Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Daniel Rižnar

# Razvoj informacijskega sistema za opravljanje analiz in predikcij pri borznoposredniških družbah

diplomsko delo na univerzitetnem študiju

Mentor: doc. dr. Dejan Lavbič

Ljubljana, 2013

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Tržaška 25*
*1001 Ljubljana, Slovenija*
*telefon: 01 476 84 11*
*01 476 83 87*
*faks: 01 426 46 47*
*01 476 87 11*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Št. naloge: 01912/2013
Datum: 04.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DANIEL RIŽNAR**

Naslov: **RAZVOJ INFORMACIJSKEGA SISTEMA ZA OPRAVLJANJE ANALIZ IN PREDIKCIJ PRI BORZNOPOSREDNIŠKIH DRUŽBAH**

**ANALYSIS AND PREDICTION INFORMATION SYSTEM DEVELOPMENT FOR BROKERAGE COMPANY**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Osnovna dejavnost borznoposredniških družb je posredovanje pri transakcijah s finančnimi inštrumenti in upravljanje premoženja svojih strank. Za podporo tej osnovni dejavnosti obstajajo številne informacijske rešitve, ki pa slabše podpirajo opravljanje analiz in predikcij, ki bi jih lahko uporabili za optimizacijo poslovnih procesov borznoposredniške družbe. V okviru diplomske naloge opredelite načrt za razvoj informacijske rešitve, ki implementira naslednje funkcionalnosti: izračun donosnosti portfelja, povprečne donosnosti finančnih instrumentov pred izvršitvijo transakcije, volatilnosti portfeljev, življenske vrednosti pogodb, ustvarjanje napovedi zaprtij pogodb in razvrščanje pogodv v gruče.

Mentor:

doc. dr. Dejan Lavbič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Daniel Rižnar, z vpisno številko **63060280**, sem avtor diplomskega dela z naslovom:

*Razvoj informacijskega sistema za opravljanje analiz in predikcij pri borznoposredniških družbah.*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 19. septembra 2013 Podpis avtorja:

# Kazalo

**Povzetek**

**Abstract**

**1 Uvod** **1**

**2 Ovrednotenje obstoječih informacijskih sistemov in problemska domena** **5**

2.1 Ovrednotenje obstoječih informacijskih sistemov . . . . . 5

2.2 Problemska domena . . . . . 15

**3 Predstavitev tehnologij in metode strojnega učenja** **19**

3.1 Tehnologije . . . . . 19

3.2 Metode strojnega učenja . . . . . 22

3.2.1 Razvrščanje . . . . . 22

3.2.2 Klasifikacija . . . . . 28

**4 Informacijski sistem za opravljanje analiz in predikcij** **33**

4.1 Podatkovna shramba . . . . . 34

4.1.1 Šifranti . . . . . 38

4.1.2 Pogodbe strank . . . . . 39

4.1.3 Viri za agregate . . . . . 39

4.1.4 Agregati . . . . . 40

4.1.5 Rezultati algoritmov . . . . . 41

4.1.6 Sistemske tabele . . . . . 42

4.2 Prenos podatkov . . . . . 43
4.2.1 Izračun vrednosti portfeljev . . . . . 47
4.2.2 Glajenje sprememb vrednosti portfeljev . . . . . 49
4.2.3 Izračun donosnosti portfeljev . . . . . 50
4.2.4 Zapiranje in odpiranje pogodb . . . . . 51
4.2.5 Izračun agregatov . . . . . 53
4.2.6 Donosnost finančnih instrumentov pred izvršitvijo transakcije . . . . . 55
4.3 Preostali izračuni in metode strojnega učenja . . . . . 56
4.3.1 Volatilnost . . . . . 58
4.3.2 Življenjska vrednost pogodb . . . . . 59
4.3.3 Napovedovanje zaprtij pogodb . . . . . 61
4.3.4 Razvrščanje pogodb . . . . . 64
4.4 Prikaz podatkov . . . . . 70
4.4.1 Pregled celotne množice pogodb . . . . . 71
4.4.2 Pregled zbirke gruč . . . . . 75
4.4.3 Pregled posamezne gruče . . . . . 76

**5 Sklepne ugotovitve** **81**

# Seznam uporabljenih kratic in simbolov

**.NET** - programsko ogrodje, znotraj katerega se izvaja *C#*

**ASP.NET MVC** - ogrodje za izdelavo spletnih aplikacij, napisanih v *C#-u*

**BPD** - okrajšava za borznoposredniško družbo

**C#** - splošnonamenski objektno orientiran programski jezik

**CARET** *(angl. Classification And Regression Training)* - knjižnica za statistično obdelavo podatkov in strojno učenje za programski jezik *R*

**CIL** *(angl. Common Intermediate Language)* - vmesni jezik, v katerega se prevede *C#* pred prevedbo v strojno kodo

**CLR** *(angl. Common Language Runtime)* - virtualni stroj, na katerem se izvaja *CIL*

**CSS** *(angl. Cascading Style Sheet)* - slogovni jezik za izdelavo spletnih strani

**Flot** - knjižnica za *JavaScript*, namenjena izdelavi grafov

**HTML** *(angl. HyperText Markup Language)* - označevalni jezik za izdelavo spletnih strani

**IIS** *(angl. Internet Information Services)* - spletni strežnik za izvajanje programov, napisanih v *C#-u*

**IS** - okrajšava za informacijski sistem

**JavaScript** - skriptni programski jezik za izdelavo dinamičnih spletnih strani

**JIT** *(angl. Just-in-Time Compilation)* - tehnika za prevajanje *CIL-a* v strojno kodo

**jQuery** - knjižnica za *JavaScript*

**LJSE** - oznaka za Ljubljansko borzo

**R** - programski jezik, primarno namenjen statističnim analizam podatkov in strojnemu učenju

**R.NET** - knjižnica za medsebojno povezavo *R-ja* in *C#-a*

**T-SQL** *(angl. Structured Query Language)* - programski jezik za izvajanje operacij nad relacijskimi podatkovnimi bazami

**URL** *(angl. Uniform Resource Locator)* - naslova za spletno stran

# Povzetek

V tem delu zgradimo prototip informacijskega sistema, ki omogoča opravljanje analiz in predikcij nad podatki borznoposredniških družb. V okviru analiz in predikcij implementiramo šest ključnih funkcionalnosti: izračun donosnosti portfeljev, izračun povprečne donosnosti finančnih instrumentov pred izvršitvijo transakcije, izračun volatilnosti portfeljev, izračun življenjske vrednosti pogodb, ustvarjanje napovedi zaprtij pogodb in razvrščanje pogodb v gruče. Pri zadnjih dveh funkcionalnostih uporabimo metode strojnega učenja: za napovedovanje zaprtij pogodb uporabimo algoritem naključnih gozdov in za razvrščanje algoritem hierarhičnega razvrščanja. Njuno učinkovitost dokažemo nad realnimi podatki, ki smo jih pridobili pri eni od domačih borznoposredniških družb. Prav tako ustvarimo dve podatkovni bazi, ki hranita podatke, nad katerimi izvajamo naštete funkcionalnosti. Implementiramo tudi samodejni prenos podatkov v omenjeni bazi, in sicer iz podatkovnih baz borznoposredniških družb. Na koncu zgradimo uporabniški vmesnik v obliki spletne aplikacije, kjer s pomočjo grafov in tabel prikažemo rezultate analiz in predikcij.

**Ključne besede:**
informacijski sistem, strojno učenje, klasifikacija, razvrščanje

# Abstract

In this thesis we build a prototype of an information system, which enables us the performance of various analyses and predictions on the data of brokerage firms. Within analyses and predictions we implement six key functionalities: portfolio return rates calculation, calculation of average return rates of financial instruments before transaction execution, portfolio volatility calculation, account lifetime value calculation, predictions about whether accounts are to be closed and clustering of accounts. For the purposes of the last two of the enumerated functionalities, we use machine learning methods: we use random forests algorithm for predictions and hierarchical clustering algorithm for clustering. We demonstrate the efficiency of both algorithms on real data that was provided to us by one of the domestic brokerage firms. We also create two databases to store the data on which all the functionalities are being performed on. We also implement the automatic data-transfer process, which transfers all of the data from brokerage firm's internal database to our databases. In the end we build a user interface in the form of a web application, which uses charts and tables to display the results of analyses and predictions.

**Keywords:**
information system, machine learning, classification, clustering

# Poglavje 1

# Uvod

Med glavne dejavnosti BPD spadata posredovanje pri transakcijah s finančnimi instrumenti in upravljanje premoženja svojih strank. Vsaka stranka, ki želi trgovati s finančnimi instrumenti, mora z BPD skleniti pogodbo. Od tipa pogodbe je potem odvisno, ali gre za posredovanje ali upravljanje. Pri prvem tipu svoja naročila za transakcije na borznih trgih izvaja preko svojega borznega posrednika, ki je zaposlen pri BPD. Pri drugem se odloči, da svoj denar prepusti BPD-ju in potem ta namesto nje odloča, kam investirati ta denar. Vsaki pogodbi pripada portfelj naložb, ki jih stranka ima v danem trenutku pri BPD. Dandanes praktično vsi BPD kot glavno podporo tem dejavnostim uporabljajo IS. V Sloveniji večina BPD uporablja IS *Shark*, podjetja *Evolve d.o.o.* [6] Ta med izvajanjem poslovnih procesov ustvari veliko količino podatkov, ki se shranijo v njegovo podatkovno bazo. Cilj te diplomske naloge je razviti IS, ki nad temi podatki omogoča opravljanje analiz in predikcij, ki potem služijo optimizaciji poslovnih procesov BPD. Analize in predikcije, ki jih IS ponuja, so sestavljene iz šestih ključnih funkcionalnosti:

- **Donosnost portfelja** meri relativno spremembo vrednosti portfelja v določenem obdobju. Mi jo bomo izmerili za štiri različna obdobja, ki so v razponu od enega meseca do enega leta. Ta mera služi kot glavni indikator uspešnosti vlagateljeve strategije in pri snovanju strategije se praktično vedno teži k njeni optimizaciji.

- **Donosnost finančnih instrumentov pred izvršitvijo transakcijo** izmeri povprečno donosnost finančnih instrumentov pred njihovim nakupom in prodajo za posamezen portfelj. Povprečni donosnosti se izračunata ločeno za nabavne in prodajne transakcije. Ta podatek BPD lahko uporabi za dajanje priporočil za nakupe in prodaje finančnih instrumentov.

- **Volatilnost** je mera za izpostavljenost portfelja nihanju njegove vrednosti. Pridobimo jo tako, da izmerimo gibanje vrednosti tečajev finančnih instrumentov znotraj portfelja. Tem bolj se ti gibljejo skupaj, tem večja je volatilnost portfelja in obratno. Mera volatilnosti BPD-ju pove, kateri portfelji so najbolj izpostavljeni riziku, in po potrebi omogoči ustrezno ukrepanje.

- **Življenjska vrednost pogodbe** oceni prihodke, ki jih bo določena pogodba ustvarila v prihodnjih letih. Ocena se naredi v več variacijah. Nekatere upoštevajo zgolj prihodke, ki so bili ustvarjeni na naslov pogodbe v preteklih letih, medtem ko druge vzamejo v zakup še verjetnost zaprtja pogodbe. S pomočjo te ocene BPD dobi pregled nad tem, katere stranke jim ustvarijo največ prihodkov, in lahko posveti večjo pozornost potrebam ter zadovoljstvu teh strank.

- **Napoved zaprtja pogodbe**, za posamezno pogodbo, poda oceno o tem, ali namerava stranka, ki ji pogodba pripada, to v prihodnosti zapreti. Zraven se izračuna še verjetnost, da se bo to zgodilo. Za tvorjenje napovedi se uporablja metode strojnega učenja, konkretno Breimanov algoritem naključnih gozdov [23]. Ta s pomočjo izbranega nabora podatkov o pogodbah zgradi napovedni model, ki se ga uporabi za napovedi zaprtja pogodb. S to napovedjo BPD dobi oceno o tem, katere stranke jih nameravajo zapustiti, in lahko pravočasno vpelje preventivne ukrepe, usmerjene k preprečitvi njihovega odhoda.

- **Razvrščanje** razdeli pogodbe v gruče, glede na njihovo medsebojno podobnost. Nad izbranim naborom vrednosti pogodb se definira mero

podobnosti in zažene izbrani algoritem za razvrščanje. Mi uporabljamo algoritem za hierarhično razvrščanje [29]. Pri tem še definiramo velikost množice gruč. S pomočjo razvrščanja BPD dobi pregled nad porazdelitvijo pogodb glede na njihove značilnosti in lahko oblikuje ukrepe ter strategije na nivoju skupine pogodb.

V diplomski nalogi bomo v drugem poglavju ovrednotili sorodne rešitve in si nato ogledali vsebinsko razlago problemske domene. V tretjem poglavju bomo na kratko predstavili tehnologije, ki so bile uporabljene pri gradnji IS, in teoretično ozadje uporabljenih metod strojnega učenja, kjer si bomo ogledali razvrščanje *(angl. clustering)* in klasifikacijo *(angl. classification)*. Naslednje poglavje, ki predstavlja osrednji del naloge, pokrije celotno zgradbo IS. Spoznali bomo, kako so podatki dolgoročno shranjeni, proces prenosa podatkov, zgoraj opisane ključne funkcionalnosti in aplikacijo za prikaz podatkov. V zadnjem poglavju končamo s sklepnimi ugotovitvami, kjer predstavimo možne izboljšave in nadgradnje IS.

# Poglavje 2

# Ovrednotenje obstoječih informacijskih sistemov in problemska domena

## 2.1 Ovrednotenje obstoječih informacijskih sistemov

V tem podpoglavju si bomo ogledali primera IS, ki sta po svojih funkcionalnostih podobna našemu. Najprej si bomo ogledali primer IS, ki nam omogoča opravljanje analiz na nivoju posameznega portfelja. Zatem bo sledil ogled primera IS, ki zna opravljati analize nad celotnimi množicami.

#### PortfolioMonkey

PortfolioMonkey [11] je IS v obliki zastonj spletne aplikacije. Omogoča nam analiziranje poljubnega števila portfeljev. Te lahko vnašamo ročno ali jih uvozimo iz datoteke oz. kakšne druge aplikacije. V nadaljevanju si bomo na kratko ogledali glavne funkcionalnosti, ki so nam na voljo v povezavi z analizo portfeljev.

Kot je razvidno s slike 2.1, nam omogoča vnos finančnih instrumentov

| Name | Symbol | Expected Return | Volatility |
|---|---|---|---|
| Verizon Communications | VZ | 13.4% | 19.1% |
| Pfizer Inc. | PFE | -0.4% | 21.2% |
| Merck & Co Inc. | MRK | 6.8% | 26.7% |
| Kraft Foods Inc. | KFT | 6.0% | 20.4% |
| CASH | | 0.0% | 0.0% |

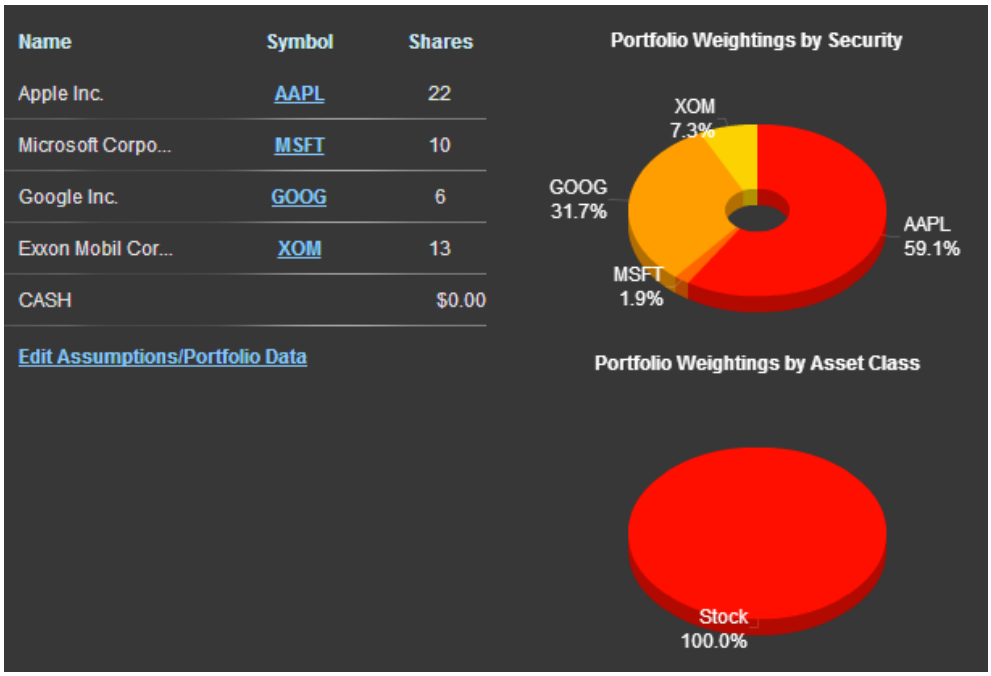
Slika 2.1: Prikaz finančnih instrumentov v portfelju z izračunanimi donosnostmi in volatilnostmi.


Slika 2.2: Prikaz donosnosti, volatilnosti in učinkovitosti na nivoju celotnega portfelja.

in računanje njihove donosnosti ter volatilnosti. Obe metriki meri na nivoju posameznega finančnega instrumenta in na nivoju celotnega portfelja, kar prikazuje slika 2.2.

Na nivoju celotnega portfelja meri še njegovo učinkovitost, ki jo definira kot razmerje med njegovo donosnostjo in volatilnostjo. Ta kriterij pomaga ovrednotiti, kakšen kompromis med donosnostjo in volatilnostjo sklepamo pri dani sestavi portfelja. Vse te analize temeljijo na gibanju vrednosti tečajev finančnih instrumentov. Žal aplikacija podpira le tiste finančne instrumente, ki kotirajo na ameriških borznih trgih, kar močno omejuje njeno uporabnost. Ker naš ciljni trg predstavlja Slovenija oz. eventualno Evropska unija, je to omejenost čutiti še toliko bolj, saj znaten delež portfeljev predstavljajo finančni instrumenti, ki kotirajo na neameriških trgih.

Za portfelj lahko za vsak finančni instrument v njem tudi vidimo količino

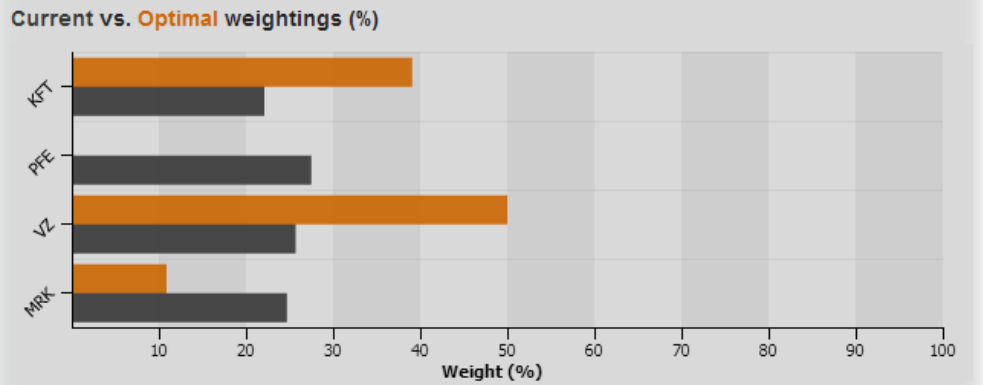



Slika 2.3: Prikaz sestave portfelja po finančnih instrumentih in tipu finančnih instrumentov.

enot oz. lotov in preračunano utež oz. delež, ki ga ima v portfelju. V obliki tortičnega diagrama so predstavljeni tudi deleži, ki jih zavzame vsak tip finančnega instrumenta v portfelju (npr. delnice, obveznice, denar). Prikaz je razviden s slike 2.3.

