni donos pa    17,10219%
Dosezeni Donos za dan 43 za delnico  LKPG na dan: 2003-09-12 bi znasal: 0,36827% Skupni donos pa    17,47046%
Dosezeni Donos za dan 44 za delnico  LKPG na dan: 2003-09-15 bi znasal: 0,27436% Skupni donos pa    17,74482%
Dosezeni Donos za dan 45 za delnico  LKPG na dan: 2003-09-16 bi znasal: 1,33563% Skupni donos pa    19,08045%
Dosezeni Donos za dan 46 za delnico  LKPG na dan: 2003-09-17 bi znasal: 0,7511% Skupni donos pa     19,83155%
Dosezeni Donos za dan 47 za delnico  LKPG na dan: 2003-09-18 bi znasal: 0,65978% Skupni donos pa    20,49133%
Dosezeni Donos za dan 48 za delnico  LKPG na dan: 2003-09-19 bi znasal: 1,46008% Skupni donos pa    21,95141%
Dosezeni Donos za dan 49 za delnico  LKPG na dan: 2003-09-22 bi znasal: -0,42161% Skupni donos pa   21,5298%
Dosezeni Donos za dan 50 za delnico  LKPG na dan: 2003-09-23 bi znasal: 1,48739% Skupni donos pa    23,01719%
Skupni donos za obdobje 50 dni bi znasal:  23,01719%
```

**Priloga C**

Izpis hipotetičnega trgovanja za obdobje 50-ih dni z metodo **k - najbližjih sosedov**.

Ponder s katerim upoštevamo verjetnost, da bo delnica kotirala nadpovprečno je izbran v višini 1,0, ponder za upoštevanje učinkovitosti modela v višini 0,0 .

```
Dosezeni Donos za dan 1 za delnico  LKPG na dan: 2003-07-15 bi znasal: -0,05583% Skupni donos pa    -0,05583%
Dosezeni Donos za dan 2 za delnico  LKPG na dan: 2003-07-16 bi znasal: 0,35516% Skupni donos pa     0,29933%
Dosezeni Donos za dan 3 za delnico  LKPG na dan: 2003-07-17 bi znasal: 0,75536% Skupni donos pa     1,0547%
Dosezeni Donos za dan 4 za delnico  LKPG na dan: 2003-07-18 bi znasal: -0,53416% Skupni donos pa    0,52053%
Dosezeni Donos za dan 5 za delnico  LKPG na dan: 2003-07-21 bi znasal: 1,42436% Skupni donos pa     1,94489%
Dosezeni Donos za dan 6 za delnico  LKPG na dan: 2003-07-22 bi znasal: -0,83627% Skupni donos pa    1,10862%
Dosezeni Donos za dan 7 za delnico  LKPG na dan: 2003-07-23 bi znasal: 0,56526% Skupni donos pa     1,67388%
Dosezeni Donos za dan 8 za delnico  LKPG na dan: 2003-07-24 bi znasal: 0,34806% Skupni donos pa     2,02194%
Dosezeni Donos za dan 9 za delnico  LKPG na dan: 2003-07-25 bi znasal: 0,58642% Skupni donos pa     2,60836%
Dosezeni Donos za dan 10 za delnico  LKPG na dan: 2003-07-28 bi znasal: 0,10323% Skupni donos pa    2,71159%
Dosezeni Donos za dan 11 za delnico  LKPG na dan: 2003-07-29 bi znasal: 1,28761% Skupni donos pa    3,9992%
Dosezeni Donos za dan 12 za delnico  LKPG na dan: 2003-07-30 bi znasal: 1,23955% Skupni donos pa    5,23875%
Dosezeni Donos za dan 13 za delnico  LKPG na dan: 2003-07-31 bi znasal: -0,04882% Skupni donos pa   5,18993%
Dosezeni Donos za dan 14 za delnico  LKPG na dan: 2003-08-01 bi znasal: 1,17112% Skupni donos pa    6,36105%
Dosezeni Donos za dan 15 za delnico  LKPG na dan: 2003-08-04 bi znasal: 0,26411% Skupni donos pa    6,62516%
Dosezeni Donos za dan 16 za delnico  LKPG na dan: 2003-08-05 bi znasal: -0,35026% Skupni donos pa   6,2749%
Dosezeni Donos za dan 17 za delnico  LKPG na dan: 2003-08-06 bi znasal: 1,07968% Skupni donos pa    7,35457%
Dosezeni Donos za dan 18 za delnico  LKPG na dan: 2003-08-07 bi znasal: 0,12586% Skupni donos pa    7,48043%
Dosezeni Donos za dan 19 za delnico  LKPG na dan: 2003-08-08 bi znasal: -0,12141% Skupni donos pa   7,35902%
Dosezeni Donos za dan 20 za delnico  LKPG na dan: 2003-08-11 bi znasal: -0,13116% Skupni donos pa   7,22785%
Dosezeni Donos za dan 21 za delnico  LKPG na dan: 2003-08-12 bi znasal: -0,1925% Skupni donos pa    7,03535%
Dosezeni Donos za dan 22 za delnico  LKPG na dan: 2003-08-13 bi znasal: 0,02193% Skupni donos pa    7,05728%
Dosezeni Donos za dan 23 za delnico  LKPG na dan: 2003-08-14 bi znasal: 0,28771% Skupni donos pa    7,34499%
Dosezeni Donos za dan 24 za delnico  LKPG na dan: 2003-08-18 bi znasal: 0,36719% Skupni donos pa    7,71218%
Dosezeni Donos za dan 25 za delnico  LKPG na dan: 2003-08-19 bi znasal: 1,69608% Skupni donos pa    9,40826%
Dosezeni Donos za dan 26 za delnico  LKPG na dan: 2003-08-20 bi znasal: 0,6332% Skupni donos pa     10,04146%
Dosezeni Donos za dan 27 za delnico  LKPG na dan: 2003-08-21 bi znasal: 0,09449% Skupni donos pa    10,13596%
Dosezeni Donos za dan 28 za delnico  LKPG na dan: 2003-08-22 bi znasal: -0,62823% Skupni donos pa   9,50773%
Dosezeni Donos za dan 29 za delnico  LKPG na dan: 2003-08-25 bi znasal:      0,311% Skupni donos pa 9,81873%
Dosezeni Donos za dan 30 za delnico  LKPG na dan: 2003-08-26 bi znasal: 0,65317% Skupni donos pa    10,4719%
```

```
Dosezeni Donos za dan 31 za delnico  LKPG na dan: 2003-08-27 bi znasal: 0,37775% Skupni donos pa    10,84965%
Dosezeni Donos za dan 32 za delnico  LKPG na dan: 2003-08-28 bi znasal: 0,49848% Skupni donos pa    11,34813%
Dosezeni Donos za dan 33 za delnico  LKPG na dan: 2003-08-29 bi znasal: 0,00805% Skupni donos pa    11,35618%
Dosezeni Donos za dan 34 za delnico  LKPG na dan: 2003-09-01 bi znasal: 0,55232% Skupni donos pa    11,9085%
Dosezeni Donos za dan 35 za delnico  LKPG na dan: 2003-09-02 bi znasal: 1,51938% Skupni donos pa    13,42788%
Dosezeni Donos za dan 36 za delnico  LKPG na dan: 2003-09-03 bi znasal: -0,1508% Skupni donos pa    13,27708%
Dosezeni Donos za dan 37 za delnico  LKPG na dan: 2003-09-04 bi znasal: 0,17048% Skupni donos pa    13,44756%
Dosezeni Donos za dan 38 za delnico  LKPG na dan: 2003-09-05 bi znasal: 0,17635% Skupni donos pa    13,62391%
Dosezeni Donos za dan 39 za delnico  LKPG na dan: 2003-09-08 bi znasal: 0,68879% Skupni donos pa    14,31269%
Dosezeni Donos za dan 40 za delnico  LKPG na dan: 2003-09-09 bi znasal: 0,88655% Skupni donos pa    15,19924%
Dosezeni Donos za dan 41 za delnico  LKPG na dan: 2003-09-10 bi znasal: 1,11536% Skupni donos pa    16,31461%
Dosezeni Donos za dan 42 za delnico  LKPG na dan: 2003-09-11 bi znasal: 0,78758% Skupni donos pa    17,10219%
Dosezeni Donos za dan 43 za delnico  LKPG na dan: 2003-09-12 bi znasal: 0,36827% Skupni donos pa    17,47046%
Dosezeni Donos za dan 44 za delnico  LKPG na dan: 2003-09-15 bi znasal: 0,27436% Skupni donos pa    17,74482%
Dosezeni Donos za dan 45 za delnico  LKPG na dan: 2003-09-16 bi znasal: 1,33563% Skupni donos pa    19,08045%
Dosezeni Donos za dan 46 za delnico  LKPG na dan: 2003-09-17 bi znasal: 0,7511% Skupni donos pa     19,83155%
Dosezeni Donos za dan 47 za delnico  LKPG na dan: 2003-09-18 bi znasal: 0,65978% Skupni donos pa    20,49133%
Dosezeni Donos za dan 48 za delnico  LKPG na dan: 2003-09-19 bi znasal: 1,46008% Skupni donos pa    21,95141%
Dosezeni Donos za dan 49 za delnico  LKPG na dan: 2003-09-22 bi znasal: -0,42161% Skupni donos pa   21,5298%
Dosezeni Donos za dan 50 za delnico  LKPG na dan: 2003-09-23 bi znasal: 1,48739% Skupni donos pa    23,01719%

Skupni donos za obdobje 50 dni bi znasal:  23,01719%
```

**Priloga D**

Izpis hipotetičnega trgovanja za obdobje 50-ih dni z **nevronskimi mrežami**.

Ponder s katerim upoštevamo verjetnost, da bo delnica kotirala nadpovprečno je izbran v višini 0,7, ponder za upoštevanje učinkovitosti modela v višini 0,3.

```
Dosezeni Donos za dan 1 za delnico  ETOG na dan: 2003-07-15 bi znasal: -1,02564% Skupni donos pa    -1,02564%
Dosezeni Donos za dan 2 za delnico  LKPG na dan: 2003-07-16 bi znasal: 0,35516% Skupni donos pa     -0,67048%
Dosezeni Donos za dan 3 za delnico  LKPG na dan: 2003-07-17 bi znasal: 0,75536% Skupni donos pa     0,08488%
Dosezeni Donos za dan 4 za delnico  LKPG na dan: 2003-07-18 bi znasal: -0,53416% Skupni donos pa    -0,44928%
Dosezeni Donos za dan 5 za delnico  LKPG na dan: 2003-07-21 bi znasal: 1,42436% Skupni donos pa     0,97508%
Dosezeni Donos za dan 6 za delnico  GRVG na dan: 2003-07-22 bi znasal: -0,61294% Skupni donos pa    0,36213%
Dosezeni Donos za dan 7 za delnico  LKPG na dan: 2003-07-23 bi znasal: 0,56526% Skupni donos pa     0,92739%
Dosezeni Donos za dan 8 za delnico  LKPG na dan: 2003-07-24 bi znasal: 0,34806% Skupni donos pa     1,27545%
Dosezeni Donos za dan 9 za delnico  LKPG na dan: 2003-07-25 bi znasal: 0,58642% Skupni donos pa     1,86187%
Dosezeni Donos za dan 10 za delnico  LKPG na dan: 2003-07-28 bi znasal: 0,10323% Skupni donos pa    1,96511%
Dosezeni Donos za dan 11 za delnico  GRVG na dan: 2003-07-29 bi znasal: 1,37762% Skupni donos pa    3,34273%
Dosezeni Donos za dan 12 za delnico  LKPG na dan: 2003-07-30 bi znasal: 1,23955% Skupni donos pa    4,58228%
Dosezeni Donos za dan 13 za delnico  ETOG na dan: 2003-07-31 bi znasal:      0% Skupni donos pa     4,58228%
Dosezeni Donos za dan 14 za delnico  ETOG na dan: 2003-08-01 bi znasal:      0% Skupni donos pa     4,58228%
Dosezeni Donos za dan 15 za delnico  GRVG na dan: 2003-08-04 bi znasal: 0,53923% Skupni donos pa    5,12151%
Dosezeni Donos za dan 16 za delnico  LKPG na dan: 2003-08-05 bi znasal: -0,35026% Skupni donos pa   4,77125%
Dosezeni Donos za dan 17 za delnico  LKPG na dan: 2003-08-06 bi znasal: 1,07968% Skupni donos pa    5,85093%
Dosezeni Donos za dan 18 za delnico  LKPG na dan: 2003-08-07 bi znasal: 0,12586% Skupni donos pa    5,97679%
Dosezeni Donos za dan 19 za delnico  LKPG na dan: 2003-08-08 bi znasal: -0,12141% Skupni donos pa   5,85537%
Dosezeni Donos za dan 20 za delnico  ETOG na dan: 2003-08-11 bi znasal:      0% Skupni donos pa     5,85537%
Dosezeni Donos za dan 21 za delnico  LKPG na dan: 2003-08-12 bi znasal: -0,1925% Skupni donos pa    5,66287%
Dosezeni Donos za dan 22 za delnico  LKPG na dan: 2003-08-13 bi znasal: 0,02193% Skupni donos pa    5,6848%
Dosezeni Donos za dan 23 za delnico  LKPG na dan: 2003-08-14 bi znasal: 0,28771% Skupni donos pa    5,97251%
Dosezeni Donos za dan 24 za delnico  LKPG na dan: 2003-08-18 bi znasal: 0,36719% Skupni donos pa    6,3397%
Dosezeni Donos za dan 25 za delnico  LKPG na dan: 2003-08-19 bi znasal: 1,69608% Skupni donos pa    8,03578%
Dosezeni Donos za dan 26 za delnico  GRVG na dan: 2003-08-20 bi znasal: 2,30086% Skupni donos pa    10,33664%
Dosezeni Donos za dan 27 za delnico  LKPG na dan: 2003-08-21 bi znasal: 0,09449% Skupni donos pa    10,43113%
Dosezeni Donos za dan 28 za delnico  ETOG na dan: 2003-08-22 bi znasal: 3,50096% Skupni donos pa    13,93209%
Dosezeni Donos za dan 29 za delnico  LKPG na dan: 2003-08-25 bi znasal: 0,311% Skupni donos pa      14,24309%
```

```
Dosezeni Donos za dan 30 za delnico  LKPG na dan: 2003-08-26 bi znasal: 0,65317% Skupni donos pa    14,89625%
Dosezeni Donos za dan 31 za delnico  GRVG na dan: 2003-08-27 bi znasal: 1,63316% Skupni donos pa    16,52941%
Dosezeni Donos za dan 32 za delnico  LKPG na dan: 2003-08-28 bi znasal: 0,49848% Skupni donos pa    17,0279%
Dosezeni Donos za dan 33 za delnico  GRVG na dan: 2003-08-29 bi znasal: -0,10385% Skupni donos pa   16,92405%
Dosezeni Donos za dan 34 za delnico  GRVG na dan: 2003-09-01 bi znasal: 1,89887% Skupni donos pa    18,82292%
Dosezeni Donos za dan 35 za delnico  ETOG na dan: 2003-09-02 bi znasal:      0% Skupni donos pa     18,82292%
Dosezeni Donos za dan 36 za delnico  LKPG na dan: 2003-09-03 bi znasal: -0,1508% Skupni donos pa    18,67212%
Dosezeni Donos za dan 37 za delnico  LKPG na dan: 2003-09-04 bi znasal: 0,17048% Skupni donos pa    18,84261%
Dosezeni Donos za dan 38 za delnico  ETOG na dan: 2003-09-05 bi znasal: -1,53681% Skupni donos pa   17,3058%
Dosezeni Donos za dan 39 za delnico  ETOG na dan: 2003-09-08 bi znasal:      0% Skupni donos pa     17,3058%
Dosezeni Donos za dan 40 za delnico  TCRG na dan: 2003-09-09 bi znasal:      0% Skupni donos pa     17,3058%
Dosezeni Donos za dan 41 za delnico  LKPG na dan: 2003-09-10 bi znasal: 1,11536% Skupni donos pa    18,42116%
Dosezeni Donos za dan 42 za delnico  ETOG na dan: 2003-09-11 bi znasal: -1,17647% Skupni donos pa   17,24469%
Dosezeni Donos za dan 43 za delnico  LKPG na dan: 2003-09-12 bi znasal: 0,36827% Skupni donos pa    17,61296%
Dosezeni Donos za dan 44 za delnico  ETOG na dan: 2003-09-15 bi znasal: 1,19048% Skupni donos pa    18,80344%
Dosezeni Donos za dan 45 za delnico  ETOG na dan: 2003-09-16 bi znasal:      0% Skupni donos pa     18,80344%
Dosezeni Donos za dan 46 za delnico  LKPG na dan: 2003-09-17 bi znasal: 0,7511% Skupni donos pa     19,55454%
Dosezeni Donos za dan 47 za delnico  LKPG na dan: 2003-09-18 bi znasal: 0,65978% Skupni donos pa    20,21433%
Dosezeni Donos za dan 48 za delnico  LKPG na dan: 2003-09-19 bi znasal: 1,46008% Skupni donos pa    21,67441%
Dosezeni Donos za dan 49 za delnico  LKPG na dan: 2003-09-22 bi znasal: -0,42161% Skupni donos pa   21,25279%
Dosezeni Donos za dan 50 za delnico  GRVG na dan: 2003-09-23 bi znasal: 0,90339% Skupni donos pa    22,15618%

Skupni donos za obdobje 50 dni bi znasal:  22,15618%
```

**Priloga E**

Izpis hipotetičnega trgovanja za obdobje 150-ih dni z metodo **nevronskih mrež**.

Ponder s katerim upoštevamo verjetnost, da bo delnica kotirala nadpovprečno je izbran v višini 0,8, ponder za upoštevanje učinkovitosti modela v višini 0,2 .

```
Dosezeni Donos za dan 1 za delnico  GRVG na dan: 2003-02-18 bi znasal: 0,75341% Skupni donos pa     0,75341%
Dosezeni Donos za dan 2 za delnico  LKPG na dan: 2003-02-19 bi znasal: -0,03846% Skupni donos pa    0,71496%
Dosezeni Donos za dan 3 za delnico  LKPG na dan: 2003-02-20 bi znasal: -1,10007% Skupni donos pa    -0,38512%
Dosezeni Donos za dan 4 za delnico  LKPG na dan: 2003-02-21 bi znasal: -0,29275% Skupni donos pa    -0,67786%
Dosezeni Donos za dan 5 za delnico  LKPG na dan: 2003-02-24 bi znasal: -0,66987% Skupni donos pa    -1,34774%
Dosezeni Donos za dan 6 za delnico  LKPG na dan: 2003-02-25 bi znasal: 0,23434% Skupni donos pa     -1,1134%
Dosezeni Donos za dan 7 za delnico  LKPG na dan: 2003-02-26 bi znasal: -0,38301% Skupni donos pa    -1,49641%
Dosezeni Donos za dan 8 za delnico  TCRG na dan: 2003-02-27 bi znasal: 0% Skupni donos pa     -1,49641%
Dosezeni Donos za dan 9 za delnico  TCRG na dan: 2003-02-28 bi znasal: 5,09814% Skupni donos pa     3,60173%
Dosezeni Donos za dan 10 za delnico  TCRG na dan: 2003-03-03 bi znasal: -1,34947% Skupni donos pa   2,25226%
Dosezeni Donos za dan 11 za delnico  LKPG na dan: 2003-03-04 bi znasal: 0,62943% Skupni donos pa    2,8817%
Dosezeni Donos za dan 12 za delnico  LKPG na dan: 2003-03-05 bi znasal: 0,51331% Skupni donos pa    3,39501%
Dosezeni Donos za dan 13 za delnico  KRKG na dan: 2003-03-06 bi znasal: -0,6526% Skupni donos pa    2,74241%
Dosezeni Donos za dan 14 za delnico  ETOG na dan: 2003-03-07 bi znasal: 0% Skupni donos pa    2,74241%
Dosezeni Donos za dan 15 za delnico  ETOG na dan: 2003-03-10 bi znasal: 0% Skupni donos pa    2,74241%
Dosezeni Donos za dan 16 za delnico  LKPG na dan: 2003-03-11 bi znasal: -1,09955% Skupni donos pa   1,64286%
Dosezeni Donos za dan 17 za delnico  LKPG na dan: 2003-03-12 bi znasal: 1,09622% Skupni donos pa    2,73908%
Dosezeni Donos za dan 18 za delnico  GRVG na dan: 2003-03-13 bi znasal: 2,55009% Skupni donos pa    5,28917%
Dosezeni Donos za dan 19 za delnico  ETOG na dan: 2003-03-14 bi znasal: -0,05003% Skupni donos pa   5,23914%
Dosezeni Donos za dan 20 za delnico  TCRG na dan: 2003-03-17 bi znasal: -1,26909% Skupni donos pa   3,97005%
Dosezeni Donos za dan 21 za delnico  ETOG na dan: 2003-03-18 bi znasal: 0,21558% Skupni donos pa    4,18563%
Dosezeni Donos za dan 22 za delnico  LKPG na dan: 2003-03-19 bi znasal: -0,37217% Skupni donos pa   3,81346%
Dosezeni Donos za dan 23 za delnico  LKPG na dan: 2003-03-20 bi znasal: 1,59158% Skupni donos pa    5,40504%
Dosezeni Donos za dan 24 za delnico  TCRG na dan: 2003-03-21 bi znasal: 1,33343% Skupni donos pa    6,73847%
Dosezeni Donos za dan 25 za delnico  LKPG na dan: 2003-03-24 bi znasal: 0,1994% Skupni donos pa     6,93787%
Dosezeni Donos za dan 26 za delnico  LKPG na dan: 2003-03-25 bi znasal: 0,14925% Skupni donos pa    7,08712%
Dosezeni Donos za dan 27 za delnico  GRVG na dan: 2003-03-26 bi znasal: 1,46065% Skupni donos pa    8,54777%
Dosezeni Donos za dan 28 za delnico  GRVG na dan: 2003-03-27 bi znasal: -1,68041% Skupni donos pa   6,86735%
Dosezeni Donos za dan 29 za delnico  ETOG na dan: 2003-03-28 bi znasal: -0,37967% Skupni donos pa   6,48768%
Dosezeni Donos za dan 30 za delnico  TCRG na dan: 2003-03-31 bi znasal: 0,19636% Skupni donos pa    6,68404%
```

```
Dosezeni Donos za dan 31 za delnico  LKPG na dan: 2003-04-01 bi znasal: 0,35004% Skupni donos pa    7,03408%
Dosezeni Donos za dan 32 za delnico  ETOG na dan: 2003-04-02 bi znasal: 1,59765% Skupni donos pa    8,63173%
Dosezeni Donos za dan 33 za delnico  GRVG na dan: 2003-04-03 bi znasal: -0,09473% Skupni donos pa   8,537%
Dosezeni Donos za dan 34 za delnico  ETOG na dan: 2003-04-04 bi znasal: 3,21716% Skupni donos pa    11,75416%
Dosezeni Donos za dan 35 za delnico  LKPG na dan: 2003-04-07 bi znasal: -0,70854% Skupni donos pa   11,04562%
Dosezeni Donos za dan 36 za delnico  TCRG na dan: 2003-04-08 bi znasal: 0% Skupni donos pa    11,04562%
Dosezeni Donos za dan 37 za delnico  TCRG na dan: 2003-04-09 bi znasal: -1,75439% Skupni donos pa   9,29123%
Dosezeni Donos za dan 38 za delnico  TCRG na dan: 2003-04-10 bi znasal: -0,71429% Skupni donos pa   8,57695%
Dosezeni Donos za dan 39 za delnico  LKPG na dan: 2003-04-11 bi znasal: 0,57931% Skupni donos pa    9,15626%
Dosezeni Donos za dan 40 za delnico  ETOG na dan: 2003-04-14 bi znasal: 1,42175% Skupni donos pa    10,57801%
Dosezeni Donos za dan 41 za delnico  KRKG na dan: 2003-04-15 bi znasal: 0,07863% Skupni donos pa    10,65663%
Dosezeni Donos za dan 42 za delnico  TCRG na dan: 2003-04-16 bi znasal: 0% Skupni donos pa    10,65663%
Dosezeni Donos za dan 43 za delnico  ETOG na dan: 2003-04-17 bi znasal: -1,28205% Skupni donos pa   9,37458%
Dosezeni Donos za dan 44 za delnico  TCRG na dan: 2003-04-18 bi znasal: -0,00047% Skupni donos pa   9,37412%
Dosezeni Donos za dan 45 za delnico  LKPG na dan: 2003-04-22 bi znasal: -0,11179% Skupni donos pa   9,26232%
Dosezeni Donos za dan 46 za delnico  LKPG na dan: 2003-04-23 bi znasal: -0,04129% Skupni donos pa   9,22103%
Dosezeni Donos za dan 47 za delnico  LKPG na dan: 2003-04-24 bi znasal: 0,07957% Skupni donos pa    9,3006%
Dosezeni Donos za dan 48 za delnico  LKPG na dan: 2003-04-25 bi znasal: -0,05974% Skupni donos pa   9,24086%
Dosezeni Donos za dan 49 za delnico  GRVG na dan: 2003-04-28 bi znasal: 0,14494% Skupni donos pa    9,38581%
Dosezeni Donos za dan 50 za delnico  GRVG na dan: 2003-04-29 bi znasal: 0,65335% Skupni donos pa    10,03916%
Dosezeni Donos za dan 51 za delnico  LKPG na dan: 2003-04-30 bi znasal: 0,95125% Skupni donos pa    10,99041%
Dosezeni Donos za dan 52 za delnico  LKPG na dan: 2003-05-05 bi znasal: 0,46676% Skupni donos pa    11,45717%
Dosezeni Donos za dan 53 za delnico  LKPG na dan: 2003-05-06 bi znasal: 1,03581% Skupni donos pa    12,49298%
Dosezeni Donos za dan 54 za delnico  LKPG na dan: 2003-05-07 bi znasal: 0,95484% Skupni donos pa    13,44782%
Dosezeni Donos za dan 55 za delnico  LKPG na dan: 2003-05-08 bi znasal: 0,42293% Skupni donos pa    13,87075%
Dosezeni Donos za dan 56 za delnico  TCRG na dan: 2003-05-09 bi znasal: -0,26786% Skupni donos pa   13,60289%
Dosezeni Donos za dan 57 za delnico  LKPG na dan: 2003-05-12 bi znasal: -0,41459% Skupni donos pa   13,1883%
Dosezeni Donos za dan 58 za delnico  LKPG na dan: 2003-05-13 bi znasal: 0,39306% Skupni donos pa    13,58136%
Dosezeni Donos za dan 59 za delnico  LKPG na dan: 2003-05-14 bi znasal: -0,02734% Skupni donos pa   13,55402%
Dosezeni Donos za dan 60 za delnico  TCRG na dan: 2003-05-15 bi znasal: -0,49643% Skupni donos pa   13,05759%
Dosezeni Donos za dan 61 za delnico  ETOG na dan: 2003-05-16 bi znasal: -1,06985% Skupni donos pa   11,98774%
Dosezeni Donos za dan 62 za delnico  ETOG na dan: 2003-05-19 bi znasal: -1,26329% Skupni donos pa   10,72445%
Dosezeni Donos za dan 63 za delnico  LKPG na dan: 2003-05-20 bi znasal: 0,16905% Skupni donos pa    10,8935%
Dosezeni Donos za dan 64 za delnico  LKPG na dan: 2003-05-21 bi znasal: -1,43303% Skupni donos pa   9,46047%
Dosezeni Donos za dan 65 za delnico  LKPG na dan: 2003-05-22 bi znasal: 0,17333% Skupni donos pa    9,6338%
Dosezeni Donos za dan 66 za delnico  LKPG na dan: 2003-05-23 bi znasal: -0,77864% Skupni donos pa   8,85517%
Dosezeni Donos za dan 67 za delnico  ETOG na dan: 2003-05-26 bi znasal: 0% Skupni donos pa    8,85517%
Dosezeni Donos za dan 68 za delnico  LKPG na dan: 2003-05-27 bi znasal: 0,93868% Skupni donos pa    9,79384%
```