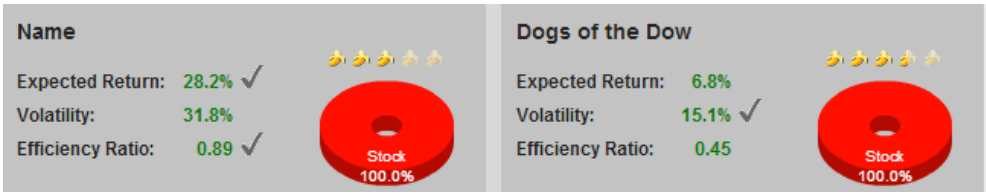
Naslednja funkcionalnost, ki jo PortfolioMonkey ponuja, je možnost optimizacije portfeljev. To v praksi pomeni, da aplikacija optimizira učinkovitost portfelja, tako da spreminja količino lotov oz. utež, ki jo posamezna pozicija predstavlja v portfelju. Primerjava uteži v trenutnem in optimiziranem portfelju je potem prikazana na grafu, skupaj s tabelo, ki napiše količino lotov, ki jo je treba dokupiti oz. prodati. Oba sta razvidna s slike 2.4. Na nivoju celotnega portfelja se tudi prikaže primerjava obeh portfeljev glede na vse tri glavne, zgoraj opisane, kazalce. Primerjava je razvidna s slike 2.5.

Prav tako nam je omogočeno nastavljanje določenih omejitev, znotraj katerih želimo, da se zgodi optimizacija. Na nivoju portfelja tako lahko

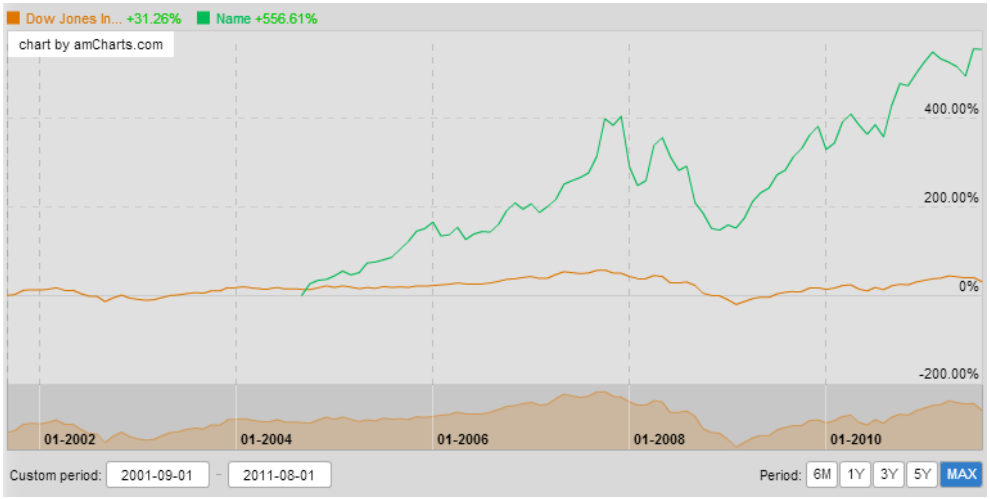



| Name | Symbol | Current Shares | Optimized Shares | Buy/Sell | Shares |
|---|---|---|---|---|---|
| Kraft Foods Inc. | KFT | 2827 | 5005 | BUY | 2178 |
| Pfizer Inc. | PFE | 5063 | 0 | SELL | 5063 |
| Verizon Communications | VZ | 2716 | 5283 | BUY | 2567 |
| Merck & Co Inc. | MRK | 2783 | 1225 | SELL | 1558 |

Slika 2.4: Prikaz optimizacije portfelja.




Slika 2.5: Prikaz primerjave portfeljev.




Slika 2.6: Prikaz grafa primerjave portfelja z indeksom.

nastavljamo maksimalno želeno volatilnost in minimalen želeni donos. Prav tako lahko za vsak finančni instrument nastavljamo razpon, znotraj katerega želimo, da leži utež njegove pozicije v optimiziranem portfelju.

Kot dodatni funkcionalnosti ponuja opravljanje primerjav donosnosti in volatilnosti med poljubnima finančnima instrumentoma ali portfeljema ter primerjavo portfelja z indeksom. Pri finančnih instrumentih se primerja njuno donosnost in volatilnost. Pri portfeljih se še dodatno primerja učinkovitost. Pri primerjavi portfelja z indeksom se primerja donosnost na letni ali mesečni ravni, v obliki grafa, ki je razviden s slike 2.6. Pri indeksih lahko izbiramo med *Dow Jones, NASDAQ Composite* in *S&P 500*.

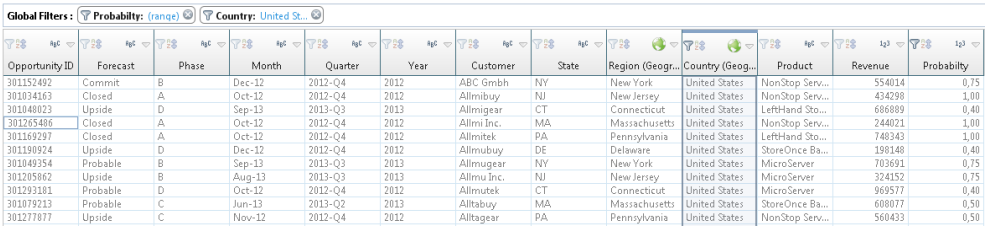
Poleg teh primerjav nam ponuja tudi oceno efekta, ki bi ga dodajanje določenega finančnega instrumenta imelo na naš portfelj. Pri tem spet izmeri vse tri ključne kazalnike in optimizira uteži, pri čemer je povsod upoštevan nov finančni instrument. Na voljo je tudi opcija, ki nam v zameno za vpis zneska, ki ga želimo investirati, priporoči finančne instrumente, za katere ocenjuje, da bi jih bilo dobro dodati v portfelj.

### SAP Predictive Analysis

*SAP Predictive Analysis* [14] je rešitev za opravljanje analiz in predikcij nad podatkovnimi množicami, podjetja *SAP*. Za razliko od našega IS ni zgrajena za specifično problemsko domeno, ampak je zamišljena kot generalna rešitev, ki jo uporabnik prilagodi za potrebe problemske domene, s katero se ukvarja. Ker med te spada tudi področje borznega posredništva, predstavlja potencialno konkurenco naši rešitvi. V nadaljevanju si bomo zato na kratko ogledali njene ključne funkcionalnosti.

Preden začnemo z analizami oz. predikcijami, moramo v aplikacijo uvoziti podatke, nad katerimi jih bomo opravljali. Tukaj nam aplikacija ponuja širok nabor različnih opcij, uvažamo lahko namreč iz: datotek *MS Excel*, datotek *CSV*, podatkovnih baz različnih tipov (*MS SQL Server, Oracle, IBM DB2, PostgreSQL, SAP HANA, Sybase, ...*) in objektov *SAP BusinessObjects Enterprise*.

Ko smo podatke uvozili, se nam za njihovo analizo ponujata dva različna tipa prikazov: v obliki tabele ali v obliki grafov. Pri tabelaričnem prikazu lahko izbiramo med prikazom v običajni tabeli in v tabeli agregatov. Najprej bomo predstavili običajno tabelo, primer katere si lahko ogledamo na sliki 2.7. Pri tem smo uporabili že vgrajeno demonstracijsko množico podatkov, kjer posamezni primer predstavlja proces prodaje nekega nedefiniranega produkta. Za posamezni primer so tako med drugim zabeleženi: verjetnost izvršitve prodaje, ali je prodaja že zaključena (v tem primeru je verjetnost enaka 100%), časovna umestitev prodaje (leto, četrtljetje, mesec) in prihodek, ki ga bo prodaja ustvarila za podjetje. Nad tabelo lahko definiramo filtre, nad katerimkoli od stolpcev. Če stolpec vsebuje številske vrednosti, nam je pri definiciji filtra na voljo določitev razpona, v katerem naj bodo filtrirane vrednosti. Če stolpec vsebuje končno diskretno množico vrednosti, kot npr. avtomobilske znamke, nam je na voljo izbor vrednosti, ki jih želimo imeti v filtrirani množici. Pri tem so vsi trenutno aktivni filtri prikazani nad tabelo, kot je prav tako razvidno s slike 2.7. Po vsakem od stolpcev lahko prav tako opravljamo sortiranje po padajočih ali naraščajočih vrednostih.



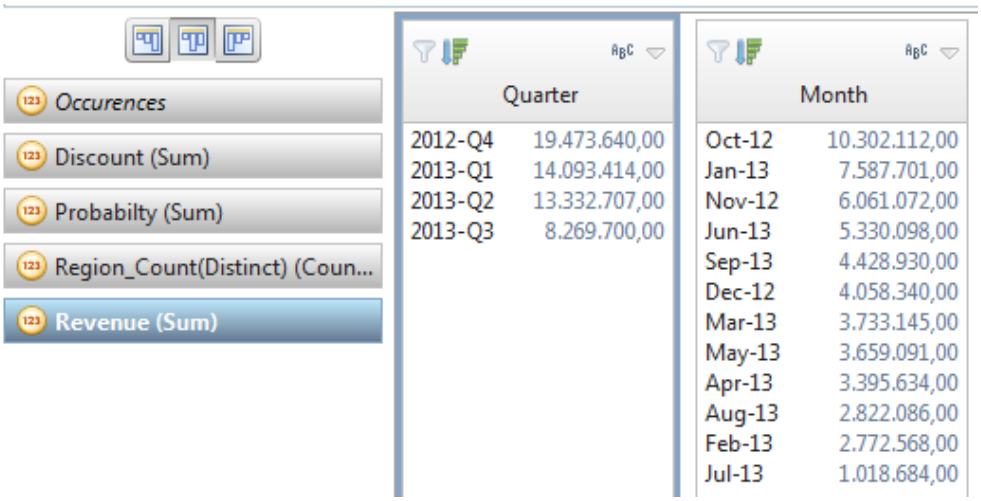
Global Filters : Probabilty: (range) Country: United St...

| Opportunity ID | Forecast | Phase | Month | Quarter | Year | Customer | State | Region (Geogr... | Country (Geog... | Product | Revenue | Probabilty |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 301152492 | Commit | B | Dec-12 | 2012-Q4 | 2012 | ABC Gmbh | NY | New York | United States | NonStop Serv... | 554014 | 0,75 |
| 301034163 | Closed | A | Oct-12 | 2012-Q4 | 2012 | Allmibuy | NJ | New Jersey | United States | NonStop Serv... | 434298 | 1,00 |
| 301048023 | Upside | D | Sep-13 | 2013-Q3 | 2013 | Allmigear | CT | Connecticut | United States | LeftHand Sto... | 686889 | 0,40 |
| 301265486 | Closed | A | Oct-12 | 2012-Q4 | 2012 | Allmi Inc. | MA | Massachusetts | United States | NonStop Serv... | 244021 | 1,00 |
| 301169297 | Closed | A | Oct-12 | 2012-Q4 | 2012 | Allmitek | PA | Pennsylvania | United States | LeftHand Sto... | 748343 | 1,00 |
| 301190924 | Upside | D | Dec-12 | 2012-Q4 | 2012 | Allmubuy | DE | Delaware | United States | StoreOnce Ba... | 198148 | 0,40 |
| 301049354 | Probable | B | Sep-13 | 2013-Q3 | 2013 | Allmugear | NY | New York | United States | MicroServer | 703691 | 0,75 |
| 301205862 | Upside | B | Aug-13 | 2013-Q3 | 2013 | Allmu Inc. | NJ | New Jersey | United States | MicroServer | 324152 | 0,75 |
| 301293181 | Probable | D | Oct-12 | 2012-Q4 | 2012 | Allmutek | CT | Connecticut | United States | MicroServer | 969577 | 0,40 |
| 301079213 | Probable | C | Jun-13 | 2013-Q2 | 2013 | Alltabuy | MA | Massachusetts | United States | StoreOnce Ba... | 608077 | 0,50 |
| 301277877 | Upside | C | Nov-12 | 2012-Q4 | 2012 | Alltagear | PA | Pennsylvania | United States | NonStop Serv... | 560433 | 0,50 |

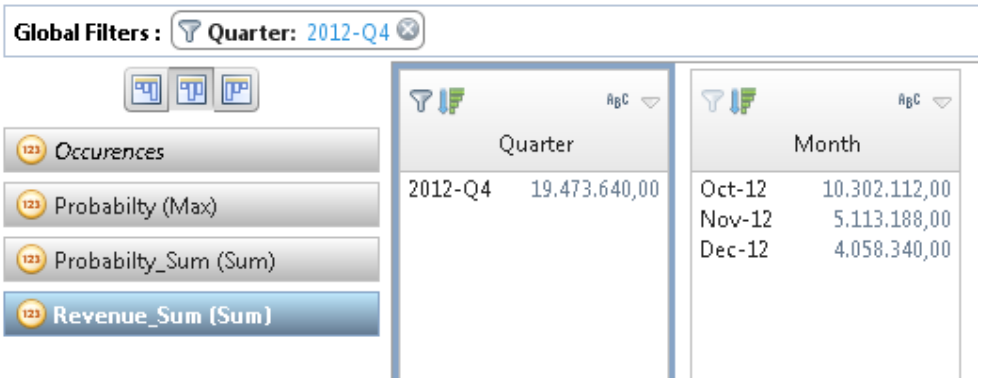
Slika 2.7: Prikaz osnovne tabele za pregled uvoženih podatkov.

Pri tabeli agregatov si podatkov ne ogledujemo po posamičnih primerih, kot pri običajni tabeli, ampak v skupinah. Pri tem določimo stolpce, ki bodo nastopali v agregatnih funkcijah, in stolpce, ki se bodo razdelili na skupine ter bodo prikazovali rezultate teh agregatnih funkcij poleg vsake skupine. Za lažjo predstavo si lahko ogledamo primer na sliki 2.8. Tukaj smo določili agregatno funkcijo sumiranja nad stolpcem za prihodek *(angl. revenue)*, medtem ko sta bila stolpca za četrtletje *(angl. quarter)* in mesec *(angl. month)* določena za prikaz po skupinah. Vsi posamezni primeri iz celotne množice, ki je prikazana v običajni tabeli na sliki 2.7, so se razdelili v skupine po četrtletju in mesecu, v katera spadajo. Poleg vsake skupine se potem prikaže prihodek, ki ga je ustvarila. Na tem mestu omenimo, da je na voljo več različnih tipov agregatnih funkcij; lahko bi npr. tudi šteli število primerov v posamezni skupini ali poiskali primer, ki je ustvaril največji prihodek. Tudi tukaj lahko filtriramo po posamezni skupini, primer česar je prikazan na sliki 2.9, kjer filtriramo po 4. četrtletju. Podatki v ostalih stolpcih se prilagodijo filtru, tako da so v tem primeru v stolpcu, ki prikazuje mesece, prikazani zgolj meseci iz 4. četrtletja.

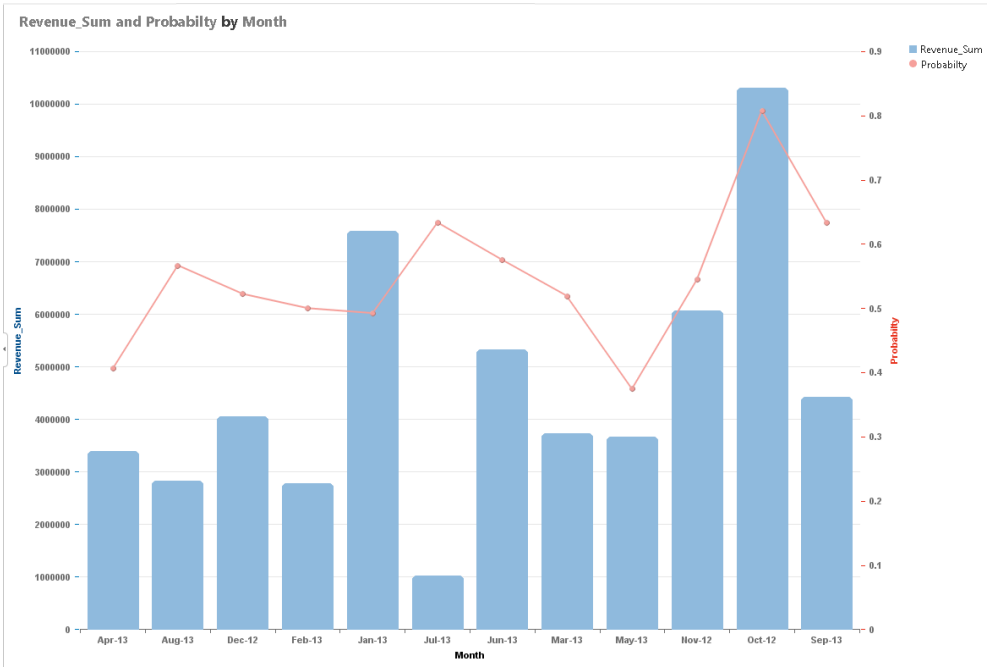
Pri prikazu podatkov v obliki grafov izbiramo med različnimi tipi grafov, kot so stolpčni graf, tortni graf, črtni graf ... Na slikah 2.10 in 2.11 si lahko ogledamo primera dveh grafov. Oba prikazujeta podatke istega podatkovnega vira, kot smo ga uporabili pri tabelaričnem prikazu. Graf na sliki 2.10 prikazuje skupne vsote prihodkov po mesecih v obliki stolpčnega grafa in povprečno verjetnost izvršitve prodaje za pripadajoči mesec v obliki črtnega grafa. Vertikalna os na levi strani označuje prihodke, na desni pa




Slika 2.8: Prikaz tabele agregatov.



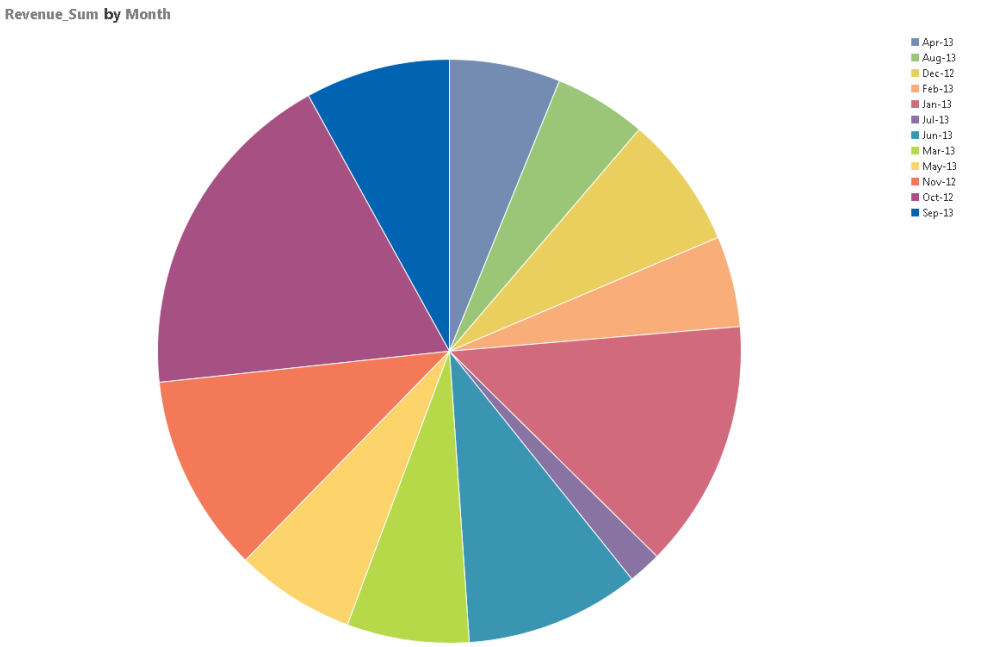

Slika 2.9: Prikaz tabele agregatov z vklopljenim filtrom po 4. četrtletju.



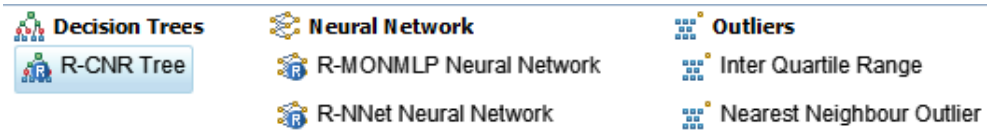

Slika 2.10: Prikaz stolpčnega in črtnega grafa, ki prikazujeta mesečne vsote prihodkov in povprečno verjetnost izvršitve prodaje.

verjetnost izvršitve prodaje. V tem primeru smo se odločili za prikaz dveh različnih podatkovnih vrst na isti površini, lahko bi prikazovali tudi vsako vrsto posamezno, vendar več kot dveh ne moremo. Primer prikaza posamezne podatkovne vrste je na sliki 2.11, kjer so prikazane mesečne vsote prihodkov v obliki tortnega grafa. Enako kot pri tabelah lahko tudi pri vseh grafih nastavljamo filtre po ostalih podatkovnih atributih, tako da prikazujejo samo podatke, ki ustrezajo postavljenim pogojem.

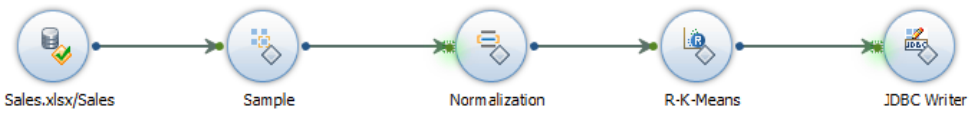
SAP Predictive Analysis ponuja tudi precejšen nabor algoritmov za predikcijo. Kot je deloma tudi razvidno s slike 2.12, imamo na voljo algoritme, kot so odločitvena drevesa, nevronske mreže in razne regresijske metode od linearne do eksponentne regresije. Med drugim podpira tudi razvrščanje (z algoritmom k-najbljižjih sosedov) in iskanje izstopajočih primerov v dani množici podatkov. Aplikacija podpira tudi različne postopke za pripravo podatkov, preden jih uporabimo kot vhodne podatke za algoritme; tako med




Slika 2.11: Prikaz tortnega grafa, ki prikazuje mesečne vsote prihodkov.




Slika 2.12: Prikaz dela celotnega nabora algoritmov in preostalih funkcionalnosti.




Slika 2.13: Prikaz celotnega procesa predpriprave podatkov, izvedbe razvrščanja in shranitve rezultatov.

drugim podpira različne metode vzorčenja, filtriranja in normalizacije podatkov. Omogočeno je tudi kreiranje novih atributov, ki so rezultat poljubne funkcije nad obstoječimi atributi. Na koncu lahko rezultate shranimo v izbrano podatkovno bazo ali datoteko. Načrtovanje celotnega procesa obdelave podatkov in izvajanja algoritmov poteka preko principa *povleci-in-spusti (angl. drag-and-drop)*. Posamezna procesiranja podatkov oz. algoritmi so predstavljeni v obliki ikon, ki jih povlečemo na delovno površino in jih medsebojno povežemo. Vrstni red povezav definira tok izvajanja celotnega procesa. Za lažjo predstavo je na sliki 2.13 prikazan primer. Najbolj leva ikona označuje izvor podatkov, nato se opravi vzorčenje nad njim, zatem normalizacija, na koncu pa se izvedeta algoritem razvrščanja in shranitev rezultatov v podatkovno bazo.

## 2.2 Problemska domena

Vse podatke za ta IS črpamo iz produkcijske podatkovne baze, ene izmed BPD, na kateri temelji IS *Shark*. Omenjen IS, produkt podjetja *Evolve d.o.o.*, nam omogoča popolno podporo borznemu trgovanju in v svoji podatkovni bazi hrani vse podatke, ki se ob tem generirajo. Konkretno ta baza vsebuje 8.068 pogodb in 262.964 transakcij, ki ustrezajo kriterijem za prenos v podatkovno bazo tega IS.

Za potrebe tega IS potrebujemo naslednje podatke:

**Finančni instrumenti in njihovi tečaji**

Finančni instrument označuje kakršnokoli sredstvo, s katerim je možno trgovati, kot so npr.: denar, obveznice, delnice, izpeljani finančni instrumenti. [17] Vsakemu finančnemu instrumentu pripada tečaj oz. njegova cena. Za potrebe tega IS se prenesejo vsi finančni instrumenti in njihovi tečaji.

**Stranke in njihove pogodbe**

Vsaka stranka BPD ima odprto eno ali več pogodb. Pogodbe se delijo

na mnogo različnih tipov, vendar sta za nas pomembna le dva: posredovanje in gospodarjenje. Kot je že iz imena razvidno, gre pri prvem tipu za pogodbe, kjer BPD služi zgolj kot posrednik za naročila, ki jih želi na borzne trge oddati stranka. Pri drugem tipu je situacija ravno nasprotna; stranka sama ne oddaja naročil, ampak BPD-ju nakaže denar, ki potem namesto njih, vendar v skladu s cilji stranke, po lastni presoji investira ta denar v razne finančne instrumente.

Pri prenosu moramo poleg še aktivnih pogodb zaobjeti tudi že zaprte pogodbe. Prenesti moramo tudi razne tabele, ki omogočajo kategorizacijo pogodb glede na določene atribute. Podrobnejši opis teh sledi v podpoglavju 4.1.

**Transakcije**

Za vsako od pogodb se vršijo razne transakcije s finančnimi instrumenti. Nas zanimajo zgolj prodaje in nakupi vrednostnih papirjev na *LJSE* in na tujih trgih. Od teh se prenesejo vse kadarkoli izvršene transakcije.

**Portfelji**

Nabor finančnih instrumentov, ki jih ima v lasti določena pogodba, tvori portfelj. Vsakemu finančnemu instrumentu v portfelju pritičeta količina, v kateri je bil nabavljen, in trenutna vrednost. Ti podatki tvorijo t. i. pozicijo. Pri prenosu portfeljev se prenesejo vse pozicije, tudi tiste, ki so v preteklosti obstajale in ne obstajajo več, ker so bili finančni instrumenti prodani.

**Provizije**

BPD vsem strankam za opravljanje storitev zaračunava provizije, ki se razlikujejo glede na tip storitve. Prenesejo se naslednje provizije:

- **Posredniške provizije:** nastanejo pri vsaki transakciji, ki jo stranka opravi.
- **Upravljavske provizije:** nastanejo pri strankah, ki imajo pogodbe tipa gospodarjenje. Zaračuna se jim provizija za upravlja-

nje njihovega portfelja.

- **Delitev dobička:** nastanejo pri strankah, ki imajo pogodbe tipa gospodarjenje. BPD stranki pobere določen delež dobička, ki ga je ustvarila z upravljanjem njenega portfelja.
- **Ležarine:** provizija, ki se zaračuna za hranjenje finančnih instrumentov.

# Poglavje 3

# Predstavitev tehnologij in metode strojnega učenja

## 3.1 Tehnologije

V tem podpoglavju si bomo na kratko ogledali ključne tehnologije, ki so bile uporabljene pri gradnji IS. V katerih delih IS so uporabljene posamezne tehnologije, je predstavljeno v sekcijah, ki opisujejo dele IS.

**SQL Server in T-SQL**

*SQL Server* je sistem za upravljanje z relacijskimi podatkovnimi bazami *(angl. relational database management system)*. Pri tem tipu podatkovnih baz so vsi podatki shranjeni v obliki tabel oz. relacij. Do podatkov se dostopa s pomočjo programskega jezika *T-SQL*. Omogoča nam povpraševanje, spreminjanje, dodajanje in brisanje podatkov ter definiranje strukture podatkovne baze. [1]

**.NET in C#**

*.NET* je programsko ogrodje *(angl. software framework)*, ki omogoča izvajanje programske kode, napisane v objektno-orientiranih programskih jezikih, kot so npr. *C#, J#, VB.NET*. Vsa koda, napisana v katerem od jezikov, ki jih ogrodje podpira, se prevede v kodo vme-

snega jezika *CIL*. Ta koda je enaka pri vseh jezikih. *CIL* se izvaja na virtualnem stroju, imenovanem *CLR*. Ta jo s pomočjo tehnike *JIT* prevaja v strojno kodo. Kot ostale pomembne funkcionalnosti ogrodja velja omeniti še upravljanje s pomnilnikom in izjemami *(angl. memory and exception management)* ter knjižnico s širokim naborom metod in razredov. [21]

**IIS in ASP.NET MVC**

*IIS* je spletni strežnik. Podpira *.NET* in omogoča izvajanje aplikacij, napisanih v kateremkoli od jezikov *.NET*.

*ASP.NET MVC* je ogrodje za gradnjo spletnih aplikacij po programskem arhitekturnem vzorcu *model-vmesnik-krmilnik (angl. model-view-controller)*, krajše *MVC*. Ta vzorec aplikacijo razdeli na tri dele:

- **Model:** skrbi za dostop do podatkovne baze. Pri tem izvaja tako bralne kot zapisovalne dostope. Prav tako skrbi za hranjenje podatkov, ki jih pridobi pri dostopih, in implementira vso potrebno poslovno logiko za obdelavo podatkov.
- **Vmesnik:** skrbi za prikazovanje podatkov uporabniku. V bistvu predstavlja uporabniško masko, implementirano v obliki spletne strani. Podatke za prikaz pridobiva iz modela.
- **Krmilnik:** skrbi za izbiro pravega vmesnika. To počne s pomočjo preslikave *URL-jev* v vmesnik. Igra tudi posrednika med vmesnikom in modelom. Vmesniku posreduje vse modele, ki jih potrebuje, in modelu posreduje vse podatke, ki jih uporabnik vnese v vmesnik (npr. vnos vrednosti v tekstovna polja).

S pomočjo tega vzorca se doseže precejšnjo mero neodvisnosti med posameznimi kompenantami. Lahko se npr. lotimo urejanja videza aplikacije, ne da moramo pri tem skrbeti za poslovno logiko. [2]

**ADO.NET Entity Framework**

Ogrodje za objektno-relacijsko preslikavo *(angl. object-relational ma-*

*pping)* znotraj *.NET-a*. Omogoča preslikavo vseh tabel in relacij podatkovne baze v objekte programskega jezika. S tem ustvari nov abstrakcijski nivo, preko katerega potekajo vsi dostopi do podatkovne baze. Na ta način se lahko tudi zamenja tehnologija, s katero je implementirana podatkovna baza, ne da je treba kakorkoli spreminjati kodo aplikacije, ki opravlja dostope. [5]

**LINQ**

Podaljšek *C#-a*, ki jeziku doda možnost pisanja in izvajanja povpraševanj ter urejanj podatkovnih zbirk, ki so implementirane kot objekti. Nas *LINQ* zanima predvsem kot sredstvo za dostop do podatkovne baze. Pri tem podatkovno zbirko predstavljajo objekti, ustvarjeni z *Entity Framework-om*. Vsa vprašanja *LINQ* so pred njihovo izvedbo na podatkovni bazi prevedena v *T-SQL*. [21]

**R in R.NET**

Programski jezik, namenjen predvsem statistični obdelavi podatkov. V njem so implementirane štiri od, v uvodu opisanih, šestih funkcionalnosti, ki jih IS nudi v okviru analize podatkov. *R.NET* je odprtokodna knjižnica, ki omogoča povezavo med *C#-om* in *R-jem*. Z njo se lahko iz *C#-ove* kode kliče funkcije *R-ja* in se jim posreduje potrebne vhodne podatke.

**HTML in CSS**

*HTML* je označevalni jezik za definiranje strukture spletnih strani. *CSS* je slogovni jezik, ki definira videz spletne strani. [8]

**JavaScript, jQuery in Flot**

*JavaScript* je skriptni jezik, ki se izvaja znotraj spletnega brskalnika. Omogoča dinamično spreminjanje vsebine spletne strani, ne da bi bilo treba dostopati do strežnika. *jQuery* je ena najbolj uporabljenih knjižnic za *JavaScript*. V širokem naboru metod implementira najpogosteje uporabljene funkcionalnosti pri gradnji spletnih strani. *Flot* je knjižnica

za *JavaScript*, ki ponuja širok nabor grafov in njihovih nastavitev za prikaz podatkov. [7, 9, 10]

## 3.2 Metode strojnega učenja

### 3.2.1 Razvrščanje

Razvrščanje se ukvarja z razvrščanjem podatkovnih točk v gruče glede na njihovo podobnost. V okviru strojnega učenja spada pod t. i. metode z nenadzorovanim učenjem. To pomeni, da podatki, ki se jih razvršča, niso označeni z razredom, ki mu pripadajo. [28]

Razvrščanje poteka tako, da se najprej izbere množico učnih primerov. Nato se izbere atribute učnih primerov, na podlagi vrednosti katerih se bo opravljalo razvrščanje, pri čemer so ti lahko diskretni ali zvezni. To se naredi avtomatično, s pomočjo temu namenjenih metod, ali se izbor opravi ročno, praviloma s strani poznavalca problemske domene, iz katere izhajajo učni primeri (v primeru te diplomske se je uporabilo zadnji način). Pri tem se teži k temu, da se izbere takšne atribute, da se bo dalo učne primere čim lepše razvrstiti v gruče oz. da bodo čim bolje odrazili strukturo, ki se skriva v množici učnih primerov. Posamezen učni primer z vrednostmi izbranih atributov si lahko predstavljamo kot točko v n-dimenzionalnem evklidskem prostoru, kjer vsaka koordinata točke predstavlja vrednost posameznega atributa.

Sledi definiranje mere podobnosti, na podlagi katere se določa medsebojna podobnost učnih primerov, ki je ključni kriterij pri razvrščanju. Podobnost med učnimi primeri se računa na podlagi vrednosti njihovih atributov. Če se pri izračunu podobnosti predpostavlja, da ima vsak atribut enakomeren vpliv na rezultat, je še pred izračunom treba vse atribute normalizirati, tako da imajo vrednosti vseh enak razpon. Za mero podobnosti se lahko izbere mere, kot so [15]:

**Evklidska razdalja**

Najkrajša razdalja med dvema točkama v n-dimenzionalnem evklid-

skem prostoru. Definirana je kot

$$d(i,j) = \sqrt{\sum_{k=1}^{n}(x_{i_k} - x_{j_k})^2}, \tag{3.1}$$

kjer $x_{i_k}$ predstavlja k-ti atribut prvega učnega primera in $x_{j_k}$ k-ti atribut drugega učnega primera.

**Kvadrirana evklidska razdalja**

Kot je že razvidno iz imena, gre tukaj zgolj za kvadrirano navadno evklidsko razdaljo. Definirana je kot

$$d(i,j) = \sum_{k=1}^{n}(x_{i_k} - x_{j_k})^2. \tag{3.2}$$

**Manhattanska razdalja**

Vsota absolutnih razlik koordinat med dvema točkama v n-dimenzionalnem evklidskem prostoru. Definirana je kot

$$d(i,j) = \sum_{k=1}^{n}|x_{i_k} - x_{j_k}|. \tag{3.3}$$

**Maksimalna razdalja**

Največja absolutna razlika koordinat med dvema točkama v n-dimenzionalnem evklidskem prostoru. Definirana je kot

$$d(i,j) = \max_{k \in n}|x_{i_k} - x_{j_k}|. \tag{3.4}$$

Na koncu se izvrši samo razvrščanje. Pri tem se uporabi eden od razvrščevalnih algoritmov, ki jih lahko grobo razdelimo na: particijske in hierarhične. Particijski algoritmi minimizirajo cenovno funkcijo oz. optimalnostni kriterij, ki vsaki razvrstitvi učnega primera v gručo pripiše neko ceno [26]. Eden bolj znanih primerov particijskih algoritmov je metoda k-najbližjih sosedov. Hierarhični algoritmi, na drugi strani, zgradijo hierarhijo gruč, v kateri velja, da se gruče s premikanjem proti vrhu hierarhije združujejo v vse večje gruče. Ker se v okvirju te diplomske naloge uporablja hierarhično razvrščanje, si bomo tega podrobneje ogledali.

### Hierarhično razvrščanje

Hierarhično razvrščanje [28] zgradi hierarhično strukturo gruč, imenovano dendrogram, prikazano na sliki 3.1. Z dendrograma je razvidno, iz katerih gruč na prejšnjem nivoju je tvorjena gruča na trenutnem nivoju dendrograma. Na horizontalni osi so predstavljeni posamezni primeri, ki na prvem nivoju tvorijo vsak svojo gručo. Vertikalna os na drugi strani označuje podobnost med paroma združenih gruč. Število gruč, ki jih želimo na koncu dobiti, določimo z višino prereza dendrograma, kar je prav tako razvidno s slike 3.1. Algoritme za hierarhično razvrščanje delimo na združevalne in delitvene. Združevalni začnejo z vsakim učnim primerom v svoji gruči in po nivojih dendrograma paroma združujejo dve najbolj podobni gruči, pri čemer je podobnost definirana z izbrano mero podobnosti gruč. Delitveni na drugi strani začnejo z eno gručo, v kateri so vsi učni primeri, in jo rekurzivno delijo na manjše gruče. Mi uporabljamo združevalni algoritem. Pri merah podobnosti lahko izbiramo med več različnimi opcijami, pri čemer med najbolj uporabljene sodijo [33]:

**Mera polne bližine**

Razdalja med gručama $A$ in $B$ je definirana kot maksimalna razdalja, med dvema primeroma, kjer je en primer iz gruče $A$ in drugi iz gruče $B$. Definirana je kot

$$d(A, B) = \max_{\vec{x} \in A, \vec{y} \in B} ||\vec{x} - \vec{y}||. \tag{3.5}$$

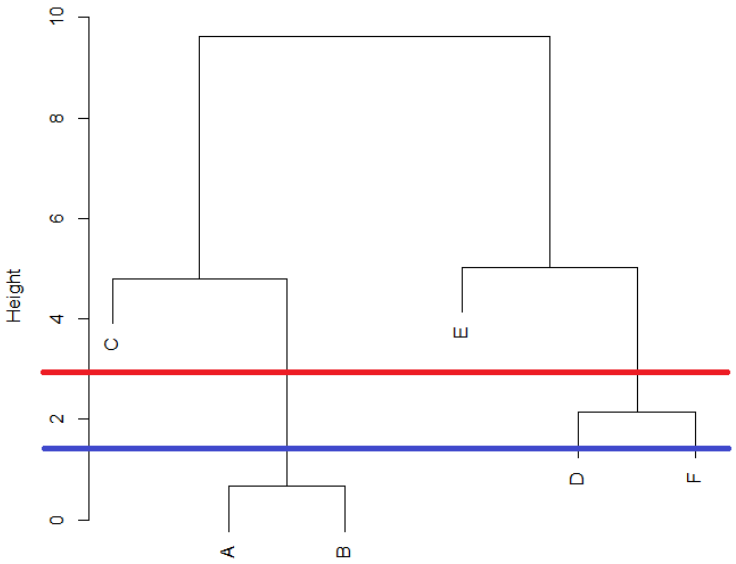
**Mera minimalne bližine**

Razdalja med gručama $A$ in $B$, definirana kot minimalna razdalja, med dvema primeroma, kjer je en primer iz gruče $A$ in drugi iz gruče $B$. Definirana je kot

$$d(A, B) = \min_{\vec{x} \in A, \vec{y} \in B} ||\vec{x} - \vec{y}||. \tag{3.6}$$

**Wardova metoda**

Razdalja med gručama $A$ in $B$ je definirana kot povečanje vsote kvadratov napake po združitvi obeh gruč. Cilj je minimizirati vsoto kvadratov




Slika 3.1: Primer dendrograma nad množico $\{A, B, C, D, E, F\}$. Prikazani sta dve opciji prereza: če ga naredimo na višini, kjer je rdeča črta, dobimo gruče $\{A, B\}, \{C\}, \{D, F\}, \{E\}$, če ga naredimo na višini modre črte, dobimo gruče $\{A, B\}, \{C\}, \{D\}, \{E\}, \{F\}$.

napake. Definirana je kot

$$\Delta(A, B) = \sum_{i \in A \cup B} ||\vec{x_i} - m_{\vec{A \cup B}}||^2 - \sum_{i \in A} ||\vec{x_i} - \vec{m_A}||^2 - \sum_{i \in B} ||\vec{x_i} - \vec{m_B}||^2, \tag{3.7}$$

kjer $\vec{x_i}$ označuje posamezni primer in $\vec{m_j}$ center gruče j.