```
Dosezeni Donos za dan 69 za delnico  LKPG na dan: 2003-05-28 bi znasal: -0,15046% Skupni donos pa   9,64338%
Dosezeni Donos za dan 70 za delnico  TCRG na dan: 2003-05-29 bi znasal: -0,65846% Skupni donos pa   8,98492%
Dosezeni Donos za dan 71 za delnico  TCRG na dan: 2003-05-30 bi znasal: -0,07735% Skupni donos pa   8,90758%
Dosezeni Donos za dan 72 za delnico  TCRG na dan: 2003-06-02 bi znasal: -0,6173% Skupni donos pa    8,29028%
Dosezeni Donos za dan 73 za delnico  ETOG na dan: 2003-06-03 bi znasal: 0% Skupni donos pa    8,29028%
Dosezeni Donos za dan 74 za delnico  TCRG na dan: 2003-06-04 bi znasal: -0,21418% Skupni donos pa   8,0761%
Dosezeni Donos za dan 75 za delnico  TCRG na dan: 2003-06-05 bi znasal: -0,00128% Skupni donos pa   8,07482%
Dosezeni Donos za dan 76 za delnico  TCRG na dan: 2003-06-06 bi znasal: 0% Skupni donos pa    8,07482%
Dosezeni Donos za dan 77 za delnico  TCRG na dan: 2003-06-09 bi znasal: 0% Skupni donos pa    8,07482%
Dosezeni Donos za dan 78 za delnico  TCRG na dan: 2003-06-10 bi znasal: 0,01177% Skupni donos pa    8,08659%
Dosezeni Donos za dan 79 za delnico  LKPG na dan: 2003-06-11 bi znasal: 0,0334% Skupni donos pa     8,11999%
Dosezeni Donos za dan 80 za delnico  TCRG na dan: 2003-06-12 bi znasal: 1,29516% Skupni donos pa    9,41516%
Dosezeni Donos za dan 81 za delnico  ETOG na dan: 2003-06-13 bi znasal: 0% Skupni donos pa    9,41516%
Dosezeni Donos za dan 82 za delnico  ETOG na dan: 2003-06-16 bi znasal: 0% Skupni donos pa    9,41516%
Dosezeni Donos za dan 83 za delnico  ETOG na dan: 2003-06-17 bi znasal: 0% Skupni donos pa    9,41516%
Dosezeni Donos za dan 84 za delnico  LKPG na dan: 2003-06-18 bi znasal: -0,10054% Skupni donos pa   9,31462%
Dosezeni Donos za dan 85 za delnico  ETOG na dan: 2003-06-19 bi znasal: 0% Skupni donos pa    9,31462%
Dosezeni Donos za dan 86 za delnico  LKPG na dan: 2003-06-20 bi znasal: 0,20046% Skupni donos pa    9,51508%
Dosezeni Donos za dan 87 za delnico  ETOG na dan: 2003-06-23 bi znasal: 0% Skupni donos pa    9,51508%
Dosezeni Donos za dan 88 za delnico  ETOG na dan: 2003-06-24 bi znasal: -0,00052% Skupni donos pa   9,51455%
Dosezeni Donos za dan 89 za delnico  ETOG na dan: 2003-06-26 bi znasal: -0,52356% Skupni donos pa   8,99099%
Dosezeni Donos za dan 90 za delnico  LKPG na dan: 2003-06-27 bi znasal: -0,54421% Skupni donos pa   8,44679%
Dosezeni Donos za dan 91 za delnico  LKPG na dan: 2003-06-30 bi znasal: -3,34831% Skupni donos pa   5,09848%
Dosezeni Donos za dan 92 za delnico  ETOG na dan: 2003-07-01 bi znasal: 0% Skupni donos pa    5,09848%
Dosezeni Donos za dan 93 za delnico  LKPG na dan: 2003-07-02 bi znasal: -0,83609% Skupni donos pa   4,26239%
Dosezeni Donos za dan 94 za delnico  LKPG na dan: 2003-07-03 bi znasal: 0,27644% Skupni donos pa    4,53883%
Dosezeni Donos za dan 95 za delnico  LKPG na dan: 2003-07-04 bi znasal: 0,3358% Skupni donos pa     4,87463%
Dosezeni Donos za dan 96 za delnico  LKPG na dan: 2003-07-07 bi znasal: 0,21048% Skupni donos pa    5,08512%
Dosezeni Donos za dan 97 za delnico  GRVG na dan: 2003-07-08 bi znasal: 0,31579% Skupni donos pa    5,4009%
Dosezeni Donos za dan 98 za delnico  TCRG na dan: 2003-07-09 bi znasal: -1,89283% Skupni donos pa   3,50807%
Dosezeni Donos za dan 99 za delnico  LKPG na dan: 2003-07-10 bi znasal: 0,44601% Skupni donos pa    3,95409%
Dosezeni Donos za dan 100 za delnico  LKPG na dan: 2003-07-11 bi znasal: -0,65076% Skupni donos pa  3,30332%
Dosezeni Donos za dan 101 za delnico  TCRG na dan: 2003-07-14 bi znasal: 0% Skupni donos pa   3,30332%
Dosezeni Donos za dan 102 za delnico  ETOG na dan: 2003-07-15 bi znasal: -1,02564% Skupni donos pa  2,27768%
Dosezeni Donos za dan 103 za delnico  LKPG na dan: 2003-07-16 bi znasal: 0,35516% Skupni donos pa   2,63284%
Dosezeni Donos za dan 104 za delnico  LKPG na dan: 2003-07-17 bi znasal: 0,75536% Skupni donos pa   3,3882%
Dosezeni Donos za dan 105 za delnico  LKPG na dan: 2003-07-18 bi znasal: -0,53416% Skupni donos pa  2,85404%
Dosezeni Donos za dan 106 za delnico  LKPG na dan: 2003-07-21 bi znasal: 1,42436% Skupni donos pa   4,2784%
```

```
Dosezeni Donos za dan 107 za delnico  GRVG na dan: 2003-07-22 bi znasal: -0,61294% Skupni donos pa  3,66545%
Dosezeni Donos za dan 108 za delnico  LKPG na dan: 2003-07-23 bi znasal: 0,56526% Skupni donos pa   4,23072%
Dosezeni Donos za dan 109 za delnico  LKPG na dan: 2003-07-24 bi znasal: 0,34806% Skupni donos pa   4,57878%
Dosezeni Donos za dan 110 za delnico  LKPG na dan: 2003-07-25 bi znasal: 0,58642% Skupni donos pa   5,16519%
Dosezeni Donos za dan 111 za delnico  LKPG na dan: 2003-07-28 bi znasal: 0,10323% Skupni donos pa   5,26843%
Dosezeni Donos za dan 112 za delnico  GRVG na dan: 2003-07-29 bi znasal: 1,37762% Skupni donos pa   6,64605%
Dosezeni Donos za dan 113 za delnico  LKPG na dan: 2003-07-30 bi znasal: 1,23955% Skupni donos pa   7,88561%
Dosezeni Donos za dan 114 za delnico  ETOG na dan: 2003-07-31 bi znasal: 0% Skupni donos pa   7,88561%
Dosezeni Donos za dan 115 za delnico  ETOG na dan: 2003-08-01 bi znasal: 0% Skupni donos pa   7,88561%
Dosezeni Donos za dan 116 za delnico  GRVG na dan: 2003-08-04 bi znasal: 0,53923% Skupni donos pa   8,42484%
Dosezeni Donos za dan 117 za delnico  LKPG na dan: 2003-08-05 bi znasal: -0,35026% Skupni donos pa  8,07457%
Dosezeni Donos za dan 118 za delnico  LKPG na dan: 2003-08-06 bi znasal: 1,07968% Skupni donos pa   9,15425%
Dosezeni Donos za dan 119 za delnico  LKPG na dan: 2003-08-07 bi znasal: 0,12586% Skupni donos pa   9,28011%
Dosezeni Donos za dan 120 za delnico  LKPG na dan: 2003-08-08 bi znasal: -0,12141% Skupni donos pa  9,1587%
Dosezeni Donos za dan 121 za delnico  ETOG na dan: 2003-08-11 bi znasal: 0% Skupni donos pa   9,1587%
Dosezeni Donos za dan 122 za delnico  LKPG na dan: 2003-08-12 bi znasal: -0,1925% Skupni donos pa   8,96619%
Dosezeni Donos za dan 123 za delnico  LKPG na dan: 2003-08-13 bi znasal: 0,02193% Skupni donos pa   8,98812%
Dosezeni Donos za dan 124 za delnico  LKPG na dan: 2003-08-14 bi znasal: 0,28771% Skupni donos pa   9,27583%
Dosezeni Donos za dan 125 za delnico  LKPG na dan: 2003-08-18 bi znasal: 0,36719% Skupni donos pa   9,64302%
Dosezeni Donos za dan 126 za delnico  LKPG na dan: 2003-08-19 bi znasal: 1,69608% Skupni donos pa   11,3391%
Dosezeni Donos za dan 127 za delnico  GRVG na dan: 2003-08-20 bi znasal: 2,30086% Skupni donos pa   13,63996%
Dosezeni Donos za dan 128 za delnico  LKPG na dan: 2003-08-21 bi znasal: 0,09449% Skupni donos pa   13,73445%
Dosezeni Donos za dan 129 za delnico  ETOG na dan: 2003-08-22 bi znasal: 3,50096% Skupni donos pa   17,23541%
Dosezeni Donos za dan 130 za delnico  LKPG na dan: 2003-08-25 bi znasal: 0,311% Skupni donos pa     17,54641%
Dosezeni Donos za dan 131 za delnico  LKPG na dan: 2003-08-26 bi znasal: 0,65317% Skupni donos pa   18,19958%
Dosezeni Donos za dan 132 za delnico  GRVG na dan: 2003-08-27 bi znasal: 1,63316% Skupni donos pa   19,83274%
Dosezeni Donos za dan 133 za delnico  LKPG na dan: 2003-08-28 bi znasal: 0,49848% Skupni donos pa   20,33122%
Dosezeni Donos za dan 134 za delnico  GRVG na dan: 2003-08-29 bi znasal: -0,10385% Skupni donos pa  20,22737%
Dosezeni Donos za dan 135 za delnico  GRVG na dan: 2003-09-01 bi znasal: 1,89887% Skupni donos pa   22,12624%
Dosezeni Donos za dan 136 za delnico  ETOG na dan: 2003-09-02 bi znasal: 0% Skupni donos pa   22,12624%
Dosezeni Donos za dan 137 za delnico  LKPG na dan: 2003-09-03 bi znasal: -0,1508% Skupni donos pa   21,97544%
Dosezeni Donos za dan 138 za delnico  LKPG na dan: 2003-09-04 bi znasal: 0,17048% Skupni donos pa   22,14593%
Dosezeni Donos za dan 139 za delnico  ETOG na dan: 2003-09-05 bi znasal: -1,53681% Skupni donos pa  20,60912%
Dosezeni Donos za dan 140 za delnico  ETOG na dan: 2003-09-08 bi znasal: 0% Skupni donos pa   20,60912%
Dosezeni Donos za dan 141 za delnico  TCRG na dan: 2003-09-09 bi znasal: 0% Skupni donos pa   20,60912%
Dosezeni Donos za dan 142 za delnico  LKPG na dan: 2003-09-10 bi znasal: 1,11536% Skupni donos pa   21,72448%
Dosezeni Donos za dan 143 za delnico  ETOG na dan: 2003-09-11 bi znasal: -1,17647% Skupni donos pa  20,54801%
Dosezeni Donos za dan 144 za delnico  LKPG na dan: 2003-09-12 bi znasal: 0,36827% Skupni donos pa   20,91629%
```

```
Dosezeni Donos za dan 145 za delnico  ETOG na dan: 2003-09-15 bi znasal: 1,19048% Skupni donos pa   22,10676%
Dosezeni Donos za dan 146 za delnico  ETOG na dan: 2003-09-16 bi znasal: 0% Skupni donos pa   22,10676%
Dosezeni Donos za dan 147 za delnico  LKPG na dan: 2003-09-17 bi znasal: 0,7511% Skupni donos pa    22,85787%
Dosezeni Donos za dan 148 za delnico  LKPG na dan: 2003-09-18 bi znasal: 0,65978% Skupni donos pa   23,51765%
Dosezeni Donos za dan 149 za delnico  LKPG na dan: 2003-09-19 bi znasal: 1,46008% Skupni donos pa   24,97773%
Dosezeni Donos za dan 150 za delnico  LKPG na dan: 2003-09-22 bi znasal: -0,42161% Skupni donos pa  24,55611%
Dosezeni Donos za dan 151 za delnico  GRVG na dan: 2003-09-23 bi znasal: 0,90339% Skupni donos pa   25,4595%
Skupni donos za obdobje 151 dni bi znasal:  25,4595%
```

## Priloga F

Programska koda računalniškega programa Napovedovalec.