Na koncu je treba še določiti število gruč. To lahko določi poznavalec domene ali se določi s pomočjo algoritma. Mi uporabimo zadnjo opcijo, in sicer t. i. *L-metodo*.

*L-metoda* [32] deluje tako, da išče koleno kriterijske funkcije. Pri grafu kriterijske funkcije je na horizontalni osi prikazano število gruč, na vertikalni pa vrednost razdalje pri danem številu gruč. Vrednost razdalje je obratno premo sorazmerna vrednosti podobnosti pri izbrani meri podobnosti. Vrednost podobnosti je enaka podobnosti dveh najbolj podobnih gruč pri danem

številu gruč. Na horizontalni osi se vrednosti, od desne proti levi, raztezajo od začetnega števila gruč pred prvim združevanjem (v našem primeru je na začetku število gruč enako številu primerov) do števila gruč pred zadnjim združevanjem.

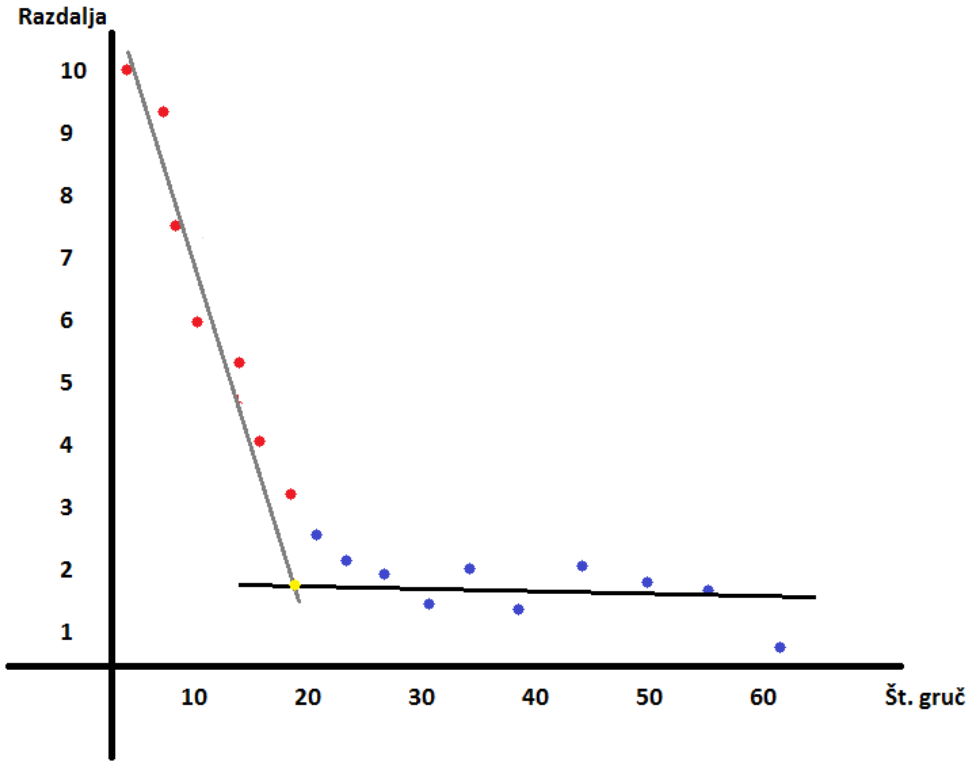
Celoten graf je prikazan na sliki 3.2. Kot je razvidno, ima funkcija ukrivljeno prehodno območje oz. koleno, ki funkcijo deli na dve območji: na desni strani kolena so si gruče medsebojno precej podobne, medtem ko se na levi strani medsebojna razdalja začne strmo vzpenjati. V obeh območjih se funkcija obnaša približno linearno.

*L-metoda* za iskanje optimalnega števila gruč izkorišča ravno to lastnost funkcije. Vsako od obeh območij namreč lahko aproksimiramo s pomočjo premice, pri čemer bo njuno presečišče v območju kolena. Obe premici in njuno presečišče so razvidni s slike 3.2.

Metoda kot končno število gruč vrne vrednost iz kolena. Na ta način zaobjamemo največ združitev gruč, preden se začne rušiti njihova notranja homogenost, in sicer zaradi povečanja medsebojnih razdalj gruč, iz katerih sestojijo. V nadaljevanju si bomo ogledali formalni opis *L-metode*.

Označimo horizontalno os kot x-os, na kateri se število gruč giblje od $x = 2, ..., n$, tako da je število vseh točk enako $n - 1$. Izberimo tudi točko $x = p$, pri čemer velja $p = 3, ...n - 2$. Ta točka razdeli množico vseh točk na dve množici: $L_p$ in $R_p$. Množica $L_p$ zaobjame $m$ točk od $x = 2, ..., p$ in množica $D_p$ $k$ točk od $x = p + 1, ..., n - 1$. Vsako od množic točk aproksimiramo s premico, ki se množici najbolje prilega. Premici najdemo s pomočjo metode najmanjših kvadratov *(angl. method of least squares)* [19]. Skupno napako pri prileganju premic za $x = p$ definiramo s pomočjo korena srednje kvadratne napake *(angl. root mean square error)* [13]. Definirana je kot

$$RMSE_p = \frac{m}{n-1} * RMSE_{L_p} + \frac{k}{n-1} * RMSE_{D_p}, \tag{3.8}$$




Slika 3.2: Primer kriterijske funkcijske. Prikazani so tudi leva in desna aproksimacijska premica ter njuno presečišče. Leva premica (označena s sivo barvo) zaobjame rdeče točke, desna (označena s črno barvo) zaobjame modre točke.

kjer $RMSE_{L_p}$ in $RMSE_{D_p}$ označujeta koren srednje kvadratne napake za levo in desno premico. Pri obeh premicah je ta definirana kot

$$RMSE = \sqrt{\frac{\sum_{i=1}^{t}(y_i - y_i^A)^2}{t}}, \tag{3.9}$$

kjer $t$ označuje število točk, ki jih zaobjame posamezna premica, $y_i$ vrednost dejanske točke na y-osi in $y_i^A$ vrednost aproksimacijske točke za dano vrednost $i$. Iščemo takšno vrednost $p = p'$, pri kateri je $RMSE_p$ minimalen. Vrednost $p'$ je v kolenu in je izbrana za število gruč, ki ga vrne metoda kot rešitev.

Pri grafih z več tisoč točkami se pojavi problem, da je na desni strani kolena neprimerno več točk kot na njegovi levi strani. *L-metoda* namreč deluje najbolje, ko obe premici zaobjameta približno enako število točk. Neuravnoteženost ima za posledico to, da locira koleno pri večjem številu gruč, kot se dejansko nahaja. Problem rešujemo z iterativnim krčenjem množice točk, ki jih upošteva *L-metoda*.

Najprej L-metodo zaženemo nad vsemi točkami. Ko nam ta vrne vrednost $p'$, v naslednji iteraciji zaženemo *L-metodo* nad prvih $2 * p'$ točk (gledano iz smeri izhodišča grafa). To ponavljamo, dokler je vrednost $p'$, ki je vrnjena v trenutni iteraciji, manjša od vrednosti $p'$ iz prejšnje iteracije. Ko ta pogoj več ne drži, se ustavimo in vrnemo vrednost $p'$. Na ta način iterativno krčimo množico točk, ki jo upoštevamo pri izvajanju *L-metode*. Z vsako iteracijo dobivamo boljši rezultat, saj sta premici vedno bolj uravnoteženi. Tako lociramo koleno pri dejanskem številu gruč.

### 3.2.2 Klasifikacija

Klasifikacija se ukvarja s klasificiranjem podatkov v razrede. Vsak podatek sestoji iz atributov in razreda, ki mu pripada. Razrede predstavlja diskretna spremenljivka s končnim naborom vrednosti, medtem ko atributi lahko hranijo zvezne ali diskretne vrednosti. [28, 29] Mi se bomo osredotočili zgolj na primer klasifikacije, pri kateri imamo dva razreda. Cilj klasifikacije je ugotoviti način, kako iz vrednosti atributa napovedati razred, ki mu podatek

pripada. Ker so podatki označeni z razredom, ki mu pripadajo, klasifikacija v okviru strojnega učenja spada med metode z nadzorovanim učenjem. [29]

Klasifikacija poteka tako, da najprej izberemo učno množico podatkov, nad katero se bo zgradil klasifikator (sredstvo, s katerim se napoveduje razrede nad novimi podatki) in validacijsko množico, s pomočjo katere se bo ocenila kakovost klasifikatorja. Močno je zaželeno, da se primeri v obeh množicah ne prekrivajo, saj je tako ocena bolj zanesljiva. Sledi izbor atributov, nad katerimi se bo opravljalo klasifikacijo, in izbira načina določitve razreda. To izvedemo na enega od načinov, ki sta bila predstavljena pri razvrščanju. V našem primeru je izbor atributov in način določitve razreda opravil poznavalec problemske domene. Na koncu se opravi izbor algoritma, s katerim se bo zgradilo klasifikator, kjer izbiramo med algoritmi, kot so npr. naivni bayes, metoda podpornih vektorjev, naključni gozdovi. Izbor lahko opravi izkušen poznavalec, ki upošteva lastnosti podatkov, s katerimi imamo opravka, ali izberemo algoritem s pomočjo preizkušanja, kjer preverjamo zadostitev izbrani metriki kakovosti klasifikatorja. Pri nekaterih algoritmih je kakovost klasifikatorja odvisna od določenih parametrov, ki jih uporabnik lahko nastavlja, in s pomočjo izbrane metrike lahko kalibriramo tudi njihovo vrednost. Običajno se za to metriko izbere natančnost klasifikatorja, ki predstavlja delež napovedi, kjer se napovedani razred sklada z dejanskim razredom glede na vse napovedi, ki jih je opravil klasifikator. Definirana je z enačbo [18]

$$Natancnost = \frac{TP + TN}{TP + FP + TN + FN}, \tag{3.10}$$

kjer $TP$ predstavlja pravilno napovedane pozitivne razrede (en razred označimo kot pozitiven, drugega kot negativen), $TN$ pravilno napovedane negativne razrede in $FP$, $FN$ napačno napovedane pozitivne ter negativne razrede. Natančnost se oceni nad validacijsko množico. Metod za izbiro validacijske množice je več, naštejmo le nekaj primerov [29]:

- Dve tretjini podatkov iz celotne množice se vzame za učno množico, nad katero se zgradi klasifikator. Preostala tretjina tvori validacijsko


Slika 3.3: Prikaz 4-kratne prečne validacije.

množico.

- Celotno množico se razdeli na $k$ enako velikih podmnožic, kjer je k poljubno celo število. Ena podmnožica se izbere za validacijsko, unija preostalih za učno množico. To se ponovi $k - krat$, tako da je na koncu vsaka podmnožica izbrana za validacijsko. Pri vsaki delitvi se zgradi klasifikator in oceni njegova natančnost. Končna natančnost je definirana kot povprečna natančnost preko vseh delitev. Ta metoda je znana kot prečna validacija. Shematski prikaz je razviden s slike 3.3.

- Ta metoda, imenovana izpusti enega, je posebni primer prečne validacije, kjer je $k$ enak številu vseh podatkov v celotni učni množici. Njen nadaljnji potek je potem enak kot pri običajni prečni validaciji.

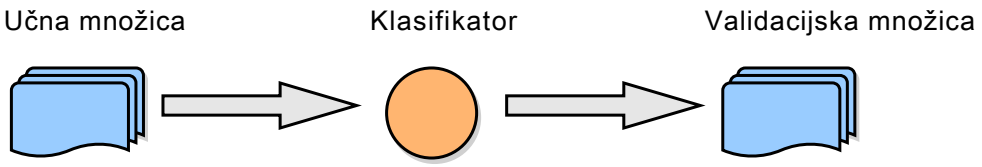
Predvsem pri podatkih, kjer je distribucija razredov neenakomerna, še je koristno izmeriti meri občutljivosti in specifičnosti [30]. Občutljivost nam pove delež pravilno napovedanih pozitivnih razredov. Definirana je kot

$$Obcutljivost = \frac{TP}{TP + FN}. \tag{3.11}$$

Na drugi strani nam specifičnost pove delež pravilno napovedanih negativnih razredov. Definirana je kot

$$Specificnost = \frac{TN}{TN + FP}. \tag{3.12}$$

Ko je algoritem izbran, se končen klasifikator lahko zgradi tudi nad celotno množico, vključno z validacijsko. Na ta način se lahko algoritem uči na dodatnih primerih in lahko zgradi še boljši klasifikator. Za lažjo predstavo je




Slika 3.4: Prikaz gradnje in preizkušanja klasifikatorja.

celoten postopek predstavljen na sliki 3.4. S končnim klasifikatorjem potem lahko napovedujemo razrede na novih, še ne označenih podatkih. Spodaj si bomo podrobneje ogledali naključne gozdove, saj so se ti izkazali kot algoritem z visoko natančnostjo nad našo problemsko domeno.

### Naključni gozdovi

Naključni gozdovi [22, 23] sestojijo iz množice odločitvenih dreves, kjer vsako odločitveno drevo nov podatek klasificira v določen razred. Pravimo, da drevo glasuje za ta razred, in na koncu se podatek klasificira v tisti razred, ki je v celotni množici dreves prejel največ glasov. Med najbolj znane algoritme naključnih gozdov spada Breimanov algoritem, ki si ga bomo podrobneje ogledali, saj v našem IS za napovedi zaprtij pogodb uporabljamo ravno tega.

Posamezno odločitveno drevo se zgradi tako, da se iz množice vseh podatkov, velikosti $n$, s pomočjo vzorčenja z zamenjavo izbere $n$ učnih primerov. Pri tem načinu izbire učne množice izberemo približno dve tretjini primerov, preostali neizbrani primeri tvorijo validacijsko množico, imenovano množica *out-of-bag(OOB)*. Na ta način že sam algoritem naključnih gozdov vključuje tvorbo ločene validacijske množice in ni potrebe po tem, da jo mi ustvarjamo ročno. Razmerje med napačno klasificiranimi in vsemi primeri v množici *OOB* tvori oceno napake *OOB (angl. OOB error estimate)*. Ta nam v bistvu pove delež napačno klasificiranih primerov. Povprečje te metrike nad vsemi drevesi tvori oceno napake *OOB* naključnega gozda in iz nje je trivialno pridobiti natančnost klasifikatorja.

Zatem se iz množice vseh atributov, velikosti $M$, pri vsakem vozlišču

naključno izbere podmnožico atributov, velikosti $m$ (ta vrednost je enaka pri vseh drevesih), kjer velja $m << M$ [23]. Iz nje se izbira atribut, ki bo tvoril vozlišče. Izbor najboljšega atributa [34] se opravlja s pomočjo *Ginijevega indeksa*. Definiran je kot

$$Gini = 1 - \sum_{i=1}^{k} p_i^2, \tag{3.13}$$

kjer $k$ označuje število vseh razredov in $p_i$ relativno frekvenco posameznega razreda v vozlišču. Vrednost indeksa pade v interval med 0 in 1, pri čemer nižja vrednost pomeni večjo homogenost vozlišča, kar se tiče distribucije vrednosti razredov v njem. Izbere se tisti atribut, pri katerem enačba

$$\Delta = Gini_p - \sum_{i=1}^{c} \frac{N_i}{N_p} * Gini_i \tag{3.14}$$

doseže največjo vrednost. Spremenljivka $Gini_p$ označuje vrednost *Ginijevega indeksa* v vozlišču, ki ga cepimo, $c$ število vrednosti cepitvenega atributa, $N_p$ število primerov v tem vozlišču, $N_i$ število primerov v $i$-tem otroku tega vozlišča in $Gini_i$ vrednost indeksa v $i$-tem otroku. Omenimo še, da je vsako drevo zgrajeno do maksimalne velikosti in se ne reže.

Algoritem je najbolj občutljiv na vrednost parametra $m$, medtem ko $n$ ni problematičen, dokler ga nastavimo na dovolj veliko število dreves; npr. Breiman [23] pri učni množici z 20 atributi in 1000 primeri uporabi 500 dreves. Za vrednost $m$ se priporoča preizkusiti $\sqrt{M}$, njegovo polovično in podvojeno vrednost [24], vendar je lahko optimalna vrednost drugje.

# Poglavje 4

# Informacijski sistem za opravljanje analiz in predikcij

Arhitektura IS se deli na štiri funkcionalnostne sklope, in sicer:

**Podatkovna shramba**

Predstavlja prostor za dolgoročno shrambo vseh podatkov IS. Iz nje se berejo in vanjo se shranjujejo vsi podatki, ki jih generirajo preostali arhitekturni sklopi IS. Podatkovna shramba je realizirana v obliki dveh podatkovnih baz, imenovanih učna in živa podatkovna baza, ki imata različno vsebino in identično podatkovno shemo, ki jo definira podatkovni model. Učna podatkovna baza služi potrebam gradnje modela za napovedovanje zaprtij pogodb, živa je namenja razvrščanju pogodb in vsem izračunom, ki se vršijo pred zagonom algoritma za razvrščanje. Obe sta implementirani s pomočjo *Microsoft SQL Serverja 2008*.

**Prenos podatkov**

Sklop, v okviru katerega se prenesejo podatki iz podatkovne baze IS *Shark* v podatkovno shrambo našega IS. Prenos podatkov je implementiran v obliki dveh *.NET* procesov: *ClusteringScheduler* in *PredictionScheduler*. Prvi skrbi za prenos podatkov v živo podatkovno bazo, drugi v učno podatkovno bazo. Vsak od procesov med izvajanjem kliče

procedure *SQL*, ki izvedejo prenos podatkov. Oba procesa se prožita periodično, vsak s svojo periodo.

**Izračuni in algoritmi**

Sklop, v okviru katerega se izvrši algoritem za razvrščanje pogodb in gradnja modela za napovedovanje zaprtij pogodb. Nad množico pogodb, ki se grupirajo, se še pred zagonom algoritma za razvrščanje opravijo izračuni volatilnosti, življenjske vrednosti pogodbe in verjetnosti zaprtja pogodbe ter pripadajoča napoved, ali se bo pogodba zaprla. Izračuni in algoritmi so implementirani v sklopu obeh procesov pri prenosu podatkov. *ClusteringScheduler* izvaja razvrščanja pogodb in vse prej navedene izračune, *PredictionScheduler* izvaja gradnjo modela za napovedovanje zaprtij pogodb.

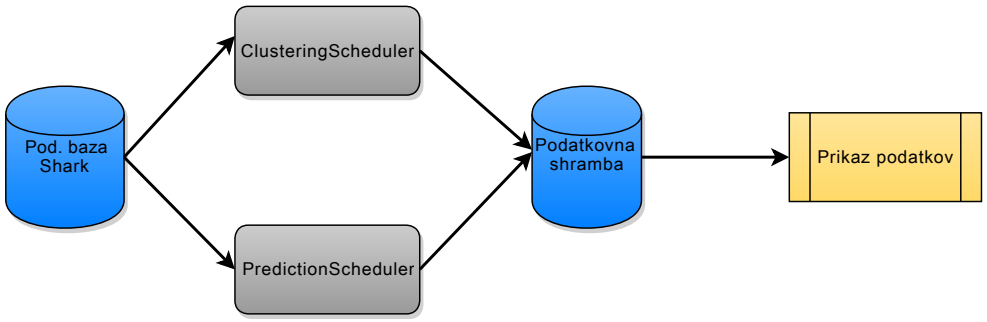
**Prikaz podatkov**

Sklop, v okviru katerega se vsi podatki prikažejo v uporabniku prijazni obliki. Za prikaz podatkov skrbi spletna aplikacija, do uporabniškega vmesnika katere se dostopa preko spletnega brskalnika. Uporabniški vmesnik s pomočjo različnih grafičnih prikazov uporabniku predstavi relevantne podatke, ki so bili pripravljeni v prejšnjih dveh sklopih. Osrednjo tehnologijo, okrog katere je zgrajena spletna aplikacija, predstavlja *ASP.NET MVC*.

Vse zgoraj opisane bistvene komponente arhitekture IS so zaradi jasnejše predstave prikazane na sliki 4.1.

## 4.1 Podatkovna shramba

Postavitev podatkovne shrambe začnemo z izdelavo podatkovnega modela. Vloga podatkovnega modela je, da ustvari podatkovno shemo za obe podatkovni bazi. Ker obe podatkovni bazi v vseh produkcijskih okoljih temeljita na enaki tehnologiji, ni nobene potrebe po izdelavi konceptualnega podatkovnega modela in lahko podatkovni model postavimo neposredno na nivoju




Slika 4.1: Prikaz arhitekture IS.

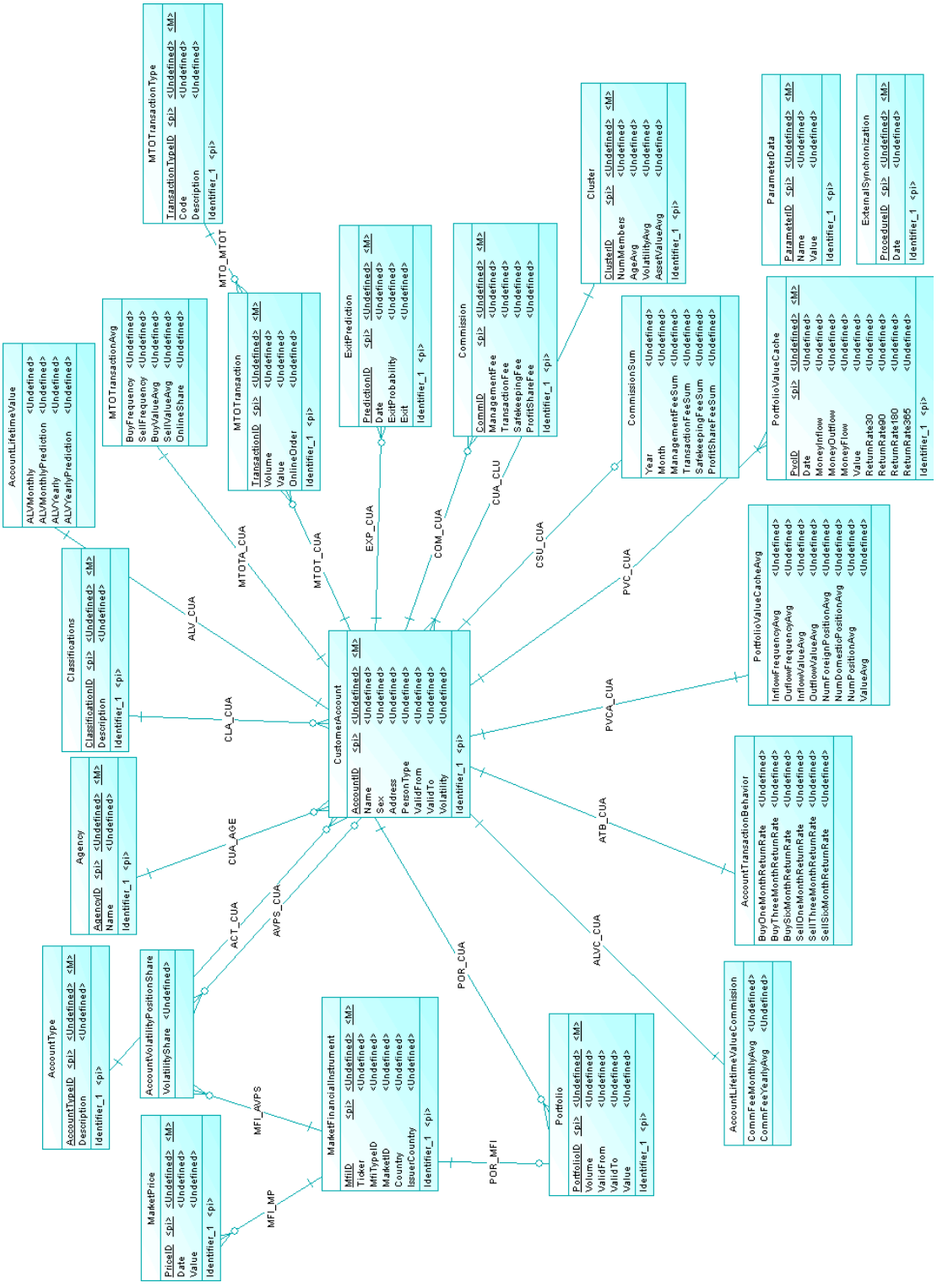
fizičnega podatkovnega modela. Ogledamo si ga lahko na sliki 4.2. Zaradi performančne optimizacije aplikacije so tabele denormalizirane in se tako posamezni podatek pojavlja v več tabelah (npr. trgovalna oznaka finančnega instrumenta). Na ta način odpade mnogokatera potreba po združevanju tabel, kar pri dani količini poizvedb in obsegu podatkov predstavlja znatno pohitritev poizvedb nad podatkovno bazo.

Glede na vlogo, ki jo igrajo pri hranjenju podatkov, se tabele iz podatkovnega modela razdelijo na naslednje skupine:

**Šifranti**

V to skupino spadajo vse tabele, ki imajo za nalogo hranjenje nabora tipov, na katere se delijo instance posamezne kategorije podatkov. Če na primer pogledamo transakcije, ki jih je opravila posamezna pogodba, se vsaka transakcija uvršča v enega izmed naslednjih tipov glede na borzni trg, na katerem kotira finančni instrument, ki je predmet transakcije, in glede na to, ali gre za prodajo ali nakup omenjenega finančnega instrumenta:

- Nakup LJSE
- Prodaja LJSE
- Nakup tuji trg
- Prodaja tuji trg




Slika 4.2: Prikaz podatkovnega modela.

**Pogodbe strank**

V to skupino se uvršča ena sama tabela, katere naloga je, da hrani vse tiste podatke o pogodbah strank, ki niso v razmerju "ena proti mnogo" s posamezno pogodbo. Na primer: hrani tip pogodbe (posredovanje/gospodarjenje), ki ga ima vsaka pogodba samo enega, ampak ne hrani transakcij, ki jih je ta pogodba izvršila, saj je teh lahko več na posamezno pogodbo.

Tabela igra centralno vlogo v podatkovnem modelu, saj se vse tabele iz skupine *Viri za agregate* navezujejo nanjo, na katere se potem navezujejo tabele skupine *Agregati*. Iz tega razloga jo je smiselno uvrstiti v lastno skupino.

**Viri za agregate**

V to skupino spadajo vse tiste tabele, ki beležijo podatke, ki se nanašajo na aktivnosti posameznih pogodb. Primera tabel, ki spadajo v to skupino, sta tabeli, ki beležita transakcije, ki jih je opravila posamezna pogodba, in vse provizije, ki jih je ta pogodba s svojimi aktivnostmi ustvarila za BPD.

**Agregati**

V to skupino spadajo tabele, ki za posamezno pogodbo hranijo agregirane vrednosti podatkov tabel skupine *Viri za agregate*. Obseg obdobja, v katerem se vrši izračun agregatov, se med tabelami razlikuje. Primeri agregiranih podatkov so: število izvršenih prodaj, število izvršenih nakupov, povprečna vrednost izvršenih prodaj, povprečna vrednost izvršenih nakupov ...

**Rezultati algoritmov**

V to skupino spadajo tabele, ki hranijo rezultate algoritmov, ki se uporabljajo v aplikaciji. Ti algoritmi, kot svoj vhod, lahko tudi uporabljajo podatke iz prejšnje skupine. Primer takšnih rezultatov predstavljajo verjetnosti zaprtja posameznih pogodb in njihova volatilnost.

**Sistemske tabele**

V to skupino spadajo tabele, ki hranijo podatke, ki služijo kot nastavitve za pravilno delovanje aplikacije. Primeri tovrstnih podatkov so datumi zadnjega zagona procedur in razni parametri, ki služijo kot vhod algoritmom, uporabljenim v aplikaciji.

V naslednjih sekcijah sledijo opisi vsebinskega pomena vseh tabel vsake od zgoraj omenjenih skupin.

### 4.1.1 Šifranti

**AccountType**

**Opis:** podatki o vseh tipih pogodb. Vsaka pogodba se uvršča v enega izmed dveh tipov: posredovanje ali gospodarjenje. Vsebinski pomen obeh tipov je že bil opisan v podpoglavju 4.1, zato ga tu ne bomo navajali.

**Agency**

**Opis:** podatki o vseh poslovalnicah BPD. Vsaka pogodba je sklenjena pri eni izmed poslovalnic.

**Classifications**

**Opis:** podatki o vseh kategorijah, v katere se lahko stranke oz. njihove pogodbe klasificira glede na njihovo tveganje za pranje denarja in vse kategorije glede na njihovo izkušenost pri trgovanju s finančnimi instrumenti.

**MarketFinancialInstrument**

**Opis:** podatki o vseh finančnih instrumentih, s katerimi se trguje. Vsak finančni instrument lahko kotira na več trgih in vsaka kotacija predstavlja svoj vhod v tabeli. Med pomembnejše podatke, ki opisujejo posamezen finančni instrument, sodijo: oznaka, pod katero se trguje z njim, država trga, na katerem kotira, država izdajatelja in tip instrumenta (delnica, obveznica, izpeljani finančni instrument ...).

**MarketPrice**

**Opis:** tečaji za vsakega od finančnih instrumentov. Za vsak trgovalni dan ima vsak finančni instrument en vhod v tabeli. Tečaji predstavljajo zaključne cene finančnih instrumentov na koncu trgovalnega dne. Koliko dni nazaj segajo tečaji, se med posameznimi primeri razlikuje.

**MTOTransactionType**

**Opis:** opisi vseh tipov transakcij. Ker je vsebinski pomen teh tipov, že razložen kot primer pri opisu skupine *Šifranti* v podpoglavju 4.2, ga tu ne bomo navajali.

### 4.1.2 Pogodbe strank

**CustomerAccount**

**Opis:** podatki o strankah in njihovih pogodbah, ki jih imajo sklenjene pri BPD. Vsaka stranka ima lahko sklenjenih več pogodb. Vsaka pogodba je svoj vhod v tabeli. Med pomembnejše podatke, ki opišejo stranko, spadajo: ime, priimek, spol, starost, kraj bivanja in tip osebe (fizična ali pravna). Pomembnejši podatki pri opisu pogodbe so: datum odprtja, datum zaprtja (če je pogodba že bila zaprta), poslovalnica, pri kateri je bila pogodba sklenjena, in tip pogodbe.

### 4.1.3 Viri za agregate

**MTOTransaction**

**Opis:** podatki o vseh transakcijah na borznih trgih, ki so bile opravljene v okviru posamezne pogodbe. Vsaka transakcija posamezne pogodbe je svoj vhod v tabeli. Pomembnejša polja so: tip transakcije; oznaka finančnega instrumenta, ki je predmet transakcije; količina lotov; skupna vrednost transakcije v valuti trga, na katerem finančni instrument kotira, in evrski valuti ter podatek o tem, ali je bilo naročilo oddano preko spletnega vmesnika (stranke BPD lahko naročila za tran-

sakcije s finančnimi instrumenti oddajajo same, neposredno preko spletnega vmesnika ali posredno preko svojega borznega posrednika).

**Commission**

**Opis:** podatki o vseh provizijah, ki jih BPD zaračuna za svoje storitve posamezni pogodbi. Vsaka obračun provizij pri posamezni pogodbi je svoj vhod v tabeli. Pomembnejša polja so: datum obračuna provizije, vrednost posredniške provizije, vrednost ležarine, vrednost upravljavske provizije, vrednost provizije pri deljenju dobička.

**Portfolio**

**Opis:** podatki o vseh pozicijah posamezne pogodbe. Vsaka pozicija pri posamezni pogodbi predstavlja svoj vhod v tabeli. Pomembnejša polja so: oznaka finančnega instrumenta, količina lotov, obdobje veljavnosti pozicije (definirata ga datuma nakupa in prodaje finančnega instrumenta), vrednost pozicije v valuti trga, na katerem kotira finančni instrument, in evrski valuti.

**PortfolioValueCache**

**Opis:** podatki o vrednosti, denarnih prilivih in odlivih ter donosnosti portfelja posamezne pogodbe. Vse vrednosti se izračunajo za vsak dan pri posamezni pogodbi in so tako svoj vhod v tabeli. Pomembnejša polja so: vrednost portfelja, vsota denarnih prilivov, vsota denarnih odlivov, bilanca prilivov in odlivov, donosnost portfelja (ta se računa za 30, 90, 180 in 365 dni dolga obdobja).

### 4.1.4 Agregati

**MTOTransactionAvg**

**Opis:** agregati nad podatki o transakcijah pri posamezni pogodbi za preteklo leto dni (gledano od datuma izračuna). Vsaka pogodba ima en vhod v tabeli. Pomembnejša polja so: število nakupov, število prodaj, povprečna vrednost nakupa, povprečna vrednost prodaje, delež transakcij, izvedenih preko spletnega vmesnika.

**CommissionSum**

**Opis:** ločene vsote provizij pri posamezni pogodbi za preteklo leto dni (gledano od datuma izračuna). Provizije se sumirajo na mesečni ravni in je tako pri posamezni pogodbi za vsak mesec svoj vhod v tabeli. Pomembnejša polja so: vsota posredniških provizij, vsota upravljavskih provizij, vsota ležarin, vsota vrednosti pri delitvi dobička.

**AccountLifetimeValueCommission**

**Opis:** agregati nad podatki o provizijah pri posamezni pogodbi. Vsaka pogodba ima en vhod v tabeli. Pomembnejši polji sta: povprečna mesečna vsota provizij (šteje se vse vrste provizij) za preteklo leto dni in povprečna letna vsota provizij za zadnja tri leta.

**PortfolioValueCacheAvg**

**Opis:** agregati nad podatki tabel *Portfolio* in *PortfolioValueCache* za preteklo leto dni. Vsaka pogodba ima en vhod v tabeli. Pomembnejša polja so: število denarnih prilivov, število denarnih odlivov, povprečna vrednost denarnih prilivov, povprečna vrednost denarnih odlivov, povprečno število pozicij, povprečno število pozicij s tujimi finančnimi instrumenti, povprečno število pozicij z domačimi finančnimi instrumenti, povprečna vrednost portfelja.

### 4.1.5 Rezultati algoritmov

**ExitPrediction**

Podatki o napovedih zaprtij pogodb. Vsaka pogodba ima lahko v tabeli več vhodov, saj se zanjo hranijo vse pretekle napovedi. Pomembnejša polja so: datum napovedi, verjetnost zaprtja in podatek o tem, ali je za pogodbo napovedano zaprtje (pogodbo se napove kot zaprto, če je verjetnost zaprtja nad določeno mejo).

**AccountLifetimeValue**

Podatki o pričakovanih prihodkih pogodb. Vsaka pogodba ima v tabeli

en vhod. Pomembnejša polja so: pričakovani prihodki pri upoštevanju povprečnih mesečnih provizij in pri upoštevanju povprečnih letnih provizij. Za vsako od obeh se hrani varianta z upoštevanjem verjetnosti zaprtja pogodbe in brez nje. Podrobnejši vsebinski opis teh polj sledi v nadaljevanju diplomskega dela.

**AccountPositionVolatilityShare**

Tabela hrani podatek o doprinosu posamezne pozicije k volatilnosti portfelja posamezne pogodbe. Volatilnost portfelja se shrani v tabelo *CustomerAccount*. Podrobnejši vsebinski opis sledi v sekciji 4.3.1.

**AccountTransactionBehavior**

Podatki o povprečni donosnosti finančnih instrumentov pred izvršitvijo transakcije, gledano pri posamezni pogodbi. Vsaka pogodba ima v tabeli en vhod. Pomembnejša polja so: povprečna donosnost finančnih instrumentov pred nakupom (gledano na eno-, tri- in šestmesečna obdobja) in povprečna donosnost finančnih instrumentov pred prodajo (gledano na enaka obdobja kot pri nakupu).

**Cluster**

Podatki o gručah. Vsaka gruča ima v tabeli en vhod. Za vsako od gruč tabela hrani agregirane podatke o lastnostih pogodb oz. strank, ki so člani te gruče. Ker se agregirani podatki raztezajo, praktično preko vseh prej naštetih skupin tabel, jih je preveč tudi samo za predstavitev pomembnejših, se bo namesto teh navedlo zgolj sledeče primere: povprečna starost strank, povprečna volatilnost portfeljev, število članov v gruči ...

### 4.1.6 Sistemske tabele

**ExternalSynchronization**

**Opis:** podatki o zadnjih zagonih procedur pri prenosu podatkov. Tabela za vsako od procedur, ki sodeluje pri prenosu podatkov, hrani

zadnji datum njenega zagona. Na ta način procedure vedo, od katerega datuma dalje morajo zaobjeti podatke pri prenosu.

**ParameterData**

**Opis:** podatki o parametrih, ki služijo kot vhodni parametri pri izračunih v aplikaciji. Za vsak parameter tabela hrani njegovo ime in vrednost.