```
// package myprojects.napovedovalec;
/*
        Predstavitev razredov oziroma skupin razredov v paketih javanskemu sistemu

-       java in javax so del izvirne osnovne knjižnice javanskega okolja, ki jih potrebujemo za
        osnovne operacije jezika Java in grafične elemente, kot so tabele, okvirji, ...

-       weka knjižnica predstavlje razrede ki predstavljajo predvsem algoritme strojnega učenja

-       jfree knjižnica pa je namenjena oblikovanju grafov ter podpira kreiranje časovnih vrst,
        drsečih povprečij itd...
*/

import java.util.*;
import java.net.*;
import java.lang.*;
import java.text.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.io.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.*;
import javax.swing.text.JTextComponent.*;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.JScrollPane;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import javax.swing.JOptionPane;
import weka.core.*;
import weka.core.Instances;
import weka.classifiers.trees.j48.J48;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.neural.NeuralNetwork;
import weka.classifiers.lazy.IBk;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.time.TimeSeriesDataItem;
import org.jfree.data.time.Day;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.ChartPanel;
import org.jfree.data.MovingAverage;

/*
Osrednji razred Napovedovalec, ki je v bistvu podrazred razreda JPanel, ki
predstavlja grafično ploskev, ki jo v okviru začetne funkcije main()
vstavimo in prikažemo v okvirju Frame
*/
public class Napovedovalec extends JPanel {
    public static Instances centralneInstance;      //Osrednje instance, katere obdelujemo
    JTabbedPane jtb = new JTabbedPane();            //listi z zavihki
    JTextArea jta = new JTextArea("", 5, 40);       //prostor za poročila
    JScrollPane jsp = new JScrollPane(jta);         //sami dodamo drsnik
    JMenuBar jmb = new JMenuBar();                  //Akcije lahko izbiramo s pomočjo menijev
    JMenu file = new JMenu("File");
    JMenu edit = new JMenu("Edit");
    JMenu analizaJM = new JMenu("Analiza");
    JMenuItem jmi;
    JToolBar toolBar = new JToolBar();              //Ali pa s pomočjo gumbkov iz orodjarne
    Clipboard clipbd = getToolkit().getSystemClipboard();// Omogočimo tudi odložišče
    Reader in;                                      //z objektom reader preberemo podatke z diska
    InstanceHolder holder = new InstanceHolder(); // Služi kot začasno odlagališče objektov


    public void setInstances(Instances prInstance) { //Prenesemo instance v Napovedovalec
```

```
        centralneInstance = prInstance;
    }

//Metoda s katero potegnemo instance iz Napovedovalca
    public Instances getInstances() {
        return centralneInstance;
    }
//Konstrukcijska metoda s katero kreiramo Napovedovalca
    public Napovedovalec() {

//Razred, ki čaka akcijske ukaze in se odzove na ukaz menija za brisanje tekstovnega prostora.
        class newL implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                jta.setDocument(new PlainDocument());   //Predstavimo prazen dokument
            }
        }

                            //Razred, ki čaka na ukaz za odprtje novega dokumenta
        class openL implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                Frame frame = new Frame();
                FileDialog d = new FileDialog(frame, "Open File(s)");
                d.setFile("*.*");
                d.setDirectory(".");
                d.show();
                String file = d.getFile();

                if (file == null) {
                    return;
                }
                String directory = d.getDirectory();
                File f = new File(directory, file);

                if (f.exists()) {
                    try {
                        Reader in = new FileReader(f);

                        char[] buff = new char[1000000];
  int nch;

                        while ((nch = in.read(buff, 0, buff.length)) != -1) {
                            jta.setDocument(new PlainDocument());
                            jta.append(new String(buff, 0, nch));                       }
                    } catch (IOException io) {
                        System.err.println("IOException: " + io.getMessage());
                    }
                } else {
                    System.err.println("No such file: " + f); // javimo napako
                }
            }
        }

                //razred ki čaka na ukaz za shranjevanje datoteke, da lahko poročilo tudi shranimo
        class saveL implements ActionListener {
            public void actionPerformed(ActionEvent e) {
                Frame frame = new Frame();
                FileDialog d = new FileDialog(frame, "", FileDialog.SAVE);

                d.setFile("*.*");
                d.setDirectory(".");
                d.show();

                String savefile = d.getFile();

                if (savefile == null) {
                    return;
                } else {
                    String directory = d.getDirectory();
                    File f = new File(directory, savefile);
                    String docToSave = jta.getText();

                    if (docToSave != null) {
                        FileOutputStream fstrm = null;
                        BufferedOutputStream ostrm = null;
                        try {
                            fstrm = new FileOutputStream(f);
                            ostrm = new BufferedOutputStream(fstrm);
```

```
                    byte[] bytes = null;

                    try {
                        bytes = docToSave.getBytes();
                    } catch (Exception e1) {
                        e1.printStackTrace();
                    }
                    ostrm.write(bytes);
                } catch (IOException io) {
                  System.err.println("IOException: " + io.getMessage());
                }
                finally {
                    try {
                        ostrm.flush();
                        fstrm.close();
                        ostrm.close();
                    } catch (IOException ioe) {
                        System.err.println(
                                "IOException: " + ioe.getMessage());
                    }
                }
            }
        }
    }


    class exitL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            System.exit(-1);
        }
    }

// Kopiranje, rezanje in leplenje dela besedil po tekstualnem polju
    class copyL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String selection = jta.getSelectedText();
            StringSelection clipString = new StringSelection(selection);

            clipbd.setContents(clipString, clipString);
        }
    }


    class cutL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String selection = jta.getSelectedText();
            StringSelection clipString = new StringSelection(selection);

            clipbd.setContents(clipString, clipString);
            jta.replaceRange("", jta.getSelectionStart(),
                    jta.getSelectionEnd());
        }
    }


    class pasteL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Transferable clipData = clipbd.getContents(Napovedovalec.this);

            try {
                String clipString = (String) clipData.getTransferData(
                        DataFlavor.stringFlavor);

                jta.replaceRange(clipString, jta.getSelectionStart(),
                        jta.getSelectionEnd());
            } catch (Exception ex) {}
        }
    }


    class jtaL implements ActionListener {
        public void actionPerformed(ActionEvent e) {}
    }

//Razred s klaterim izvedemo vnos podatkov iz datoteke v sam program.
//Vnos je izveden preko FileReader razreda ki dostavi ARFF datoteko Wekinemu razredu Instances
```

```
//ta pa oblikuje instance(primerke) podatkov. Iz tako prebranih podatkov izdelamo časovne vrste
//(Time series), katere potem prikažemo v grafih JFreeChart.

        class Import extends Thread {
            String[] grafi;
            String graf;
            J48[] classifiers;
            NeuralNetwork[] classifiersN;
            IBk[] classifiersK;
            public Utils orodjarna;
            public Instance instancaZaVred;
            public Instance instancaZaVredNext;
            public Instances vneseneInstance;
            public Instances vneseneInstanceZDatum;
            public Instances instancesZavrednotit;
            public Instances instancesZaTrenirat;
            public Instances[] poljeRjev;
            public Instances[] poljeAjev;
            public Evaluation[] evaluation;
            public Evaluation[] evaluationN;
            public Evaluation[] evaluationK;
            public double[] classificiranaInstanca;
            private double[] distributionForInstN[];
            private double[] distributionForInst[];
            private double[] distributionForInstK[];
            private double[] VelikostDrevesa;
            private String[] imenaDelnic;
            private double[] ZanesljivostZadetka;
            private double[] PondKriterij;
            private int[] indexDelnice;
            private int stDelnice;
            private double delDonos;
            private double deljeniDelDonos;
            private double skupniDonos;
            private String alg;
            private String algStr;
            private String dneviStr;
            private int stDni;
            private int stSkupkovVPolju;
            private int steviloInstanc;
            private int stAttR;
            private double polovicka = 0.5;
            private JLabel slikaLabela;

            public Instances instance;
            public TimeSeries timeSeries;
            public TimeSeries movingAverageSeriesPos;
            public TimeSeriesCollection timeSeriesCollection;
            public Day day;
            public JFreeChart jfreeChart;
            public ChartPanel slika;
            public ChartPanel slikaPos;
            public MovingAverage movingAverage;
            public JPanel slike;
            public TimeSeries[] drsecaPovprecjaDelnic;
            public TimeSeries[] casovneVrsteDelnic;

            public TimeSeries timeSeries50;
            public TimeSeries movingAverageSeriesPos50;
            public TimeSeriesCollection timeSeriesCollection50;

            public JFreeChart jfreeChart50;
            public ChartPanel slikaPos50;
            public MovingAverage movingAverage50;
            public JPanel slike50;

            public void run() {

                System.out.println("run metoda iz ImportL ");
                Frame frame = new Frame();
                FileDialog d = new FileDialog(frame, "Open File(s)");
                d.setFile("*.*");
                d.setDirectory(".");
                d.show();
                String file = d.getFile();

                if (file == null) {
```

```
    return;
}
String directory = d.getDirectory();
File f = new File(directory, file);

if (f.exists()) {
    try {
        Reader in = new FileReader(f);

        vneseneInstanceZDatum = new Instances(in);
        vneseneInstance = new Instances(vneseneInstanceZDatum);
        holder.vpisiInstance(vneseneInstanceZDatum);

        JPanel slike = new JPanel();
        slike.setLayout(new BoxLayout(slike, BoxLayout.Y_AXIS));
        JPanel slike50 = new JPanel();
        slike50.setLayout(new BoxLayout(slike50, BoxLayout.Y_AXIS));
        TimeSeriesCollection timeSeriesCollection =new TimeSeriesCollection();
        TimeSeriesCollection timeSeriesCollection50 = new TimeSeriesCollection();
        drsecaPovprecjaDelnic = new TimeSeries[vneseneInstanceZDatum.numAttributes()];
        casovneVrsteDelnic =new TimeSeries[vneseneInstanceZDatum.numAttributes()];

        for (int b = 1; b< vneseneInstanceZDatum.numAttributes(); b++) {
            TimeSeriesCollection timeSeriesCollectionPos =
            new TimeSeriesCollection();
            TimeSeriesCollection timeSeriesCollectionPos50 =new TimeSeriesCollection();
            timeSeries = new TimeSeries(vneseneInstanceZDatum.attribute(b).name());
            timeSeries50 = new TimeSeries(vneseneInstanceZDatum.attribute(b).name()+ " 50 dni");
            movingAverage = new MovingAverage();
            movingAverage50 = new MovingAverage();

            for (int s = 0; s< vneseneInstanceZDatum.numInstances(); s++){
                String datumString = new String();
                datumString =vneseneInstanceZDatum.instance(s).toString(0);
                Day dan = Day.parseDay(datumString);
                TimeSeriesDataItem timeSeriesDataItem =new TimeSeriesDataItem(dan,
                vneseneInstanceZDatum.instance(s).value(b));
                timeSeries.add(timeSeriesDataItem);
            }

            timeSeries50 = timeSeries.createCopy(timeSeries.getItemCount() - 50,timeSeries.getItemCount() - 1);
            timeSeriesCollectionPos50.addSeries(timeSeries50);
            timeSeriesCollection.addSeries(timeSeries);
            timeSeriesCollectionPos.addSeries(timeSeries);
            movingAverageSeriesPos =movingAverage.createMovingAverage(timeSeries, "drsece povprecje", 10, 0);
            timeSeriesCollectionPos.addSeries(movingAverageSeriesPos);
            casovneVrsteDelnic[b] = new TimeSeries(vneseneInstanceZDatum.attribute(b).toString());
            casovneVrsteDelnic[b] = timeSeries;
            drsecaPovprecjaDelnic[b] = new TimeSeries(vneseneInstanceZDatum.attribute(b).toString());
            drsecaPovprecjaDelnic[b] = movingAverageSeriesPos;
            holder.vpisiDrsecaPovprecjaDelnic(drsecaPovprecjaDelnic);
            holder.vpisiCasovneVrsteDelnic(casovneVrsteDelnic);

            JFreeChart jFreeChartPos =ChartFactory.createTimeSeriesChart(
                        vneseneInstanceZDatum.attribute(b).name(),
                        "cas", "vrednost", timeSeriesCollectionPos,
                        true,
                        false, false);

            ChartPanel  slikaPos = new ChartPanel(jFreeChartPos);
            slike.add(slikaPos);
            movingAverageSeriesPos50 =movingAverage.createMovingAverage(
            timeSeries50, "drsece povprecje 50 dni", 5,0);
            timeSeriesCollectionPos50.addSeries(movingAverageSeriesPos50);
            JFreeChart jFreeChartPos50 =ChartFactory.createTimeSeriesChart(
                vneseneInstanceZDatum.attribute(b).name(),
                "cas", "vrednost", timeSeriesCollectionPos50,
                true, // show legend
                false, false);

            ChartPanel  slikaPos50 = new ChartPanel(jFreeChartPos50);
            slike50.add(slikaPos50);