Iz tega podatkovnega modela se zgradita dve podatkovni bazi, t. i. učna in živa podatkovna baza. Prva podatkovna baza služi učenju algoritma za napovedovanje zaprtij pogodb, druga vsem ostalim namenom. Medtem ko sta shemi pri obeh bazah enaki, se izbira podatkov, ki gredo vanju, razlikuje, pri čemer pri obeh predstavlja vir podatkov podatkovna baza aplikacije *Shark*. Poleg različne podatkovne vsebine odločitvi po ustvarjanju dveh baz botruje tudi medsebojno različna frekvenca osveževanja podatkov oz. zagona procedur za prenos podatkov. V naslednjem podpoglavju si bomo podrobneje ogledali proces prenosa podatkov.

## 4.2 Prenos podatkov

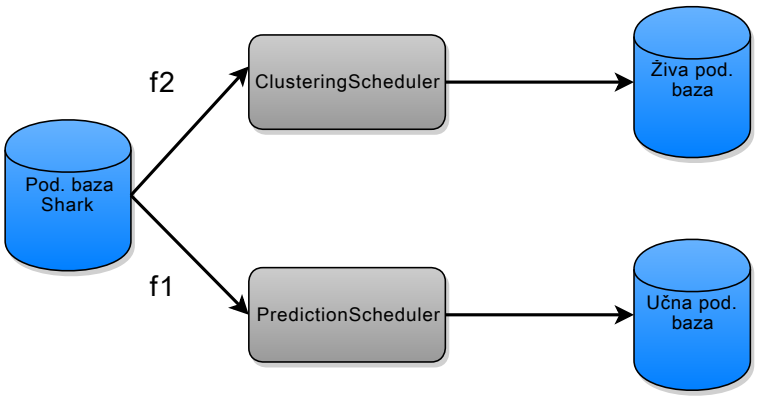
V fazi prenosa podatkov se iz *Sharkove* podatkovne baze prenesejo podatki v obe IS-ovi podatkovni bazi. Če označimo frekvenco osveževanja podatkov v učni bazi s $f_1$ in v živi bazi s $f_2$, potem velja neenakost

$$f_1 < f_2. \tag{4.1}$$

Razlog za različno frekvenco osveževanja je v tem, da klasifikatorja ni treba osveževati tako pogosto kot podatkov v živi bazi.

Proces prenosa podatkov je implementiran v obliki dveh *.NET* procesov: *ClusteringScheduler* in *PredictionScheduler*. Prvi skrbi za prenos podatkov v živo bazo, drugi pa v učno bazo. Oba se zaganjata periodično s pomočjo sistemske aplikacije *Windows Server-ja Task scheduler*. Periodi sta definirani kot

$$p_1 = \frac{1}{f_1} \tag{4.2}$$




Slika 4.3: Prenos podatkov

in

$$p_2 = \frac{1}{f_2}. \tag{4.3}$$

Vsak od teh procesov, kliče na bazi, za katero vrši prenos, *SQL* proceduro *PrenosPodatkov*, ki sproži prenos podatkov. Pri tem se omenjena procedura pri obeh bazah v svoji vsebini delno razlikuje, vendar o tem več v nadaljevanju podpoglavja. Celoten proces je za lažjo predstavo prikazan na sliki 4.3.

Omenjena procedura *PrenosPodatkov* na vsaki od obeh ciljnih podatkovnih baz igra vlogo vrhnje procedure, ki kliče vse preostale procedure, ki potem izvršijo sam prenos podatkov v tabele in potrebne prilagoditve ter izračune nad prenesenimi podatki, tako da so ti primerni za izvajanje preostalih operacij nad njimi. Procedura *PrenosPodatkov* vrši prenos v naslednjem vrstnem redu glede na, v prejšnjem podpoglavju definirane skupine tabel: šifranti, pogodbe strank in viri za agregate. Pri tem zaobjame vse podatke, ki so bili generirani od datuma zadnjega prenosa dalje, ki ga pridobi iz sistemske tabele *ExternalSynchronization* (obstoječim podatkom se osvežijo vrednosti). Če se prenos izvaja prvič, se zaobjame vse podatke od datuma vzpostavitve izvorne podatkovne baze dalje. Psevdokoda procedure 1 prikazuje potek procedure *PrenosPodatkov*. Metoda *DatumZadnjegaPrenosa*, po prej opisanem postopku, pridobi datum, od katerega naprej se vrši prenos podatkov. Za

zgornjo mejo se vzame trenutni datum. Zatem se izvrši prenos podatkov po skupinah tabel. Pri prvih dveh skupinah ni nobenih posebnosti, zato jih ne bomo opisovali. Podrobneje si bomo ogledali prenos virov za agregate, kjer se izvajajo prilagoditve in izračuni prenesenih podatkov. Ker v nadaljevanju tega podpoglavja sledi opis teh operacij, je prenos podatkov pri tej skupini prikazan ločeno po posameznih metodah, ki izvajajo te operacije.

**Procedura 1** Prikaz procedure PrenosPodatkov

$DatumOd \leftarrow DatumZadnjegaPrenosa()$

$PrenosSifrantov(DatumOd, DatumDo)$
$PrenosPogodbStrank()$

$IzracunVrednostiPortfeljev(DatumOd, DatumDo)$
$GlajenjeSpremembVrednostiPortfeljev(DatumDo)$
$IzracunDonosnostiPortfeljev(DatumDo)$
$ZapiranjeOdpiranjePogodb(DatumDo)$
$PrenosOstalihVirovZaAgregate(DatumOd, DatumDo)$

**if** $PodatkovnaBaza = ZivaPB$ **then**
    $IzracunAgregatovZivaBaza()$
    $IzracunDonosnostiFIPredTransakcijo()$
    $PolnjenjeSistemskihTabel()$
**else**
    $IzracunAgregatovUcnaBaza()$
**end if**
$ShraniDatumZadnjegaPrenosa(DanasnjiDatum())$

Sledi izračun agregatov nad prenesenimi podatki, kjer je tudi točka, v kateri se procedura *PrenosPodatkov* razlikuje med obema podatkovnima bazama. Pri živi bazi se preprosto izračuna agregate za vse še odprte pogodbe.

Pri učni bazi je način izbora pogodb nekoliko kompleksnejši; tukaj se agregate izračuna za vse zaprte pogodbe, ki so bile odprte vsaj leto dni, in za vsako od njih še za eno odprto pogodbo, ki je bila odprta vsaj leto dni pred zaprtjem njene parne pogodbe. Pri tem se nobena pogodba v naboru odprtih pogodb ne sme podvojiti. Za boljšo predstavo je priložena psevdokoda v proceduri 2, kjer je ta postopek prikazan.

Pri živi bazi se zatem izvedeta še polnjenje sistemske tabele s parametri, ki se bodo uporabljali pri izračunih, in izračun povprečne donosnosti finančnih instrumentov pred nakupom in prodajo, pri vsaki pogodbi. Na koncu se shrani še datum prenosa, tako da se naslednjič lahko določi pravilno obdobje zaobjema podatkov za prenos.

**Procedura 2** Prikaz izbire pogodb za učno bazo

**Vhod:** $vseZaprtePogodbe$
**Vhod:** $vseOdprtePogodbe$

**for all** $zaprtaPogodba \in vseZaprtePogodbe$ **do**
  $veljaDoZP \leftarrow zaprtaPogodba.veljaDo$
  $veljaOdZP \leftarrow zaprtaPogodba.veljaOd$
  $dolzinaOdprtja \leftarrow razlikaSteviloDni(veljaOdZP, veljaDoZP)$
  **if** $dolzinaOdprtja >= 365$ **then**
    $izbranePogodbe.dodaj(zaprtaPogodba)$
  **end if**
  **for all** $odprtaPogodba \in vseOdprtePogodbe$ **do**
    $veljaOdOP \leftarrow odprtaPogodba.veljaOd$
    $razlikaDni \leftarrow razlikaSteviloDni(veljaOdOP, veljaDoZP)$
    $neVsebuje \leftarrow izbranePogodbe.neVsebuje(odprtaPogodba)$
    **if** $razlikaDni \geq 365 \wedge neVsebuje$ **then**
      $izbranePogodbe.dodaj(odprtaPogodba)$
    **end if**
  **end for**

**end for**

**return** *izbranePogodbe*

V prihajajočih sekcijah sledi opis metod, naštetih pri prenosu podatkov iz skupine viri za agregate v proceduri 1 in pomembnejših operacij pri agregiranju podatkov.

### 4.2.1 Izračun vrednosti portfeljev

Glavna naloga te metode je, da ovrednoti portfelje vseh pogodb. Vrednotenje portfelja je definirano kot

$$\sum_{i=1}^{n} KonverzijaEUR_i * VrednostPoz_i, \tag{4.4}$$

pri čemer zgornja meja vsote, označena kot $n$, predstavlja število pozicij v portfelju. $KonverzijaEUR$ je faktor za pretvorbo vrednosti $VrednostPoz_i$ v evrsko valuto, ki je definiran kot

$$KonverzijaEUR = \frac{1}{TecajValute}, \tag{4.5}$$

kjer je $TecajValute$ enak trenutnemu menjalnemu razmerju med valuto vrednosti $VrednostPoz_i$ in EUR (ko je valuta EUR, je enak 1). $VrednostPoz_i$ predstavlja vrednost pozicije, ki se določa različno glede na tip finančnega instrumenta, in sicer:

**Obveznice**

Vrednost se računa po enačbi

$$Kolicina * ((\frac{Tecaj}{100} * Glavnica) + Obresti), \tag{4.6}$$

pri čemer *Kolicina* predstavlja število obveznic, *Tecaj* predstavlja njeno trenutno tržno ceno, ki je odražena kot procent glavnice, *Glavnica* je osnova, glede na katero se izračunajo obresti, in *Obresti* so vrednost vseh natečenih obresti do datuma, na katerega se izvaja vrednotenje.

**Depoziti**

Vrednost se računa po enačbi

$$Kolicina + Obresti, \tag{4.7}$$

kjer *Kolicina* označuje vrednost denarja, ki je bil položen kot depozit, in *Obresti* označujejo vrednost vseh natečenih obresti do datuma, na katerega se izvaja vrednotenje.

**Denar**

Tukaj ni nobenega izračuna, saj je vrednost preprosto enaka vrednosti denarja.

**Delnice**

Vrednost se računa po enačbi

$$Kolicina * Tecaj, \tag{4.8}$$

kjer *Kolicina* predstavlja število delnic in *Tecaj* končno tržno ceno na dan vrednotenja.

**Izvedeni finančni instrumenti**

Vrednost se računa po enačbi

$$Kolicina * Multiplikator, \tag{4.9}$$

kjer je multiplikator neka interna vrednost, ki jo določi BPD, in nas njen pomen ne zanima.

Pri vseh pogodbah se portfelje ovrednoti za vse dni od datuma odprtja pogodbe naprej. Posamezne pozicije metoda pridobiva iz tabele *Portfolio*, tečaje vseh finančnih instrumentov iz tabele *MarketPrice* in končni rezultat shranjuje v tabelo *PortfolioValueCache*.

Ta metoda je prav tako zadolžena za izračun denarnih prilivov in odlivov, ki se pri vseh portfeljih, poleg vrednosti, izračunajo za vsak datum. Vsi ti podatki so pridobljeni iz izvorne podatkovne baze, kjer se že opravijo ustrezni izračuni, zato dodatna razlaga na tem mestu ni potrebna.

### 4.2.2 Glajenje sprememb vrednosti portfeljev

Pri posameznih finančnih instrumentih se lahko zgodi, da je bil za kakšen datum napačno vnesen tečaj, kar se lahko odrazi v nenadnem opaznem skoku oz. padcu vrednosti portfeljev, ki držijo ta finančni instrument. Magnituda spremembe vrednosti pri posameznem portfelju je odvisna od tega, koliko vneseni tečaj odstopa od dejanskega tečaja in količine lotov, kar je razvidno tudi iz enačbe za izračun vrednosti portfelja. Naloga te metode je glajenje tovrstnih sprememb vrednosti portfelja.

Najprej metoda poišče vse datume, znotraj v argumentih podanega datumskega razpona, na katere vrednost portfelja ustreza enemu od naslednjih dveh kriterijev:

- Vrednost portfelja je enaka ali večja 200% vrednosti predhodnega in naslednjega datuma.
- Vrednost je enaka ali manjša 50% vrednosti predhodnega in naslednjega datuma.

Zatem za vsakega od teh datumov preveri, ali se ni zgodil priliv oz. odliv, saj bi to pomenilo, da je sprememba vrednosti rezultat tega in ne napačno vnesenega tečaja. Če se izkaže, da ni bilo priliva oz. odliva, se vrednost prepiše z vrednostjo portfelja na prejšnji dan.

Procedura 3 prikazuje ta postopek, ki se izvede na vseh portfeljih. Kot vhod je podana identifikacija portfelja z *portfeljID*, datumski razpon z *datumOd* in *datumDo*, seznam parov *(datum, vrednost)*, imenovan *vrednost*, ki hrani preslikavo $datum \rightarrow vrednostPortfelja$, ter seznam parov *(datum, netoPriliv)*, imenovan *netoPrilivi*, ki hrani preslikavo $datum \rightarrow (priliv - odliv)$. Oba seznama hranita vrednosti na vse datume, ki so se izračunali v metodi *IzracunVrednostiPortfeljev*.

**Procedura 3** Glajenje nenadnih skokov oz. padcev vrednosti portfelja

**Vhod:** portfeljID

**Vhod:** $datumOd, datumDo$
**Vhod:** $vrednosti$
**Vhod:** $netoPrilivi$
  **for all** $vrednost \in seznamVsehVrednostiPortfelja$ **do**
    **if** $vrednost[datum] >= 2 * vrednost[prejsnjiDatum]$ **then**
      **if** $vrednost[datum] >= 2 * vrednost[naslednjiDatum]$ **then**
        **if** $netoPrilivi[datum] = 0$ **then**
          $vrednost[datum] \leftarrow vrednost[prejsnjiDatum]$
        **end if**
      **end if**
    **end if**
    **if** $vrednost[datum] <= 0.5 * vrednost[prejsnjiDatum]$ **then**
      **if** $vrednost[datum] <= 0.5 * vrednost[naslednjiDatum]$ **then**
        **if** $netoPrilivi[datum] = 0$ **then**
          $vrednost[datum] \leftarrow vrednost[prejsnjiDatum]$
        **end if**
      **end if**
    **end if**
  **end for**

### 4.2.3 Izračun donosnosti portfeljev

Za vsak portfelj se izračuna njegova 30-, 90-, 180- in 365-dnevna donosnost. To se izračuna s pomočjo t. i. *prilagojene Dietzove metode* [16]. V grobem povedano, ta metoda izračuna razliko v vrednosti portfelja na dan izračuna donosnosti in začetnim datumom obdobja, za katerega se računa donosnost, ter pri tem upošteva časovno obtežene denarne prilive in odlive, ki so nastali v tem obdobju (npr. priliv, nastal tik pred koncem obdobja, ima manjšo težo kot priliv, nastal tik po začetku obdobja, saj se je ta dlje obrestoval).

Enačba *prilagojene Dietzove metode* se glasi

$$Donosnost = \frac{EMV - BMW - F}{BMW + \sum_{i=1}^{n} W_i * F_i} = \frac{Dobicek}{PovprecniKapital}, \tag{4.10}$$

pri čemer posamezni faktorji označujejo naslednje:

- $EMV$**:** vrednost portfelja na datum izračuna donosnosti oz. na koncu obdobja, v katerem se računa donos.
- $BMV$**:** vrednost portfelja na začetku obdobja, v katerem se računa donos.
- $F$**:** bilanca vseh prilivov in odlivov v obdobju računanja donosnosti opisana z enačbo

$$F = \sum Prilivi - \sum Odlivi. \tag{4.11}$$

- $F_i$**:** vrednost posameznega priliva oz. odliva v obdobju računanja donosnosti. Prilivi imajo pozitiven, odlivi negativen predznak.
- $W_i$**:** utež, ki prilive in odlive časovno obteži, glede na čas njihovega nastanka v obdobju računanja donosnosti. Utež se izračuna po enačbi

$$W_i = \frac{CD - D_i}{CD}, \tag{4.12}$$

kjer $CD$ označuje število dni v obdobju računanja donosnosti in $D_i$ število dni od začetka obdobja do nastanka $F_i$.

### 4.2.4 Zapiranje in odpiranje pogodb

Potem ko je za vsak portfelj znana njegova zgodovina vrednosti, se določi, katere pogodbe se bodo umetno zaprle. Zaprte pogodbe se ločujejo, glede na način zaprtja, na dva tipa:

- **Naravno zaprte:** pogodbe, za katere so se stranke same odločile, da jih bodo zaprle.
- **Umetno zaprte:** pogodbe, ki jih kot zaprte označi metoda *Zapiranje-OdpiranjePogodb*.

| Datum | Vrednost |
|---|---|
| 30. 3. 2013 | 40 |
| 29. 3. 2013 | 40 |
| ... | ... |
| 30. 9. 2012 | 10 |
| 29. 9. 2012 | 10 |
| 28. 9. 2012 | 115.0 |

Tabela 4.1: Primer portfelja, ki ustreza kriteriju za zaprtje.

Medtem ko pri naravno zaprtih pogodbah dodatna razlaga ni potrebna, je pri umetno zaprtih treba razložiti mehanizem, po katerem se odloča o njihovem zaprtju. Metoda zapre vsako pogodbo, pri kateri v roku zadnjih šest mesecev ni bilo dneva, ko bi vrednost njenega portfelja presegala 100 EUR. Kot datum zaprtja se določi prvi datum, pri katerem je bila vrednost večja od 100 EUR. Če tak datum ne obstaja, se datum zaprtja nastavi na datum odprtja. Vse umetno zaprte pogodbe se loči od naravno zaprtih, in sicer po polju *NoActivityClose*, ki je v tabeli *CustomerAccount*: umetno zaprte pogodbe imajo to polje nastavljeno na 1, naravno zaprte na 0. Razlog za umetno zapiranje pogodb je v tem, da so takšne pogodbe za vse praktične namene v aplikaciji identične naravno zaprtim pogodbam, saj na njih ne poteka omembe vredna aktivnost.

Tabela 4.1 prikazuje primer portfelja, ki ustreza kriteriju za zaprtje pogodbe. V levem stolpcu je datum vrednotenja, v desnem vrednost portfelja na ta datum. Za referenčni datum, od katerega nazaj se gleda 6-mesečni kriterij za zaprtje pogodbe, se vzame 30. 3. 2013. Ko se od referenčnega datuma odšteje šest mesecev, se pride na 30. 9. 2012. Če se privzame, da je na vse vmesne dni bila vrednost portfelja manj kot 100 EUR (vmesni dnevi so zgolj simbolično prikazani), potem na dan 30. 9. 2012 pogodba ustreza kriteriju za zaprtje, saj je bila njegova vrednost šest mesecev neprekinjeno manjša od 100 EUR. Kot datum zaprtja se nastavi 28. 9. 2012, saj je takrat

nazadnje bila njegova vrednost večja od 100 EUR.

Na koncu metoda še za vse, pri svojem prejšnjem zagonu, umetno zaprte pogodbe preveri, ali se je od datuma njihovega zaprtja kdaj vrednost njihovega portfelja povzpela nad 100 EUR. Če obstaja datum, na katerega je bila večja od te meje, se takšno pogodbo ponovno odpre. To se stori z izbrisom vrednosti v poljih *ValidTo* in *NoActivityClose* v tabeli *CustomerAccount*.