        }

        JFreeChart jFreeChart = ChartFactory.createTimeSeriesChart(
            "Vse Skupaj", "cas", "vrednost",
```

```
                    timeSeriesCollection, true, // show legend
                    false, false);

            ChartPanel  slika = new ChartPanel(jFreeChart);
            slike.add(slika);
            JScrollPane jspG = new JScrollPane(slike);
            jtb.addTab("Grafi", null, jspG, "Grafi podatkov");
            JScrollPane jspG50 = new JScrollPane(slike50);
            jtb.addTab("Grafi zadnjih 50 dni", null, jspG50,"Grafi podatkov 50");
            MyTableModel myModel = new MyTableModel();
            myModel.setirajInstance(vneseneInstanceZDatum);
            JTable jt = new JTable(myModel);
            jt.setRowSelectionAllowed(true);
            ExcelAdapter myAd = new ExcelAdapter(jt);
            JScrollPane jspT = new JScrollPane(jt);
            jspT.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
            jtb.addTab("Podatki", null, jspT, "Podatki za obdelavo");
            jtb.setSelectedComponent(jspT);

          } catch (Exception io) {System.err.println("Exception: " + io.getMessage());
          }
        } else {
            System.err.println("No such file: " + f);
        }

        // this.stop();
      }
    }


// Objekt na podlagi tega razreda "Analiza" je osrednji objekt, kjer se izvaja strojno učenje
// in se zgradijo na tej podlagi učni modeli. Odločitveno drevo, nevronska mreža, k-najbližjih
// sosedov
//
    class Analiza extends Thread {
      String[] grafi;
      String graf;
      J48[] classifiers;
      NeuralNetwork[] classifiersN;
      IBk[] classifiersK;
      public Utils orodjarna;
      public Instance instancaZaVred;
      public Instance instancaZaVredNext;
      public Instance instancaZaVredDatum;
      public Instance instancaZaVredNextDatum;
      public Instances vneseneInstance;
      public Instances vneseneInstanceZDatum;
      public Instances vneseneInstanceZaDatume;
      public Instances instancesZavrednotit;
      public Instances instancesZaTrenirat;
      public Instances instancesZavrednotitDatum;
      public Instances instancesZaTreniratDatum;

      public Instances[] poljeRjev;
      public Instances[] poljeAjev;
      public Evaluation[] evaluation;
      public Evaluation[] evaluationN;
      public Evaluation[] evaluationK;
      public double[] classificiranaInstanca;
      private double[] distributionForInstN[];
      private double[] distributionForInst[];
      private double[] distributionForInstK[];
      private double[] VelikostDrevesa;
      private String[] imenaDelnic;
      private double[] ZanesljivostZadetka;
      private double[] PondKriterij;
      private double PondKriterijPrever;
      private int[] indexDelnice;
      private int stDelnice;
      private double delDonos;
      private double deljeniDelDonos;
      private double skupniDonos;
      private String alg;
      private String algStr;
      private String dneviStr;
      private int stDni;
      private int stSkupkovVPolju;
```

```
private int steviloInstanc;
private int stAttR;
private double polovicka = 0.5;
private JLabel slikaLabela;

public Instances instance;
public TimeSeries timeSeries;
public TimeSeriesCollection timeSeriesCollection;
public Day day;
public JFreeChart jfreeChart;
public ChartPanel slika;
public ChartPanel slikaPos;
public JPanel slike;
public String AnalizaLString = new String("AnalizaLString");
public String datum;

public TimeSeries najboljsaCasovnaVrsta;
public TimeSeries najboljseDrsecePovprecje;
public TimeSeries[] vrstaDelAnaliza;
public TimeSeries[] movingAverageAnaliza;

public void run() {
   jta.setDocument(new PlainDocument());
   System.out.println("run metoda iz AnalizaL ");
   System.out.println(AnalizaLString + " iz run()"); {
      try {
         vneseneInstance = new Instances(holder.instance);
         vneseneInstanceZDatum = new Instances(holder.instance);
         vneseneInstanceZaDatume = new Instances(holder.instance);
         vneseneInstance.delete();
         vneseneInstance.deleteAttributeAt(0);
         vrstaDelAnaliza = new TimeSeries[vneseneInstance.numAttributes()];
         movingAverageAnaliza = new TimeSeries[vneseneInstance.numAttributes()];
         vrstaDelAnaliza = holder.casovneVrsteDelnic;
         movingAverageAnaliza = holder.drsecaPovprecjaDelnic;
         for (int s = 0; s < vneseneInstanceZDatum.numInstances(); s++) {
            Instance nInstance = new Instance(
            vneseneInstanceZDatum.numAttributes() - 1);
            nInstance.setDataset(vneseneInstance);
            vneseneInstance.add(nInstance);
            for (int b = 1; b < vneseneInstanceZDatum.numAttributes(); b++)                                    {
               vneseneInstance.instance(s).setValue(b - 1,
               vneseneInstanceZDatum.instance(s).value(b));
               }
         }

         Napovedovalec.centralneInstance = vneseneInstance;
         Object[] possibilities = {"Neuronska Mreza","J48 Drevo",  "K-sosedov"};
         String algStr = (String) JOptionPane.showInputDialog(null,
                              "Izberi vrsto algoritma " + "za strojno  učenje:",
                              "Vnos Algoritma", JOptionPane.PLAIN_MESSAGE,
                              null, possibilities, "J48 Drevo");

         if ((algStr != null) && (algStr.length() > 0)) {alg = algStr;}
         dneviStr = JOptionPane.showInputDialog( "Vnesite število dni za analizo:");
         stDni = Integer.parseInt(dneviStr);
         System.out.println(" Rezultati algoritma " + "{" + alg + "}"
         + " za " + stDni + "\n");
         delDonos = 0;
         deljeniDelDonos = 0;
         skupniDonos = 0;
         int deleztrain = vneseneInstance.numInstances()- (stDni + 1);
         int deleztest = vneseneInstance.numInstances()- deleztrain;
         instancesZaTrenirat = new Instances(vneseneInstance, 0,deleztrain);
         instancesZavrednotit = new Instances(vneseneInstance,deleztrain, deleztest);
         instancesZaTreniratDatum = new Instances(vneseneInstanceZaDatume, 0,  deleztrain);
         instancesZavrednotitDatum = new Instances(vneseneInstanceZaDatume, deleztrain, deleztest);
         NumberFormat nf2 = NumberFormat.getInstance();
         NumberFormat nfPR = NumberFormat.getPercentInstance();
         NumberFormat nfPR2 = NumberFormat.getPercentInstance();
         nf2.setMaximumFractionDigits(2);
         nfPR.setMaximumFractionDigits(2);
         nfPR2.setMaximumFractionDigits(5);

         for (int g = 0; g < deleztest - 1; g++) {
            instancaZaVred = instancesZavrednotit.instance(g);
            instancaZaVredNext = instancesZavrednotit.instance(g + 1);
```

```
                instancaZaVredDatum = instancesZavrednotitDatum.instance(g);
                instancaZaVredNextDatum = instancesZavrednotitDatum.instance(g + 1);
                datum = new String(instancaZaVredDatum.toString(0));
                UstvariPoljePrimerkov narejeniprimerki = new UstvariPoljePrimerkov(instancesZaTrenirat);
                poljeAjev = narejeniprimerki.vseinstance;
                steviloInstanc = narejeniprimerki.vseinstance[0].numInstances();
                stAttR = narejeniprimerki.vseinstance[0].numAttributes();
                stSkupkovVPolju = narejeniprimerki.vseinstance.length;
                orodjarna = new Utils();
                grafi = new String[narejeniprimerki.stAtt];
                imenaDelnic = new String[narejeniprimerki.stAtt];
                indexDelnice = new int[narejeniprimerki.stAtt];
                VelikostDrevesa = new double[narejeniprimerki.stAtt];
                ZanesljivostZadetka = new double[narejeniprimerki.stAtt];
                PondKriterij = new double[narejeniprimerki.stAtt];

                try {
                       for (int stTrenutniSkupek = 0;stTrenutniSkupek<
                       stSkupkovVPolju; tTrenutniSkupek++) { Instances trenutniSkupek =
                                                              narejeniprimerki.vseinstance[stTrenutniSkupek];

                      for (int stTrenutnaInstanca = 0; stTrenutnaInstanca< steviloInstanc; stTrenutnaInstanca++) {
                         Instance trenutnaInstanca = trenutniSkupek.instance(stTrenutnaInstanca);

                         for (int stAtributa = 0; stAtributa< stAttR; stAtributa++) {
                             double valueR = trenutnaInstanca.value(stAtributa);
                             double A;
                             double stevec;
                             double imenovalec;
                             double rezultat;
                             int stManjsih = 0;

                                for (int m = 0; m < stSkupkovVPolju; m++) {
                                        if (narejeniprimerki.vseinstance[m].instance(stTrenutnaInstanca).value(stAtributa)
                                        <= valueR) {stManjsih++;}
                                        else {
                                   stManjsih = stManjsih;
                                  }
                                }
                             stevec = stManjsih - 1;
                             imenovalec = stSkupkovVPolju - 1;
                             rezultat = stevec / imenovalec;
                             A = rezultat - polovicka;
                             poljeAjev[stTrenutniSkupek].instance(stTrenutnaInstanca).setValue(stAtributa, A);
                         }
                      }
                   }

                   Precistilec cistiPrimerki = new Precistilec(poljeAjev);
                   jtb.setSelectedComponent(jsp);

// ***************   Nevronska mreza ******************************

                   if (alg == "Neuronska Mreza") {
                      classifiersN = new NeuralNetwork[narejeniprimerki.stAtt];
                      evaluationN = new Evaluation[narejeniprimerki.stAtt];
                      distributionForInstN = new double[narejeniprimerki.stAtt][2];
                      for (int s = 0; s < narejeniprimerki.stAtt; s++) {
                         classifiersN[s] = new NeuralNetwork();
                         classifiersN[s].buildClassifier(cistiPrimerki.trainInstances[s]);
                         distributionForInstN[s] =classifiersN[s].distributionForInstance(cistiPrimerki.zadnjeInstance[s]);
                         evaluationN[s] = new Evaluation(cistiPrimerki.trainInstances[s]);
                         evaluationN[s].evaluateModel(classifiersN[s],cistiPrimerki.testInstances[s]);

                         if (distributionForInstN[s][1]> distributionForInstN[s][0]) {
                            ZanesljivostZadetka[s] = evaluationN[s].pctCorrect();
                            menaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
                            indexDelnice[s] = s;
                            PondKriterij[s] = ((distributionForInstN[s][1]* 0.7)+ (ZanesljivostZadetka[s]* 0.3));
                             PondKriterijPrever = PondKriterijPrever+ PondKriterij[s];

                         } else if (distributionForInstN[s][1]< distributionForInstN[s][0]) {
                            ZanesljivostZadetka[s] = 0;
                            PondKriterij[s] = 0;
                            imenaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
                            indexDelnice[s] = s;
                         }
```

```
        if (stDni <= 5) {
            System.out.println(evaluationN[s].toSummaryString());
            jta.append("======================================"+ "\n");
            jta.append("Tole je evaluacija z neuronsko mrezo za delnico "
            + cistiPrimerki.trainInstances[s].relationName()
            + ":" + "\n");
            jta.append(evaluationN[s].toSummaryString());
            jta.append("\n"
            + "porazdelitev verjetnosti za delnico "
            + cistiPrimerki.trainInstances[s].relationName()
            + " za podpovprecen znasa "
            + nfPR.format(distributionForInstN[s][0])+ "\n");
            jta.append("\n"
            + "porazdelitev verjetnosti za delnico "
            + cistiPrimerki.trainInstances[s].relationName()
            + " za nadpovprecen znasa "
            + nfPR.format(distributionForInstN[s][1])+ "\n");
        }
    }
}


// **********************  J48 Drevo **********************
else if (alg == "J48 Drevo") {
    classifiers = new J48[narejeniprimerki.stAtt];
    evaluation = new Evaluation[narejeniprimerki.stAtt];
    classificiranaInstanca = new double[narejeniprimerki.stAtt];
    distributionForInst = new double[narejeniprimerki.stAtt][2];

    for (int s = 0; s < narejeniprimerki.stAtt; s++) {
        classifiers[s] = new J48();
        classifiers[s].buildClassifier(cistiPrimerki.trainInstances[s]);
        classificiranaInstanca[s] = classifiers[s].classifyInstance(cistiPrimerki.zadnjeInstance[s]);
        distributionForInst[s] = classifiers[s].distributionForInstance(cistiPrimerki.zadnjeInstance[s]);
        evaluation[s] = new Evaluation(cistiPrimerki.trainInstances[s]);
        evaluation[s].evaluateModel(classifiers[s],cistiPrimerki.testInstances[s]);

        if (distributionForInst[s][1]> distributionForInst[s][0]) {
            ZanesljivostZadetka[s] = evaluation[s].pctCorrect();
            VelikostDrevesa[s] = classifiers[s].measureTreeSize();
            imenaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
            indexDelnice[s] = s;
            PondKriterij[s] = ((distributionForInst[s][1]* 0.8)+ (ZanesljivostZadetka[s]* 0.2)+
               (VelikostDrevesa[s]* 0.0));
            PondKriterijPrever = PondKriterijPrever+ PondKriterij[s];
        } else if (distributionForInst[s][1]< distributionForInst[s][0]) {
            VelikostDrevesa[s] = 0;
            ZanesljivostZadetka[s] = 0;
            PondKriterij[s] = 0;
            imenaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
            indexDelnice[s] = s;
            //PondKriterijPrever = PondKriterijPrever
             //       + PondKriterij[s];
        }

        if (stDni <= 5) {

            System.out.println(evaluation[s].toSummaryString());
            jta.append("================================================="+ "\n");
            jta.append("Tole je evaluacija z drevesom za delnico "
                        + cistiPrimerki.trainInstances[s].relationName()
                        + ":" + "\n");
            jta.append(evaluation[s].toSummaryString());
            jta.append("\n"
                        + " Tole je graf za delnico "
                        + cistiPrimerki.trainInstances[s].relationName()
                        + ":");
            grafi[s] = new String( classifiers[s].toString());
            jta.append(grafi[s]);
            jta.append(
                  "\n" + "Delnica "
                  + cistiPrimerki.trainInstances[s].relationName()
                  + " je klasificirana   "
                  + cistiPrimerki.trainInstances[s].classAttribute().value( (int) classificiranaInstanca[s])
                  + "\n");
            jta.append("\n"
```

```
                    + "porazdelitev verjetnosti za delnico "
                    + cistiPrimerki.trainInstances[s].relationName()
                    + " za podpovprecen znasa "
                    + nfPR.format(distributionForInst[s][0])
                    + "\n");
            jta.append("\n"
                    + "porazdelitev verjetnosti za delnico "
                    + cistiPrimerki.trainInstances[s].relationName()
                    + " za nadpovprecen znasa "
                    + nfPR.format(distributionForInst[s][1])
                    + "\n");
        }
    }
}

// ********************************  K-sosedov   ******************************
else if (alg == "K-sosedov") {
    classifiersK = new IBk[narejeniprimerki.stAtt];
    evaluationK = new Evaluation[narejeniprimerki.stAtt];
    distributionForInstK = new double[narejeniprimerki.stAtt][2];
    for (int s = 0; s < narejeniprimerki.stAtt; s++) {
        classifiersK[s] = new IBk();
        classifiersK[s].buildClassifier(
                cistiPrimerki.trainInstances[s]);
        distributionForInstK[s] = classifiersK[s].distributionForInstance(
                cistiPrimerki.zadnjeInstance[s]);
        evaluationK[s] = new Evaluation(
                cistiPrimerki.trainInstances[s]);
        evaluationK[s].evaluateModel(
                classifiersK[s],
                cistiPrimerki.testInstances[s]);

        if (distributionForInstK[s][1]
                > distributionForInstK[s][0]) {
            ZanesljivostZadetka[s] = evaluationK[s].pctCorrect();
            imenaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
            indexDelnice[s] = s;
            PondKriterij[s] = ((distributionForInstK[s][1]* 1.0)+ (ZanesljivostZadetka[s] * 0.0) );
             PondKriterijPrever = PondKriterijPrever+ PondKriterij[s];

        } else if (distributionForInstK[s][1]< distributionForInstK[s][0]) {
            ZanesljivostZadetka[s] = 0;
            PondKriterij[s] = 0;
            imenaDelnic[s] = cistiPrimerki.trainInstances[s].relationName();
            indexDelnice[s] = s;
        }

        if (stDni <= 5) {
            System.out.println(evaluationK[s].toSummaryString());
            jta.append("================================================"+ "\n");
            jta.append(
                    "Tole je evaluacija z K-najbližjih sosedov za delnico "
                     + cistiPrimerki.trainInstances[s].relationName()
                     + ":" + "\n");
            jta.append(evaluationK[s].toSummaryString());
            jta.append(
                      "\n"
                    + "porazdelitev verjetnosti za delnico "
                    + cistiPrimerki.trainInstances[s].relationName()
                    + " za podpovprecen znasa "
                    + nfPR.format(distributionForInstK[s][0])
                    + "\n");
            jta.append(
                      "\n"
                    + "porazdelitev verjetnosti za delnico "
                    + cistiPrimerki.trainInstances[s].relationName()
                    + " za nadpovprecen znasa "
                    + nfPR.format(
                          distributionForInstK[s][1])
                          + "\n");
        }
    }
}


if (stDni <= 5) {
    stDelnice = indexDelnice[orodjarna.maxIndex(PondKriterij)];
```

```
double vrednostNext = instancaZaVredNext.value(stDelnice);
double vrednost = instancaZaVred.value(stDelnice);
delDonos = vrednostNext - vrednost;
deljeniDelDonos = delDonos / vrednost;
skupniDonos = skupniDonos + deljeniDelDonos;
jta.append("\n\n\n\n\n"+"=====================================================");
jta.append( "\n"+ "  ");
jta.append( "\n"+ "  POVZETEK ANALIZE ");
jta.append("\n"+ "   ");
jta.append("\n"+ "=====================================================");
jta.append("\n\n\n\n\n");
System.out.println( "\n\n\n\n\n"+ "=====================================================");
System.out.println("\n"+ "/ POVZETEK ANALIZE                                    /");
System.out.println("\n"+ "=====================================================");
System.out.println("\n\n\n\n\n");

if (alg == "Neuronska Mreza") {
   System.out.println(
         "DELNICA" + "\t" + "INDEX"
         + "\t" + "NADPOVPRECEN" + "\t"
         + "\t" + "PODPOVPRECEN" + "\t"
         + "\t" + "ZANESLJIVOST ZADETKA"
         + "\t" + "\t"
         + "PONDER. KRITERIJ" + "\n");
   jta.append("\n\n\n\n\n"
               + "_____________________________POVZETEK_____________________________"
               + "\n\n");
   jta.append(
         "DELNICA" + "\t" + "INDEX"
         + "\t" + "NADPOVPRECEN" + "\t"
         + "PODPOVPRECEN" + "\t"
         + "ZANESLJIVOST ZADETKA" + "\t"
         + "PONDER. KRITERIJ" + "\n");

   for (int stTrenutnaVrstica = 0; stTrenutnaVrstica
         < narejeniprimerki.stAtt; stTrenutnaVrstica++) {

      jta.append(
            imenaDelnic[stTrenutnaVrstica]
                  + "\t"
                  + indexDelnice[stTrenutnaVrstica]
                  + "\t"
                  + nf2.format(
                        distributionForInstN[stTrenutnaVrstica][1])
                        + "\t"
                        + "\t"
                        + nf2.format(
                           distributionForInstN[stTrenutnaVrstica][0])
                              + "\t"
                              + "\t"
                              + nf2.format(
                               ZanesljivostZadetka[stTrenutnaVrstica])
                                    + "\t"
                                    + "\t"
                                    + nf2.format(PondKriterij[stTrenutnaVrstica])
                                     + "\n");

      System.out.println(
                  imenaDelnic[stTrenutnaVrstica]
                  + "\t"
                  + indexDelnice[stTrenutnaVrstica]
                  + "\t"
                  + nf2.format(
                        distributionForInstN[stTrenutnaVrstica][1])
                        + "\t"
                        + "\t"
                        + "\t"
                        + nf2.format(distributionForInstN[stTrenutnaVrstica][0])
                              + "\t"
                              + "\t"
                              + "\t"
                              + nf2.format(ZanesljivostZadetka[stTrenutnaVrstica])
                                    + "\t"
                                    + "\t"
                                    + "\t"
                                    + "\t"
                                    + nf2.format(PondKriterij[stTrenutnaVrstica]));
```

```
    }

    if (PondKriterijPrever > 0) {
        jta.append(
                        "\n"
                    + "Potencialno najbolj donosna delnica je "
                    + imenaDelnic[orodjarna.maxIndex(PondKriterij)]);
        jta.append(
                        "\n"
                    + "Dosezeni Donos za dan "
                    + (g + 1)
                    + " za delnico "
                    + imenaDelnic[orodjarna.maxIndex(PondKriterij)]
                    + " bi znasal: "
                    + nfPR2.format( deljeniDelDonos)
                    + "\n");

        System.out.println(
                        "Dosezeni Donos za dan "
                    + (g + 1)
                    + " bi znasal: "
                    + nfPR2.format(deljeniDelDonos));
    } else {
        jta.append(
                        "\n"
                    + "nespremenjeni vlozki");
    }
}

else if (alg == "J48 Drevo") {

    System.out.println(
            "DELNICA" + "\t" + "INDEX"
            + "\t" + "NADPOVPRECEN" + "\t"
            + "\t" + "PODPOVPRECEN" + "\t"
            + "\t" + "ZANESLJIVOST ZADETKA"
            + "\t" + "\t"
            + "VELIKOST DREVESA" + "\t"
            + "\t" + "PONDER. KRITERIJ"
            + "\n");
    jta.append(
            "\n\n\n\n"
              +"_____POVZETEK_____"
              + "\n\n");
    jta.append(
            "DELNICA" + "\t" + "INDEX"
            + "\t" + "NADPOVPRECEN" + "\t"
            + "PODPOVPRECEN" + "\t"
            + "ZANESLJIVOST ZADETKA" + "\t"
            + "VELIKOST DREVESA" + "\t"
            + "PONDER. KRITERIJ" + "\n");

    for (int stTrenutnaVrstica = 0; stTrenutnaVrstica< narejeniprimerki.stAtt; stTrenutnaVrstica++) {
        jta.append( imenaDelnic[stTrenutnaVrstica]
                        + "\t"
                        + indexDelnice[stTrenutnaVrstica]
                        + "\t"
                        + nf2.format(distributionForInst[stTrenutnaVrstica][1])
                              + "\t"
                              + "\t"
                              + nf2.format(distributionForInst[stTrenutnaVrstica][0])
                                    + "\t"
                                    + "\t"
                                    + nf2.format(ZanesljivostZadetka[stTrenutnaVrstica])
                                          + "\t"
                                          + "\t"
                                          +VelikostDrevesa[stTrenutnaVrstica]
                                          + "\t"
                                          + "\t"
                                          + nf2.format(PondKriterij[stTrenutnaVrstica])
                                          + "\n");

        System.out.println(imenaDelnic[stTrenutnaVrstica]
                        + "\t"
                        + indexDelnice[stTrenutnaVrstica]
                        + "\t"
                        + nf2.format(
```

```
                        distributionForInst[stTrenutnaVrstica][1])
                        + "\t"
                        + "\t"
                        + "\t"
                        + nf2.format(distributionForInst[stTrenutnaVrstica][0])
                                + "\t"
                                + "\t"
                                + "\t"
                                + nf2.format(ZanesljivostZadetka[stTrenutnaVrstica])
                                        + "\t"
                                        + "\t"
                                        + "\t"
                                        + "\t"
                                        +VelikostDrevesa[stTrenutnaVrstica]
                                        + "\t"
                                        + "\t"
                                        + "\t"
                                        + "\t"
                                        + nf2.format(PondKriterij[stTrenutnaVrstica]));
    }

    if (PondKriterijPrever > 0) {
        jta.append(
                         "\n"
                        + "Potencialno najbolj donosna delnica je "
                        + imenaDelnic[orodjarna.maxIndex(PondKriterij)]);
        jta.append(
                       "\n"
                    + "Dosezeni Donos za dan "
                    + (g + 1)
                    + " za delnico "
                    + imenaDelnic[orodjarna.maxIndex(PondKriterij)]
                    + " bi znasal: "
                    + nfPR2.format(deljeniDelDonos)+ "\n");

        System.out.println(
                        "Dosezeni Donos za dan "
                    + (g + 1)
                    + " bi znasal: "
                    + nfPR2.format(deljeniDelDonos));
    } else {
        jta.append("\n"+ "nespremenjeni vlozki");
    }
}
else if (alg == "K-sosedov") {

    System.out.println(
            "DELNICA" + "\t" + "INDEX"
            + "\t" + "NADPOVPRECEN" + "\t"
            + "\t" + "PODPOVPRECEN" + "\t"
            + "\t" + "ZANESLJIVOST ZADETKA"
            + "\t" + "\t"
            + "PONDER. KRITERIJ" + "\n");
    jta.append(
                    "\n\n\n\n"
                + "_____POVZETEK_____"
                + "\n\n");
    jta.append(
            "DELNICA" + "\t" + "INDEX"
            + "\t" + "NADPOVPRECEN" + "\t"
            + "PODPOVPRECEN" + "\t"
            + "ZANESLJIVOST ZADETKA" + "\t"
            + "PONDER. KRITERIJ" + "\n");


    for (int stTrenutnaVrstica = 0; stTrenutnaVrstica< narejeniprimerki.stAtt; stTrenutnaVrstica++) {
        jta.append(imenaDelnic[stTrenutnaVrstica]
                        + "\t"
                        + indexDelnice[stTrenutnaVrstica]
                        + "\t"
                        + nf2.format(distributionForInstK[stTrenutnaVrstica][1])
                                + "\t"
                                + "\t"
                                + nf2.format(distributionForInstK[stTrenutnaVrstica][0])
                                        + "\t"
                                        + "\t"
                                        + nf2.format(ZanesljivostZadetka[stTrenutnaVrstica])
```

```
                                        + "\t"
                                        + "\t"
                                        + nf2.format(PondKriterij[stTrenutnaVrstica])
                                        + "\n");

            System.out.println(imenaDelnic[stTrenutnaVrstica]
                        + "\t"
                        + indexDelnice[stTrenutnaVrstica]
                        + "\t"
                        + nf2.format(distributionForInstK[stTrenutnaVrstica][1])
                              + "\t"
                              + "\t"
                              + "\t"
                              + nf2.format(distributionForInstK[stTrenutnaVrstica][0])
                                    + "\t"
                                    + "\t"
                                    + "\t"
                                    + nf2.format(ZanesljivostZadetka[stTrenutnaVrstica])
                                          + "\t"
                                          + "\t"
                                          + "\t"
                                          + "\t"
                                          + nf2.format(PondKriterij[stTrenutnaVrstica]));
         }
         jta.append(
                        "\n"
                     + "Potencialno najbolj donosna delnica je "
                     + imenaDelnic[orodjarna.maxIndex(PondKriterij)]);
         jta.append("\n" + "Dosezeni Donos za dan "+ (g + 1) + " za delnico "
                           + imenaDelnic[orodjarna.maxIndex(PondKriterij)]
               + " bi znasal: "
               + nfPR2.format(deljeniDelDonos)
               + "\n");