### 4.2.5 Izračun agregatov

Kot že rečeno, vsi izračuni v tabelah iz skupine agregati temeljijo na podatkih, prenesenih v tabele iz skupine viri za agregate. Procedura *PrenosPodatkov* v sklopu klicev procedur *IzracunAgregatovZivaBaza* in *IzracunAgregatovUcnaBaza* napolni vse tabele skupine agregati. Posamezni vnos v vsaki od teh tabel predstavlja izračunani agregat za posamezen portfelj.

Razlika med procedurama pri obeh bazah je v izbiri pogodb, za katere se bodo izračunali agregati, in v obdobju, v katerem se bodo zaobjeli podatki za izračun agregatov. Pri živi bazi se zaobjame vse odprte pogodbe, kot obdobje pa se vzame zadnje leto dni, gledano od datuma izračuna agregatov. V primeru učne baze se vzame pogodbe, izbrane po postopku, prikazanem v proceduri 2, pri čemer se za obdobje vzame zadnje leto dni od zaprtja pogodbe pri zaprtih pogodbah in enako obdobje pri njihovih parnih, še odprtih, pogodbah. Polnjenje tabel oz. njihovih polj se vrši na naslednji način:

**PortfolioValueCacheAvg**

- **InflowFrequencyAvg**: število vseh, v podanem obdobju, izvršenih denarnih prilivov.
- **OutflowFrequencyAvg**: število vseh, v podanem obdobju, izvršenih denarnih odlivov.
- **NumPositionAvg**: povprečno število različnih pozicij, ki jih je portfelj držal v podanem obdobju. Povprečje se izračuna po na-

slednji formuli

$$\frac{\sum_{i=1}^{365} StPozicijDan_i}{365}, \tag{4.13}$$

pri čemer $StPozicijDan_i$ označuje število pozicij v portfelju na i-ti dan (vsi dnevi so znotraj podanega obdobja).

- **NumDomesticPositionAvg**: povprečno število različnih pozicij s finančnimi instrumenti, ki kotirajo na LJSE in jih je portfelj držal v podanem obdobju. Povprečje se izračuna po isti formuli kot za *NumPositionAvg*, s to razliko, da se šteje samo pozicije, ki kotirajo na LJSE.
- **NumForeignPositionAvg**: povprečno število različnih pozicij s finančnimi instrumenti, ki ne kotirajo na LJSE in jih je portfelj držal v podanem obdobju. Povprečje se izračuna po isti formuli kot za *NumPositionAvg*, s to razliko, da se šteje samo pozicije, ki ne kotirajo na LJSE.
- **ValueAvg:** povprečna vrednost portfelja v podanem obdobju. Povprečje se izračuna po naslednji formuli

$$\frac{\sum_{i=1}^{365} VrednostDan_i}{365}, \tag{4.14}$$

pri čemer $VrednostDan_i$ označuje vrednost portfelja na i-ti dan (vsi dnevi so znotraj podanega obdobja).

**MTOTransactionAvg**

- **BuyFrequency:** število nabavnih transakcij v podanem obdobju.
- **SellFrequency:** število prodajnih transakcij v podanem obdobju.
- **BuyValueAvg:** povprečna vrednost nabavne transakcije v podanem obdboju.

- **SellValueAvg:** povprečna vrednost prodajne transakcije v podanem obdboju.
- **OnlineOrderShare:** delež transakcij, ki so bile opravljene preko spletnega vmesnika. Te gre razpoznati po polju *IsOnlineOrder* v tabeli *MTOTransaction*.

**CommissionSum**

Vsa spodnja polja so sumirana na mesečni ravni, za vse mesece v podanem obdobju.

- **ManagementFeeSum:** vsota upravljalskih provizij.
- **TransactionFeeSum:** vsota posredniških provizij.
- **SafekeepingFeeSum:** vsota ležarin.
- **ProfitshareFeeSum:** vsota vrednosti deleža pri delitvi dobička.

### 4.2.6 Donosnost finančnih instrumentov pred izvršitvijo transakcije

Izračun donosnosti finančnih instrumentov pred izvršitvijo transakcije meri povprečno donosnost finančnih instrumentov pred njihovim nakupom in prodajo pri posamezni pogodbi. Donosnosti se izračunajo ločeno za nabavne in prodajne transakcije. Pri računanju povprečja se vzamejo vse nabavne oz. prodajne transakcije pri posamezni pogodbi in se zanje izračuna povprečna donosnost za izbrano obdobje. Obdobja so naslednja: 1 mesec, 3 meseci in 6 mesecev. Enačba za izračun donosnosti je pri vseh finančnih instrumentih enaka in je definirana kot

$$Donosnost = \frac{KoncniTecaj}{ZacetniTecaj} - 1, \tag{4.15}$$

kjer $KoncniTecaj$ predstavlja tečaj na končni datum v obdobju in $ZacetniTecaj$ označuje tečaj na začetni datum v obdobju.

Izračuni donosnosti se opravijo za vse pogodbe. Na koncu se rezultati shranijo v tabelo *AccountTransactionBehavior*.

## 4.3 Preostali izračuni in metode strojnega učenja

V prejšnjem podpoglavju smo si ogledali dve od, v uvodu naštetih, šestih ključnih funkcionalnosti IS. V tem podpoglavju sledi še opis preostalih štirih. Potem ko se je prenos podatkov izvršil, se začnejo vsi izračuni in zagon algoritmov za razvrščanje ter napovedovanje zaprtij pogodb. Vse to se izvaja v okviru procesov *ClusteringProcess* in *PredictionProcess*, potem ko sta izvedla prenos podatkov.

V okviru prvega procesa se za vse pogodbe, ki so še odprte, napove in izračuna pripadajočo verjetnost za zaprtje pogodbe ter se izvršita izračuna volatilnosti in življenjske vrednosti pogodbe. Na koncu se opravi še razvrščanje pogodb. Procedura 4 nazorno prikaže opisani potek izvajanja znotraj procesa *ClusteringProcess*. Najprej pride do prenosa podatkov, opisanega v prejšnjem podpoglavju, nato sledijo izračuni, ki si jih bomo ogledali v tem podpoglavju. *PredictionProcess* na drugi strani, poleg prenosa podatkov, izvaja zgolj gradnjo klasifikatorja za napovedovanje zaprtij pogodb. Njegov potek prikazuje procedura 5.

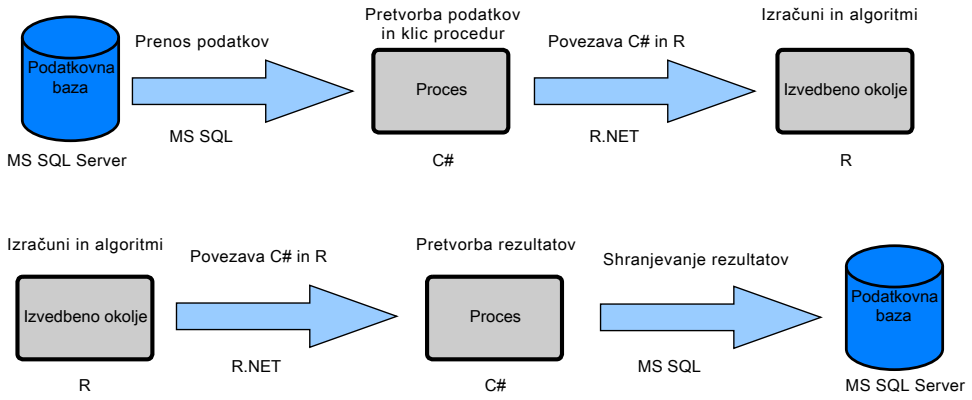
**Procedura 4** Prikaz procesa ClusteringProcess

$PrenosPodatkovZivaBaza()$
$IzracunVolatilnostiPortfelja()$
$IzracunZivljenjskeVrednostiPogodb()$
$NapovedZaprtjaPogodb()$
$RazvrscanjePogodb()$

**Procedura 5** Prikaz procesa PredictionProcess

$PrenosPodatkovUcnaBaza()$
$GradnjaKlasifikatorja()$

Oba procesa sta implementirana s pomočjo tehnologij *C#, MS SQL Server, R.NET* in *R*. Sami izračuni in gradnja algoritmov so napisani v *R-ju*, medtem ko ostale tehnologije skrbijo za dostavljanje vhodnih podatkov




Slika 4.4: Potek uporabljenih tehnologij.

izračunom in algoritmom ter za shranjevanje njihovih rezultatov. Najprej se podatki preberejo iz podatkovne baze s pomočjo *SQL-a* (podatkovna baza je implementirana s pomočjo *MS SQL Server 2008*), nato jih *C#* ustrezno pretvori za posredovanje izračunom in algoritmom v *R-ju* ter sproži njihov klic. Medtem ko *R.NET* skrbi za samo povezavo med *C#* in *R*. Nato *R-jevo* izvedbeno okolje opravi izračune in izvede ustrezne algoritme. Njihovi rezultati se potem preko obratnega vrstnega reda rabe teh tehnologij shranijo nazaj v podatkovno bazo. Za lažjo predstavo je celoten postopek prikazan na sliki 4.4. Zgornji del slike prikazuje potek do zagona izračunov in algoritmov, spodnji del pa do shranitve njihovih rezultatov.

V nadaljevanju sledi podroben opis izvajanja vsake od procedur obeh procesov (razen procedur za prenos podatkov, ki sta bili že opisani v prejšnjem poglavju). Vsaka sekcija vsebuje opis ene procedure, z izjemo napovedi zaprtja pogodb, kjer sta proceduri *NapovedZaprtjaPogodb()* in *GradnjaKlasifikatorja()* opisani v skupni sekciji, saj je zaradi njune tesne vsebinske in funkcionalne povezanosti to najbolj smiselno.

### 4.3.1 Volatilnost

Volatilnost portfelja meri, koliko je posamezni portfelj izpostavljen nihanju njegove vrednosti oz. tveganju. Na začetku postopka izračuna volatilnosti portfelja [12, 31], se za vse finančne instrumente v portfelju pridobi njihove dnevne donosnosti za zadnjih 120 trgovalnih dni. Dnevna donosnost se izračuna kot

$$DnevnaDonosnost = \frac{T_i}{T_{i-1}}, \tag{4.16}$$

kjer $T_i$ označuje vrednost tečaja na $i$-ti trgovalni dan in $T_{i-1}$ na en trgovalni dan prej. Časovna vrsta dnevnih donosnosti za posamezen finančni instrument tvori vektor $X_i$, kjer indeks $i$ predstavlja zaporedno številko posameznega finančnega instrumenta v portfelju (določitev vrstnega reda je poljubna). Zatem se za vse možne permutacije parov vektorjev, označene kot $(X, Y)$, izračuna njihove kovariance po enačbi [3]

$$s_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}, \tag{4.17}$$

kjer zgornja meja vsote $n$ označuje dolžino vektorjev, $x_i$ in $y_i$ posamezno vrednost vsakega od vektorjev ter $\bar{x}$ in $\bar{y}$ povprečno vrednost vsakega od vektorjev.

Ko so za vse pare izračunani njihove kovariance, se iz njih tvori t. i. kovariančna matrika

$$S = \begin{pmatrix} s_{X_1,X_1} & s_{X_1,X_2} & \cdots & s_{X_1,X_n} \\ s_{X_2,X_1} & s_{X_2,X_2} & \cdots & s_{X_2,X_n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{X_n,X_1} & s_{X_n,X_2} & \cdots & s_{X_n,X_n} \end{pmatrix},$$

pri kateri posamezni element predstavlja kovarianco para vektorjev (tokrat je drugi vektor iz para označen kot $X_j$, da je bolj jasno, da diagonalni elementi matrike predstavljajo kovariance vektorjev samih s sabo) in $n$ označuje število vektorjev v danem portfelju. Ker pri parih vrstni red vektorjev ne vpliva na

izračun kovariance (npr. vrednosti $s_{X_1,X_2}$ in $s_{X_2,X_1}$ sta enaki), velja, da je kovariančna matrika simetrična.

Za izračun volatilnosti portfelja je poleg kovariančne matrike treba izračunati še vektor uteži, ki pripadajo posamezni poziciji v portfelju. Utež pove, kolikšen delež vrednosti portfelja predstavlja vrednost dane pozicije. Za vse pozicije v portfelju se njihove uteži izračuna po enačbi

$$w_i = \frac{v_i}{P}, \tag{4.18}$$

kjer $v_i$ predstavlja vrednost posamezne pozicije in $P$ vrednost portfelja. Uteži vseh pozicij v portfelju tvorijo vektor $W$.

Sledi izračun volatilnosti portfelja, ki je definirana kot

$$Volatilnost = \sqrt{W \times S \times W^T}. \tag{4.19}$$

Pri tem se za vsako od pozicij v portfelju izračuna še njen prispevek k volatilnosti [25], ki je izražen v obliki deleža volatilnosti celotnega portfelja. Vsi izračunani prispevki tvorijo vektor $V$, ki se ga pridobi na naslednji način:

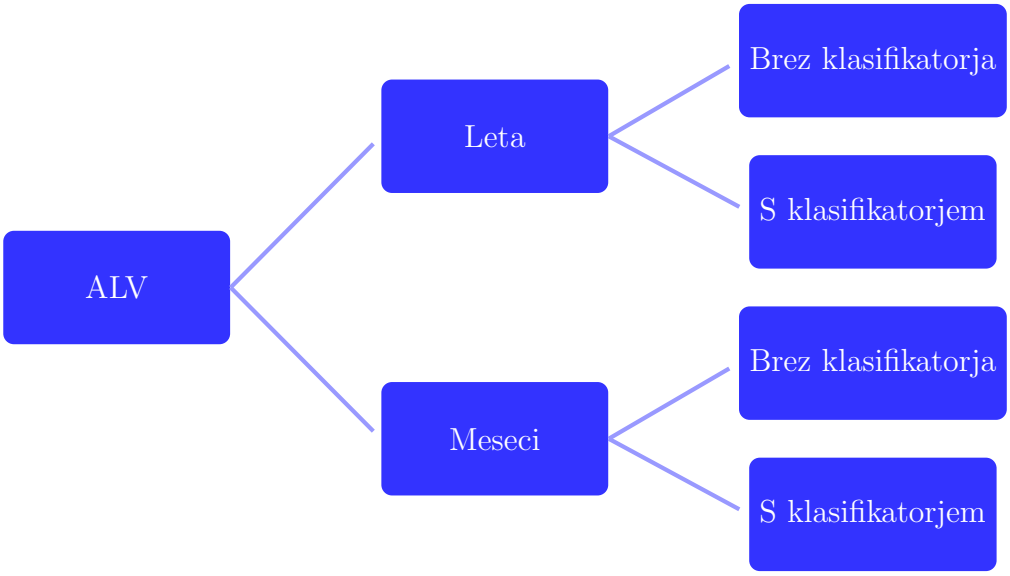
$$V' = \frac{S \times W}{Volatilnost * W}, \tag{4.20}$$

$$V = \frac{V'}{Volatilnost}. \tag{4.21}$$

Na koncu, ko so vsi izračuni opravljeni, se volatilnost celotnega portfelja shrani v tabelo *CustomerAccount*, vrednosti iz vektorja $V$ pa v tabelo *AccountPositionVolatilityShare*.

### 4.3.2 Življenjska vrednost pogodb

Izračun življenjske vrednosti pogodbe *(ALV) (angl. account lifetime value)*, na podlagi preteklih prihodkov, ki so bili ustvarjeni na račun posamezne pogodbe, zanjo oceni prihodke, ki bodo nastali v prihodnosti. Prihodki sestojijo iz vseh štirih tipov provizij, ki so bili opisani v podpoglavju 2.2. Enačba za




Slika 4.5: Prikaz vseh štirih variacij izračuna življenjske vrednosti pogodbe.

izračun $ALV$ je nekoliko modificirana verzija enačbe [27] in se glasi

$$ALV = \sum_{t=0}^{T} \frac{c * r_t}{(1+i)^t}, \tag{4.22}$$

kjer $T$ predstavlja dolžino obdobja (izraženo v številu let oz. številu mesecev; podrobnejši opis izbire obdobij sledi v nadaljevanju), $c$ je povprečna letna oz. mesečna vrednost provizij, $r_t$ je verjetnost, da je pogodba še odprta po preteku $t$ časovnih enot in $i$ predstavlja diskontni faktor (njegova funkcija je diskontiranje vseh prihodnjih prihodkov, pri čemer je diskontna stopnja sorazmerna s časovno oddaljenostjo prihodkov; načeloma se za diskontni faktor vzame povprečna stopnja inflacije).

Izračunajo se štiri različne variacije $ALV$, ki se medsebojno razlikujejo po vrednostih parametrov v enačbi, pri čemer je zgolj diskontni faktor $i$ enak pri vseh. Slika 4.5 prikazuje vse variacije (te predstavljajo listi grafa).

Najprej se $ALV$ deli glede na časovne enote, v katerih je izraženo obdobje $T$: v številu let ali številu mesecev. $T$ je pri obeh letnih variantah definiran kot navzdol zaokroženo povprečno število let, ki so jih bile pogodbe odprte pred njihovim zaprtjem, in kot enaka vrednost, izražena v mesecih pri

mesečnih variantah (to se doseže preprosto tako, da se letno povprečje pomnoži z 12). Prva stopnja delitve $ALV$ vpliva tudi na vrednost parametra $c$; pri letnih variantah je ta enak letnemu povprečju vseh provizij za zadnja tri leta pri dani pogodbi in pri mesečnih variantah mesečnemu povprečju vseh provizij za zadnje leto dni.

Zadnja stopnja delitve $ALV$ na variacije se nanaša na vključitev oz. izključitev klasifikatorja za napovedovanje zaprtij pogodb. Pri njegovi vključitvi sta prisotna dva dodatna faktorja, ki vplivata na izračun parametra $r_t$, in sicer:

- **Natančnost klasifikatorja:** ta označuje odstotek pogodb v učni množici, katerih stanje (pogoba je lahko odprta ali zaprta) je klasifikator pravilno napovedal.
- **Verjetnost zaprtja:** verjetnost, ki jo klasifikator napove za zaprtje posamezne pogodbe.

Izračun parametra $r_t$ se, odvisno glede na varianto pri delitvi, glasi:

$$r_t = 1 - p_t \tag{4.23}$$

oz.

$$r_t = \frac{(1 - p_t) * (1 - p_E)}{N}. \tag{4.24}$$

Prva enačba se uporabi pri varianti brez vključitve natančnosti klasifikatorja za napovedovanje zaprtja pogodb, druga pri preostali varianti, kjer je natančnost vključena. Parameter $p_t$ označuje verjetnost, da je bil račun zaprt v času, manjšem ali enakem od $t$ enot, $p_E$ označuje verjetnost zaprtja pogodbe in $N$ označuje natančnost tega klasifikatorja.

Na koncu, ko so bile vse variante $ALV$ izračunane, se jih shrani v tabelo $AccountLifetimeValue$.

### 4.3.3 Napovedovanje zaprtij pogodb

Napovedovanje zaprtij pogodb za vsako pogodbo napove, ali se bo zaprla, in kakšna je verjetnost za to. Ker je stanje zaprtja pogodbe po svoji naravi

binarno, gre za t. i. klasifikacijski problem. Napovedi se opravlja s t. i. klasifikatorjem, ki ga zgradimo s pomočjo Breimanovega algoritma naključnih gozdov. Podrobnejšo razlago delovanja in teoretičnega ozadja si je možno prebrati v sekciji 3.2.2, zato bomo tu zgolj na kratko povzeli in aplicirali na to problemsko domeno tiste stvari, ki so ključne za razumevanje postopka napovedovanja zaprtij pogodb.