         System.out.println(
                        "Dosezeni Donos za dan "
                     + (g + 1)
                     + " bi znasal: "
                     + nfPR2.format(deljeniDelDonos));
      }
   } else {

      if (PondKriterijPrever > 0) {
         stDelnice = indexDelnice[orodjarna.maxIndex(PondKriterij)];
         double vrednostNext = instancaZaVrednNext.value(stDelnice);
         double vrednost = instancaZaVred.value(stDelnice);
         delDonos = vrednostNext - vrednost;
         deljeniDelDonos = delDonos / vrednost;
         skupniDonos = skupniDonos+ deljeniDelDonos;

         jta.append(
               "\n" + "Dosezeni Donos za dan "
               + (g + 1) + " za delnico  "
               + imenaDelnic[orodjarna.maxIndex(PondKriterij)]
               + " na dan: " + datum
               + " bi znasal: "
               + nfPR2.format(deljeniDelDonos)
               + " Skupni donos pa " + "\t"
               + nfPR2.format(skupniDonos));
         System.out.println(
                        "Dosezeni Donos za dan "
                     + (g + 1)
                     + " za delnico  "
                     + imenaDelnic[orodjarna.maxIndex(PondKriterij)]
                     + " na dan: " + datum
                     + " bi znasal: "
                     + nfPR2.format(
                           deljeniDelDonos)
                           + " Skupni donos pa "
                           + "\t"
                           + nfPR2.format(
                                 skupniDonos));

         } else {
            deljeniDelDonos = 0.0;
            skupniDonos = skupniDonos+ deljeniDelDonos;
            jta.append(
```

```
                        "\n" + "nespremenjeni vlozki"
                        + " Skupni donos pa " + "\t"
                        + nfPR2.format(skupniDonos));
                }

                najboljseDrsecePovprecje = movingAverageAnaliza[orodjarna.maxIndex(PondKriterij)];
                najboljsaCasovnaVrsta = vrstaDelAnaliza[orodjarna.maxIndex(PondKriterij)];

              }

            } catch (Exception ex) {
               System.err.println(
                     "Exception: " + ex.getMessage());
            }

            instancesZaTrenirat.add(
                  instancesZavrednotit.instance(g));

         }
         System.out.println(
               "\n" + "Skupni donos za obdobje " + stDni
               + " dni bi znasal:  "
               + nfPR2.format(skupniDonos));

         jta.append(
               "\n" + "Skupni donos za obdobje " + stDni
               + " dni bi znasal:  "
               + nfPR2.format(skupniDonos));

      } catch (Exception io) {
         System.err.println("Exception: " + io.getMessage());
      }
   }
   // this.stop();
  }
}


class importL implements ActionListener {
  public void actionPerformed(ActionEvent e) {
     Import importL1 = new Import();
     importL1.start();
     importL1.setPriority(Thread.MIN_PRIORITY);

  }
}


class analizaL implements ActionListener {
  public void actionPerformed(ActionEvent e) {
     Analiza analizaL1 = new Analiza();
     analizaL1.start();
     analizaL1.setPriority(Thread.MIN_PRIORITY);

  }
}

file.add(jmi = new JMenuItem("New"));
jmi.addActionListener(new newL());
file.add(jmi = new JMenuItem("Open"));
jmi.addActionListener(new openL());
file.add(jmi = new JMenuItem("Save"));
jmi.addActionListener(new saveL());
file.addSeparator();
file.add(jmi = new JMenuItem("Exit"));
jmi.addActionListener(new exitL());
edit.add(jmi = new JMenuItem("Copy"));
jmi.addActionListener(new copyL());
edit.add(jmi = new JMenuItem("Cut"));
jmi.addActionListener(new cutL());
edit.add(jmi = new JMenuItem("Paste"));
jmi.addActionListener(new pasteL());
analizaJM.add(jmi = new JMenuItem("model"));
jmi.addActionListener(new analizaL());
setLayout(new BorderLayout());
jmb.add(file);
jmb.add(edit);
```

```
jmb.add(analizaJM);

String imgLocationNew = "toolbarButtonGraphics/general/New24.gif";
String imgLocationOpen = "toolbarButtonGraphics/general/Open24.gif";
String imgLocationSaveAs = "toolbarButtonGraphics/general/SaveAs24.gif";
String imgLocationCopy = "toolbarButtonGraphics/general/Copy24.gif";
String imgLocationPaste = "toolbarButtonGraphics/general/Paste24.gif";
String imgLocationImport = "toolbarButtonGraphics/general/Import24.gif";
String imgLocationAnaliza = "toolbarButtonGraphics/general/Stop24.gif";

URL imageURLNew = getClass().getResource(imgLocationNew);
URL imageURLOpen = getClass().getResource(imgLocationOpen);
URL imageURLSaveAs = getClass().getResource(imgLocationSaveAs);
URL imageURLCopy = getClass().getResource(imgLocationCopy);
URL imageURLPaste = getClass().getResource(imgLocationPaste);
URL imageURLAnaliza = getClass().getResource(imgLocationAnaliza);
URL imageURLImport = getClass().getResource(imgLocationImport);

JButton buttonNew = null;
JButton buttonOpen = null;
JButton buttonSaveAs = null;
JButton buttonCopy = null;
JButton buttonPaste = null;
JButton buttonAnaliza = null;
JButton buttonImport = null;

if (imageURLNew != null) {
    buttonNew = new JButton(new ImageIcon(imageURLNew));
    buttonNew.addActionListener(new newL());
}
toolBar.add(buttonNew);

if (imageURLOpen != null) {
    buttonOpen = new JButton(new ImageIcon(imageURLOpen));
    buttonOpen.addActionListener(new openL());
}
toolBar.add(buttonOpen);

if (imageURLSaveAs != null) {
    buttonSaveAs = new JButton(new ImageIcon(imageURLSaveAs));
    buttonSaveAs.addActionListener(new saveL());
}
toolBar.add(buttonSaveAs);

if (imageURLCopy != null) {
    buttonCopy = new JButton(new ImageIcon(imageURLCopy));
    buttonCopy.addActionListener(new copyL());
}
toolBar.add(buttonCopy);

if (imageURLPaste != null) {
    buttonPaste = new JButton(new ImageIcon(imageURLPaste));
    buttonPaste.addActionListener(new pasteL());
}
toolBar.add(buttonPaste);

toolBar.addSeparator();
toolBar.setFloatable(false);

if (imageURLImport != null) {
    buttonImport = new JButton(new ImageIcon(imageURLImport));
    buttonImport.addActionListener(new importL());
}
toolBar.add(buttonImport);

if (imageURLAnaliza != null) {
    buttonAnaliza = new JButton(new ImageIcon(imageURLAnaliza));
    buttonAnaliza.addActionListener(new analizaL());
}
toolBar.add(buttonAnaliza);
JPanel bars = new JPanel();
bars.setLayout(new BorderLayout());
bars.add(jmb, BorderLayout.NORTH);
bars.add(toolBar, BorderLayout.CENTER);
jtb.addTab("Poročilo", null, jsp, "Poročilo opravljene obdelave");
add(bars, BorderLayout.NORTH);
add(jtb, BorderLayout.CENTER);
```

```
    }

/*
            Začetna funkcija razreda napovedovalec, katero po zagonu programa pokliče javansko okolje
            Osnovna funkcionalnost zbrana v tej metodi je ustvarjanje samega objekta Napovedovalca,
            ki je v bistvu JPanel in ga je potrebno postaviti v okenski okvir ter prikazati.
            Poleg ustvarjanja objekta napovedovalec začetna metoda main ustvari še objekt InstanceHolder

*/
    public static void main(String args[]) {
        JFrame f = new JFrame();
        Napovedovalec napovedovalec = new Napovedovalec();
        f.setTitle("Napovedovalec");
        f.setBackground(Color.lightGray);
        f.getContentPane().add(napovedovalec, BorderLayout.CENTER);
        f.addWindowListener(new appCloseL());
        f.setSize(800, 500);
        f.setVisible(true);
        InstanceHolder holder = new InstanceHolder();

    }

/*
            Razred, ki skrbi za končanje aplikacije in vrnitev v operacijski sistem.
*/
    protected static final class appCloseL extends WindowAdapter {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    }

/*
            Razred UstvariPoljePrimerkov, je zadolžen, da utvari časovne vrste rangov z časovnimi zamiki,
            Najprej na osnovi vhodnih instanc, ki predstavljajo cene delnic za vsako posamezno delnico,
            nato ustvari časovne vrste donosov, ter jih na osnovi donosov rangira po vrsti tako, da
            rangi delnic ostanejo v razponu med -0,5 in +0,5.
*/
    class UstvariPoljePrimerkov {
        public int stAtt;
        public Instances[] vseinstance;
        private FastVector[] atributi;
        private Instances preneseneInstance;
        private String[] imenaRelacij;
        private Instance novaInstanca;
        private Instances instanceZaNapolnit;
        private Instance instst0;
        private Instance instst1;
        private Instance instst2;
        private Instance instst3;
        private Instance instst4;
        private Instance instst5;
        private Instance instst20;
        private Instance instst prva;
        public Attribute uspesnost;

        public UstvariPoljePrimerkov(Instances instances) {
            try {
                preneseneInstance = instances;
                stAtt = preneseneInstance.numAttributes();
                atributi = new FastVector[stAtt];
                imenaRelacij = new String[stAtt];
                vseinstance = new Instances[preneseneInstance.numAttributes()];

                for (int i = 0; i < preneseneInstance.numAttributes(); i++) {
                    imenaRelacij[i] = new String();
                    imenaRelacij[i] = preneseneInstance.attribute(i).name();
                    atributi[i] = new FastVector();
                    atributi[i] = naredi Vector(preneseneInstance.attribute(i).name());
                    vseinstance[i] = new Instances(imenaRelacij[i], atributi[i],
                    preneseneInstance.numInstances() - 20);
                    vseinstance[i].setClassIndex(0);
                    napolniInstance(vseinstance[i], i);
                }

            } catch (Exception ex) {
                System.err.println("Exception: " + ex.getMessage());
            }
```

```
}

public FastVector narediVector(String imeatributa) {
   String atribut = new String(imeatributa);
   FastVector vektor = new FastVector();

   for (int j = 0; j < 6; j++) {
      vektor.addElement(new Attribute(atribut + j));
   }
   return vektor;
}

public void izpisiprimerke() {
   for (int i = 0; i < preneseneInstance.numAttributes(); i++) {
      System.out.println("instanca" + i + ":\n");
      System.out.println(vseinstance[i]);
   }
}

public void napolniInstance(Instances prazneinstance, int kateriAtt) {
   int Attstev = kateriAtt;
   instanceZaNapolnit = prazneinstance;

   for (int g = 1; g <= preneseneInstance.numInstances() - 20; g++) {
      int st0 = preneseneInstance.numInstances() - g;
      int st1 = st0 - 1;
      int st2 = st0 - 2;
      int st3 = st0 - 3;
      int st4 = st0 - 4;
      int st5 = st0 - 5;
      int st20 = st0 - 20;
      int stprva = preneseneInstance.numInstances()- preneseneInstance.numInstances();

      instst0 = preneseneInstance.instance(st0);
      instst1 = preneseneInstance.instance(st1);
      instst2 = preneseneInstance.instance(st2);
      instst3 = preneseneInstance.instance(st3);
      instst4 = preneseneInstance.instance(st4);
      instst5 = preneseneInstance.instance(st5);
      instst20 = preneseneInstance.instance(st20);
      inststprva = preneseneInstance.instance(stprva);
      novaInstanca = narediInstanco(instst0, instst1, instst2, instst3,instst4, instst5, instst20, Attstev);
      vseinstance[Attstev].add(novaInstanca);
   }
}

public Instance narediInstanco(Instance instanca0, Instance instanca1,
                               Instance instanca2, Instance instanca3, Instance instanca4,
                               Instance instanca5, Instance instanca20, int i) {
   int Attstev = i;
   int koliko;

   double instancniValue0;
   double instancniValue1;
   double instancniValue2;
   double instancniValue3;
   double instancniValue4;
   double instancniValue5;
   double instancniValue20;

   double n_instancniValue0;
   double n_instancniValue1;
   double n_instancniValue2;
   double n_instancniValue3;
   double n_instancniValue4;
   double n_instancniValue5;
   double n_instancniValue20;

   double[] n_instancniValueSkupaj;

   String ime0;
   String ime1;
   String ime2;
   String ime3;
   String ime4;
   String ime5;
   String ime20;
```

```
        Instance n_instanca0 = instanca0;
        Instance n_instanca1 = instanca1;
        Instance n_instanca2 = instanca2;
        Instance n_instanca3 = instanca3;
        Instance n_instanca4 = instanca4;
        Instance n_instanca5 = instanca5;
        Instance n_instanca20 = instanca20;

        instancniValue0 = n_instanca0.value(Attstev);
        instancniValue1 = n_instanca1.value(Attstev);
        instancniValue2 = n_instanca2.value(Attstev);
        instancniValue3 = n_instanca3.value(Attstev);
        instancniValue4 = n_instanca4.value(Attstev);
        instancniValue5 = n_instanca5.value(Attstev);
        instancniValue20 = n_instanca20.value(Attstev);

        n_instancniValue0 = (instancniValue0 - instancniValue1)/ instancniValue1;
        n_instancniValue1 = (instancniValue0 - instancniValue2) / instancniValue2;
        n_instancniValue2 = (instancniValue0 - instancniValue3)/ instancniValue3;
        n_instancniValue3 = (instancniValue0 - instancniValue4)/ instancniValue4;
        n_instancniValue4 = (instancniValue0 - instancniValue5)/ instancniValue5;
        n_instancniValue20 = (instancniValue0 - instancniValue20)/ instancniValue20;
        n_instancniValueSkupaj = new double[6];
        n_instancniValueSkupaj[0] = n_instancniValue0;
        n_instancniValueSkupaj[1] = n_instancniValue1;
        n_instancniValueSkupaj[2] = n_instancniValue2;
        n_instancniValueSkupaj[3] = n_instancniValue3;
        n_instancniValueSkupaj[4] = n_instancniValue4;
        n_instancniValueSkupaj[5] = n_instancniValue20;
        Instance narejenaInstanca = new Instance(6);
        narejenaInstanca.setDataset(instanceZaNapolnit);

        for (int z = 0; z < narejenaInstanca.numAttributes(); z++) {
            narejenaInstanca.setValue(z, n_instancniValueSkupaj[z]);
        }
        return narejenaInstanca;
    }
  }

/*
razred Precistilec je zadolžen za nadomestitev vrednosti prvih dni (razred podatkovnega primera)
z vrednostjo "nadpovprecen" ali pa "podpovprecen" v skladu s tem ali je razredni rang večji od 0
ali pa manjši.Nato razdeli instance tako ustvarjenih rangov na tri dele in sicer del za učenje,
kjer se zgradi model, del podatkov za preizkus modela ter dneve, ki so potrebni za trgovanje.
*/

   class Precistilec {
      public Instances[] poljeAjevString;
      public Instances[] poljeAjevStringPrazni;
      public FastVector nominalneVrednosti;
      public Attribute uspesnost;
      public Instances[] trainInstances;
      public Instances[] testInstances;
      public Instance[] zadnjeInstance;
      public double deleztrain;
      public double deleztest;


      public Precistilec(Instances[] poljeAjevRaw) {
         try {
            poljeAjevString = poljeAjevRaw;
            poljeAjevStringPrazni = new Instances[poljeAjevString.length];
            nominalneVrednosti = new FastVector(2);
            nominalneVrednosti.addElement("podpovprecen");
            nominalneVrednosti.addElement("nadpovprecen");
            uspesnost = new Attribute("uspesnost", nominalneVrednosti);

            for (int stTrenutniSkupekA = 0; stTrenutniSkupekA< poljeAjevString.length; stTrenutniSkupekA++) {
               Instances trenutniSkupek = poljeAjevString[stTrenutniSkupekA];
               trenutniSkupek.insertAttributeAt(uspesnost, 6);
               trenutniSkupek.setClassIndex(6);
               for (int stTrenutnaInstanca = 0; stTrenutnaInstanca< trenutniSkupek.numInstances(); stTrenutnaInstanca++) {
                  Instance trenutnaInstanca = poljeAjevString[stTrenutniSkupekA].instance(stTrenutnaInstanca);
                  double vrednost = trenutniSkupek.instance(stTrenutnaInstanca).value(0);

                  if (trenutniSkupek.instance(stTrenutnaInstanca).value(0)<= 0.0) {
```

```
                    trenutniSkupek.instance(stTrenutnaInstanca).setValue(6, "podpovprecen");
                } else {
                    trenutniSkupek.instance(stTrenutnaInstanca).setValue(6, "nadpovprecen");
                }
            }
            poljeAjevStringPrazni[stTrenutniSkupekA] = trenutniSkupek;
            poljeAjevStringPrazni[stTrenutniSkupekA].deleteAttributeAt(0);
        }
        trainInstances = new Instances[poljeAjevStringPrazni.length];
        testInstances = new Instances[poljeAjevStringPrazni.length];
        zadnjeInstance = new Instance[poljeAjevStringPrazni.length];
        deleztrain = 0.6666666;
        deleztest = 0.33333333;

        for (int stTrenutniSkupekPrazni = 0; stTrenutniSkupekPrazni< poljeAjevStringPrazni.length;
        stTrenutniSkupekPrazni++) {
            trainInstances[stTrenutniSkupekPrazni] = new Instances(
                    poljeAjevStringPrazni[stTrenutniSkupekPrazni], 0,
                    (int) (poljeAjevStringPrazni[stTrenutniSkupekPrazni].numInstances()* deleztrain));
                        testInstances[stTrenutniSkupekPrazni] = new Instances(poljeAjevStringPrazni[stTrenutniSkupekPrazni],
                    ((int) (poljeAjevStringPrazni[stTrenutniSkupekPrazni].numInstances()* deleztrain)),
                            poljeAjevStringPrazni[stTrenutniSkupekPrazni].numInstances()
                             - (((int) (poljeAjevStringPrazni[stTrenutniSkupekPrazni].numInstances()
                             * deleztrain))+ 1));
            zadnjeInstance[stTrenutniSkupekPrazni] = poljeAjevStringPrazni[stTrenutniSkupekPrazni].lastInstance();
        }
      } catch (Exception ex) {
        System.err.println("Exception: " + ex.getMessage());
      }
    }

    public void izpisiprimerke() {
      for (int i = 0; i < poljeAjevStringPrazni.length; i++) {
        System.out.println("trainInstanca" + i + ":\n");
        System.out.println(trainInstances[i]);
        System.out.println("testInstanca" + i + ":\n");
        System.out.println(testInstances[i]);
      }
    }
  }


  class ExcelAdapter implements ActionListener {
    private String rowstring, value;
    private Clipboard system;
    private StringSelection stsel;
    private JTable jTable1;

    /*
     ExcelAdapter je skonstruiran s pomočjo JTable razreda, na kateri omogočiCopy-Paste
     funkcijo ter se obnaša kot razred, ki posluša ukaz odložišča.
     */

    public ExcelAdapter(JTable myJTable) {
      jTable1 = myJTable;
      KeyStroke copy = KeyStroke.getKeyStroke(KeyEvent.VK_C,ActionEvent.CTRL_MASK, false);
      KeyStroke paste = KeyStroke.getKeyStroke(KeyEvent.VK_V,ActionEvent.CTRL_MASK, false);


      jTable1.registerKeyboardAction(this, "Copy", copy,JComponent.WHEN_FOCUSED);
      jTable1.registerKeyboardAction(this, "Paste", paste,JComponent.WHEN_FOCUSED);
      system = Toolkit.getDefaultToolkit().getSystemClipboard();
    }
/*
Javne metode, ki omogočajo dostop do tabele in prenos tabele na kateri ta
adapter deluje
*/

    public JTable getJTable() {
      return jTable1;
    }

    public void setJTable(JTable jTable1) {
      this.jTable1 = jTable1;
    }
/*
Ta metoda deluje na podlagi ukaznih tipk, ki jih poslušamo. To so ukazi za kopiranje in leplenje
```

```
Če izberemo celice ki se ne dotikajo je izbor nepravilen in tako kopiranja ter leplenja ne
moremo izvesti. Leplenje se izvede tako da se zgornji levi rob izbora poravna z prvim elementom
v trenutnem izboru tabele JTable.
*/

      public void actionPerformed(ActionEvent e) {
         if (e.getActionCommand().compareTo("Copy") == 0) {
            StringBuffer sbf = new StringBuffer();
            // Zagotovljen mora biti izbor celic ki se dotikajo

            int numcols = jTable1.getSelectedColumnCount();
            int numrows = jTable1.getSelectedRowCount();
            int[] rowsselected = jTable1.getSelectedRows();
            int[] colsselected = jTable1.getSelectedColumns();

            if (!((numrows - 1
                  == rowsselected[rowsselected.length - 1]
                        - rowsselected[0]
                              && numrows == rowsselected.length)
                                    && (numcols - 1
                                          == colsselected[colsselected.length - 1]
                                                - colsselected[0]
                                                      && numcols
                                                            == colsselected.length))) {
               JOptionPane.showMessageDialog(null, "Invalid Copy Selection",
                     "Invalid Copy Selection", JOptionPane.ERROR_MESSAGE);
               return;
            }
            for (int i = 0; i < numrows; i++) {
               for (int j = 0; j < numcols; j++) {
                  sbf.append(
                        jTable1.getValueAt(rowsselected[i],
                        colsselected[j]));
                  if (j < numcols - 1) {
                     sbf.append("\t");
                  }
               }
               sbf.append("\n");
            }
            stsel = new StringSelection(sbf.toString());
            system = Toolkit.getDefaultToolkit().getSystemClipboard();
            system.setContents(stsel, stsel);
         }
         if (e.getActionCommand().compareTo("Paste") == 0) {

            int startRow = (jTable1.getSelectedRows())[0];
            int startCol = (jTable1.getSelectedColumns())[0];

            try {
               String trstring = (String) (system.getContents(this).getTransferData(DataFlavor.stringFlavor));
               StringTokenizer st1 = new StringTokenizer(trstring, "\n");

               for (int i = 0; st1.hasMoreTokens(); i++) {
                  rowstring = st1.nextToken();
                  StringTokenizer st2 = new StringTokenizer(rowstring,"\t");

                  for (int j = 0; st2.hasMoreTokens(); j++) {
                     value = (String) st2.nextToken();
                     if (startRow + i < jTable1.getRowCount()
                           && startCol + j < jTable1.getColumnCount()) {jTable1.isCellEditable(startRow + i,startCol + j);
                     }

                     jTable1.setValueAt(value, startRow + i, startCol + j);

                  }
               }
            } catch (Exception ex) {
               ex.printStackTrace();
            }
         }
      }
   }


   class MyTableModel extends AbstractTableModel {
      public Instances instance;
```

```
        public void setirajInstance(Instances prInstance) {
            instance = prInstance;
        }

        public int getColumnCount() {
            return instance.numAttributes();

        }

        public int getRowCount() {
            return instance.numInstances();
        }

        public String getColumnName(int col) {
            return instance.attribute(col).name();
        }

        public Object getValueAt(int row, int col) {

            return instance.instance(row).toString(col);

        }

        public boolean isCellEditable(int row, int col) {

            return true;
        }

        public void setValueAt(Object value, int row, int col) {

            if (col == 1) {

                instance.instance(row).setValue(col, (String) value);

            } else if (col >= 1) {

                instance.instance(row).setValue(col, value.toString());
            }

            fireTableCellUpdated(row, col);
        }

    }

}
/*
Razred, ki služi za prenos instanc med razredi.
*/

class InstanceHolder {

    public Instances instance = null;
    public TimeSeries[] drsecaPovprecjaDelnic;
    public TimeSeries[] casovneVrsteDelnic;

    public Instances dajInstance() {
        return instance;
    }

    public void vpisiInstance(Instances instZaVpis) {
        instance = instZaVpis;
    }

    public void vpisiDrsecaPovprecjaDelnic(TimeSeries[] drsecaPovprecjaDelnicP) {

        drsecaPovprecjaDelnic = drsecaPovprecjaDelnicP;
    }

    public void vpisiCasovneVrsteDelnic(TimeSeries[] casovneVrsteDelnicP) {

        casovneVrsteDelnic = casovneVrsteDelnicP;
    }
}
```

## PRILOGA G

Podatki o kotacijah delnic urejeni v formatu ARFF.

```
@relation podatki8datum

@attribute DATUM string
@attribute AELG numeric
@attribute ETOG numeric
@attribute GRVG numeric
@attribute IEKG numeric
@attribute KRKG numeric
@attribute LKPG numeric
@attribute MER numeric
@attribute TCRG numeric

@data
1998-11-30,2620.00,20900.00,1693.88,1899.97,25049.78,2927.92,18004.84,10709.09
1998-12-01,2648.94,20933.51,1696.57,1900.09,25164.49,2938.28,18014.94,10900.00
1998-12-02,2651.62,20896.11,1688.37,1900.29,25257.53,2948.74,18001.71,10870.00
1998-12-03,2659.02,20870.00,1692.82,1900.69,25138.37,2945.44,18001.71,10710.00
1998-12-04,2650.15,21010.00,1694.90,1904.89,25114.64,2957.28,18000.00,10710.00
1998-12-07,2655.00,21010.00,1690.39,1898.75,25069.83,2946.34,18037.97,10956.49
1998-12-08,2650.00,20891.59,1680.16,1890.50,25028.91,2930.62,18000.00,10956.49
1998-12-09,2655.07,20918.82,1685.18,1900.03,25033.51,2929.10,18000.00,10992.95
                                    .      .      .
                                    .      .      .
                                    .      .      .
```

## SLOVARČEK UPORABLJENIH TUJIH IZRAZOV

machine learning – strojno učenje
lazy learning – leno učenje
k-nearest neighbours – k-najbližjih sosedov
backpropagation algorithm – algoritem vzvratnega širjenja napake
support vector machine – metoda podpornih vektorjev
overtrained – preveč naučen
cross-validation – križno preverjanje
efficient market hypothesis – hipoteza učinkovitih trgov
buy & hold – kupi in drži
market to book ratio – razmerje med tržno in knjigovodsko ceno(delnice)
price earning ratio (P/E) – multiplikator čistega dobička
relative strength index – kazalnik relativne moči
moving average – drseča sredina
hit rate – stopnja pravilnih zadetkov
root mean square error (RMSE) – koren vsote kvadratov napak