Najprej se izbere vse že zaprte pogodbe, ki so bile pred zaprtjem odprte vsaj leto dni, in za vsako od njih se izbere še eno pogodbo, ki je že bila odprta v času zaprtja njene parne zaprte pogodbe (ta del se opravi že med prenosom podatkov in je v njegovem sklopu tudi podrobneje opisan, v okviru podpoglavja 4.2). Te pogodbe tvorijo celotno množico, ki je hkrati tudi učna množica, nad katero se zgradi klasifikator. Validacijsko množico algoritem tvori sam, tako da je nam ni treba.

Pri pridobivanju teh pogodb se iz podatkovne baze zanje tudi pridobi seznam vseh atributov, nad katerimi se bo opravljala klasifikacija. V grobem povedano, se gradnja klasifikatorja vrti okoli tega, da se klasifikator nauči, kateri atributi in pri kakšnih vrednostih najbolje napovedo ciljni razred. Tega predstavlja stanje pogodbe, ki je lahko odprta ali zaprta. Pri napovedovanju klasifikator to povezavo uporablja za svoje napovedi.

V aplikaciji se za gradnjo klasifikatorja uporablja knjižnico CARET [4] za programski jezik R. Ta knjižnica ponuja metode za vse zgoraj opisane postopke. Najprej se iz učne podatkovne baze pridobi množico pogodb in vrednosti vseh atributov, nad katerimi se bo opravljala klasifikacija. Razred se nastavi na binarno vrednost 'da' ali 'ne'. Prva označuje, da je pogodba zaprta, druga, da še ni. Ta se nastavi, tako da se pogleda, ali ima pogodba že vpisan datum zaprtja ali še ne. Sledi gradnja klasifikatorja nad učno množico, ki se zgradi s pomočjo algoritma naključnih gozdov. Najprej se izbere tri različne vrednosti za število izbranih atributov $m$ in za vsako od njih se zgradi klasifikator. Za nizko vrednost se izbere 2, visoka je enaka številu vseh atributov, torej v našem primeru 18, in srednja se določi z enačbo

$$\lfloor \frac{NizkaVrednost + VisokaVrednost}{2} \rfloor. \tag{4.25}$$

Število dreves je medtem povsod enako 500 (s preizkušanjem smo ugotovili, da večje število dreves ne izboljša kvalitete klasifikatorja). Pri vsaki vrednosti se nad zgrajenim klasifikatorjem oceni tudi njegova natančnost nad validacijsko množico. Za končni klasifikator se vzame tistega z največjo natančnostjo. Ta se potem shrani v datoteko, od koder je dostopen za izvajanje napovedi v živi bazi. Prav tako se v podatkovno bazo shrani njegova natančnost, in sicer v tabelo *ParameterData*. Potrebujemo jo namreč pri izračunu življenjske vrednosti pogodb. Procedura 6 prikazuje celoten postopek:

**Procedura 6** Postopek gradnje napovednega klasifikatorja

$KoncniKlasifikator$
$MaxNatancnost \leftarrow 0$
$MaxVrednost \leftarrow 0$

$Podatki \leftarrow pridobiPodatkeIzPodatkovneBaze()$
$Podatki \leftarrow nastaviRazred(Podatki)$
$SteviloAtributov \leftarrow pridobiSteviloAtributov(Podatki)$
$SeznamVrednosti \leftarrow ustvariSeznamVrednosti(SteviloAtributov)$

**for all** $Vrednost \in SeznamVrednosti$ **do**
  $Klasifikator \leftarrow zgradiKlasifikator(Podatki, Vrednost)$
  **if** $Klasifikator.Natancnost > MaxNatancnost$ **then**
    $KoncniKlasifikator \leftarrow Klasifikator$
    $MaxNatancnost \leftarrow Klasifikator.Natancnost$
  **end if**
**end for**
$ShraniVDatoteko(KoncniKlasifikator)$
$ShraniVPodatkovnoBazo(MaxNatancnost)$

Pri tej podatkovni bazi celotna množica pogodb vsebuje 3758 primerov in 18 atributov, ki jih uporabljajo za klasifikacijo. Za optimalno vrednost števila

| | | **Napovedan razred** | | |
|---|---|---|---|---|
| | | **Yes** | **No** | **Skupno:** |
| **Dejanski razred** | **Yes** | 1864 | 15 | 1879 |
| | **No** | 128 | 1751 | 1879 |
| | **Skupno:** | 1992 | 1766 | |

Slika 4.6: Prikaz napovedanih in dejanskih razredov za vse primere. Zaradi velikosti naključnega gozda se je vsak primer znašel pri približno tretjini dreves v validacijski množici. V tabeli je primer klasificiran v razred, ki je prejel večino glasov.

izbranih atributov je bilo izbrano število 10, kjer je bila natančnost klasifikatorja enaka 96,2 %. Tabela 4.6 prikazuje rezultat klasifikacije vseh primerov. Na podlagi izračunane natančnosti lahko zaključimo, da smo zgradili dovolj kakovosten klasifikator.

### 4.3.4 Razvrščanje pogodb

Ko so bili vsi izračuni opravljeni, se začne razvrščanje pogodb. Sam algoritem za hierarhično razvrščanje, ki se uporablja, je že bil opisan v sekciji 3.2.1, zato bo na tem mestu zgolj na kratko povzeto bistvo razvrščanja na primeru te problemske domene. Namen razvrščanja je, da množico vseh pogodb razbije na več gruč, pri čemer se pripadnost posamezne pogodbe določeni gruči določi glede na to, koliko je podobna preostalim pogodbam v gruči. Pogodbe znotraj gruče so si medsebojno bolj podobne, kot so podobne pogodbam iz ostalih gruč. Podobnost je definirana z izbrano mero podobnosti, ki se izračuna nad naborom izbranih atributov pogodbe (atribute predstavljajo

polja v podatkovni bazi).

Postopek razvrščanja se začne tako, da se najprej za vse pogodbe iz podatkovne baze prebere vrednosti za atribute, nad katerimi se bo izvajalo razvrščanje. Tabela 4.2 prikazuje seznam teh atributov.

Ker imajo vsi ti atributi različne razpone vrednosti, jih je treba normalizirati, tako da imajo pri izračunu mere podobnosti vsi enako težo. Za normalizacijo se uporabi metodo *z-score* [20], ki je definirana kot

$$z = \frac{x - \mu}{\sigma}, \tag{4.26}$$

kjer $x$ označuje vrednost atributa pri posamezni pogodbi, $\mu$ povprečno vrednost atributa na populaciji vseh pogodb in $\sigma$ standardno deviacijo na celotni populaciji. V bistvu so prejšnje vrednosti atributov pretvorjene v vrednosti, ki predstavljajo odklon vrednosti atributa od povprečne vrednosti, izražen v številu standardnih deviacij.

Poleg teh atributov se iz baze prebere še naslednje atribute: povprečne donosnosti za vsa štiri obdobja, vse štiri variacije ALV, verjetnost zaprtja, volatilnost in povprečne donose za vsa obdobja pred izvršitvijo transakcije. Ti atributi niso vključeni v nabor tistih atributov, ki sodelujejo pri razvrščanju, vendar so vključeni na koncu, ko se izračuna povprečna vrednost atributov, pri vsaki od ustvarjenih gruč. Na ta način predstavljajo relevantne značilnosti gruč.

Potem se ustvari t. i. posebno gručo. Te gruče za razliko od vseh ostalih gruč ne ustvari algoritem za hierarhično razvrščanje, ampak je ustvarjena ročno, še pred zagonom algoritma. V njej so vse tiste pogodbe, pri katerih gibanje vrednosti njihovega portfelja ustreza enemu izmed, v nadaljevanju navedenih, kriterijev. Kasneje, po zagonu algoritma za razvrščanje, so vanjo dodane še vse pogodbe iz tistih gruč, ki vsebujejo manj kot 30 pogodb. V prejšnjem stavku omenjeni kriteriji za gibanje vrednosti portfelja so trije, in sicer:

- V portfelju, v roku zadnjega leta dni, obstaja dnevna rast donosnosti,

| Ime | Vsebina |
|---|---|
| Age | Starost stranke, kateri pripada dotična pogodba |
| InflowFrequencyAvg | Število denarnih prilivov v zadnjem letu dni |
| OutflowFrequencyAvg | Število denarnih odlivov v zadnjem letu dni |
| InflowValueAvg | Povprečna vrednost denarnih prilivov v zadnjem letu dni |
| OutflowValueAvg | Povprečna vrednost denarnih odlivov v zadnjem letu dni |
| NumPositionAvg | Povprečno število finančnih instrumentov v portfelju za zadnje leto dni |
| DomesticShare | Delež finančnih instrumentov, ki kotirajo na tujih trgih |
| ValueAvg | Povprečna vrednost portfelja v zadnjem letu dni |
| BuyFrequency | Število nabavnih transakcij v zadnjem letu dni |
| SellFrequency | Število prodajnih transakcij v zadnjem letu dni |
| BuyValueAvg | Povprečna vrednost nabavne transakcije v zadnjem letu dni |
| SellValueAvg | Povprečna vrednost prodajne transakcije v zadnjem letu dni |

Tabela 4.2: Polja iz podatkovne baze, nad katerimi se izvaja razvrščanje.

večja ali enaka 50 %, izražena v neenačbi kot

$$(Vrednost_i - DenarniPriliv_i) >= 1.5 * Vrednost_{i-1}, \qquad (4.27)$$

kjer $Vrednost_i$ označuje vrednost portfelja na $i$ - ti dan, $DenarniPriliv_i$ vsoto denarnih prilivov na $i$ - ti dan in $Vrednost_{i-1}$ vrednost $i-1$ - ti dan. $DenarniPriliv_i$ je vračunan v $Vrednost_i$, zato ga je potrebno odšteti, da se lahko izračuna donosnost. Velja še dodaten pogoj, da obe vrednosti portfelja iz enačbe, ne smeta biti hkrati enaki nič, saj v tem primeru ustrezata pogoju, čeprav ni nobene spremembe v donosnosti portfelja.

- V portfelju, v roku zadnjega leta dni, obstaja padec dnevne donosnosti, večji ali enak 25 %, izražen v neenačbi kot

$$(Vrednost_i + DenarniOdliv_i) <= 0.75 * Vrednost_{i-1}, \qquad (4.28)$$

kjer $DenarniOdliv_i$ označuje vsoto odlivov na $i$-ti dan. Tudi ta je vračunana v vrednost, zato jo moramo prišteti. Velja enak dodaten pogoj, kot pri prejšnjem kriteriju.

- V portfelju, v roku zadnjega leta dni, obstaja datum, na katerega se je zgodil odliv, s katerim je vrednost portfelja padla na 0, in je vrednost portfelja po tem datumu več kot 10 zaporednih dni enaka 0.

Pogodbe, ki ustrezajo enemu izmed teh kriterijev, imajo, v večini primerov (zakaj ne v vseh primerih, bo opisano v nadaljevanju), napačno izračunane in izrazito previsoke 30-, 90-, 180- in 365-dnevne donosnosti portfelja. Pri prvih dveh kriterijih so ti skoki oz. padci donosnosti rezultat človeške napake, saj prilivi oz. odlivi niso bili zabeleženi v pravilni vrednosti ali sploh niso bili zabeleženi (ti se v aplikaciji *Shark*, od koder se prenašajo podatki o prilivih in odlivih, vnašajo ročno). Lahko je tudi posledica napačno vnesenega tečaja.

Tretji kriterij odpravi hibo *metode prilagojenega Dietz-a*, po kateri se računajo omenjene donosnosti, ki povzroči izračun napačnih donosnosti, če je vrednost portfelja že nekaj dni, vključno do datuma, na katerega se vrši

izračun, enaka nič. Ti kriteriji poskrbijo, da so takšne pogodbe, še pred zagonom algoritma razvrščanja, izključene v svojo gručo in tako ne kvarijo prikazane donosnosti pri ostalih gručah.

Žal ti kriteriji ne uspejo izluščiti vseh problematičnih pogodb ali zaobjamejo kakšne po krivici. Obstajajo primeri, ko so ti skoki oz. padci vrednosti rezultat dejanske spremembe vrednosti pozicije v portfelju (manj ima portfelj različnih pozicij, bolj je izpostavljen temu riziku, saj sprememba vrednosti posamezne pozicije sorazmerno bolj vpliva na spremembo vrednosti celotnega portfelja). Obstajajo takšni primeri, kjer skok oz. padec donosnosti ne ustreza pragu kriterijev, čeprav je rezultat napačnega vnosa prilivov oz. odlivov. Obe meji pri kriterijih za donosnost sta bili izbrani na podlagi preteklih izkušenj o gibanju vrednosti portfeljev, tako da v veliki večini primerov delujeta pravilno. Kriterije, ki bi popolnoma izluščili vse problematične pogodbe, je iz omenjenih razlogov nemogoče zasnovati.

Ko so bile problematične pogodbe izločene v posebno gručo, se nad preostalimi pogodbami zažene algoritem hierarhičnega razvrščanja. Pri tem se za mero podobnosti med posameznimi primeri uporabi kvadrirana evklidska razdalja, saj to terja funkcija *hclust*, ki je R-jeva implementacija algoritma hierarhičnega razvrščanja. Za mero podobnosti med gručami se uporabi *Wardovo metodo* [33].

Rezultat zagona algoritma je dendrogram. Pri njem se s pomočjo *L-metode* določi število gruč. V našem primeru se to določi na 14 gruč. Ko se določi število gruč, se v dendrogramu opravi odrez na takšni višini, da se dobi želeno število gruč. Nato se pogleda, katere gruče vsebujejo manj kot 30 pogodb, in se vse pogodbe iz teh gruč pridruži posebni gruči, tako da na koncu ostanejo samo gruče z več kot 30 pogodbami. Nam tako na koncu ostane skupno 6 gruč, skupaj s posebno gručo. Razlog za eliminacijo teh gruč je v tem, da pri tako majhnih gručah pogodbe s prej opisanimi anomalijami pri izračunanih donosnosti lahko znatno pokvarijo povprečno donosnost na nivoju gruče, če so v gruči.

Na koncu se za končen nabor gruč, pri vsaki od njih izračuna povprečne

| ClusterID | Volume | Age | IFA | OFA | ValueAvg | OnlineOrderShare |
|---|---|---|---|---|---|---|
| 1 | 1381 | 32,82 | 1,00 | 0,15 | 41.819 | 0 |
| 2 | 1315 | 60,44 | 0,99 | 0,14 | 14.701 | 0 |
| 3 | 338 | 46,69 | 1,54 | 21,91 | 62.551 | 0,002 |
| 4 | 415 | 39,90 | 1,05 | 2,35 | 13.659 | 0,018 |
| 5 | 259 | 38,88 | 6,10 | 6,12 | 21.770 | 0,959 |
| 6 | 1032 | 44,49 | 5,18 | 4,67 | 19.415 | 0,146 |

Tabela 4.3: Izbrana polja končnega nabora gruč. IFA in OFA sta okrajšavi za InflowFrequencyAvg in OutflowFrequencyAvg. Zadnja, gruča številka 6, predstavlja posebno gručo.

vrednosti za vse atribute. Nad temi povprečji se potem še opravi normalizacija z *z-score* metodo. Gruče se nato zaporedno oštevilči in se jih skupaj z vrednostmi njihovih atributov shrani v tabelo *Cluster*. Potem še se vsaki pogodbi v tabeli *CustomerAccout* v polje *ClusterID* vpiše številko gruče, ki ji pripada.

V tabeli 4.3 je prikazan končni nabor gruč pri razvrščanju naših podatkov. Za vsako gručo je prikazanih nekaj izbranih stolpcev, iz katerih so razvidna odstopanja posameznih gruč. Na podlagi tega lahko sklepamo, glede na katera polja je algoritem razvrščal primere v gruče. Spodaj si bomo ogledali kratko analizo teh podatkov. Vanjo ne bomo vključevali 6. gruče, čeprav je prikazana v tabeli, saj predstavlja, zgoraj opisano, posebno gručo. Ker so primeri vanjo izbrani na podlagi ročno postavljenih kriterijev, ta ne odraža strukture podatkov tako kot preostale gruče, ki so neposredni rezultat hierarhičnega razvrščanja.

Pri povprečni starosti članov izstopa 2. gruča, kjer povprečje doseže 60,44 let. Po povprečnem številu denarnih prilivov izstopa 5. gruča, kjer se jih je v preteklem letu zgodilo v povprečju 6,1, medtem ko se ta vrednost pri preostalih gručah giblje okoli 1,0. Po povprečnem številu denarnih odlivov imamo odstopanja v obe smeri; v pozitivno zelo močno izstopa 3. gruča,

v negativno pa prvi dve gruči. Podoben vzorec se pojavlja pri zadnjem faktorju, kjer imata prve dve gruči 0% delež spletnih naročil, medtem ko je pri 5. gruči ta 95,9%.

## 4.4 Prikaz podatkov

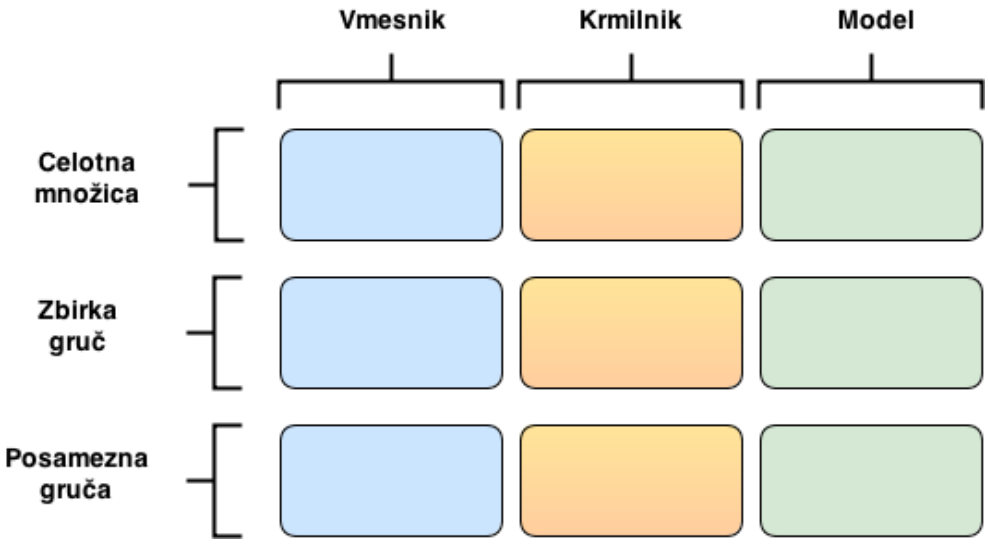
Prikaz podatkov je implementiran v obliki spletne aplikacije. Aplikacija sestoji iz treh funkcionalnostnih sklopov, kjer posamezni sklop izvaja agregacijo in omogoča pregled na nivoju:

- celotne množice pogodb (npr. število vseh pogodb BPD, skupna vsota vseh sredstev ...),
- zbirke gruč (tukaj se opravijo razne primerjave med gručami, glede na določene podatke) in
- posamezne gruče (pregled značilnosti za izbrano gručo).

Osrednjo tehnologijo aplikacije predstavlja *ASP.NET MVC*, saj je z njim zgrajeno celotno ogrodje. Ta tehnologija sledi programski arhitekturni šabloni *MVC*. Kot je razvidno s slike 4.7, ima vsak od sklopov svojo komponento v vseh treh delih *MVC*.

Model vsebuje poizvedbe, ki pridobivajo podatke iz podatkovne baze in potem interno hrani rezultate teh poizvedb. Pri tem ima najpomembnejšo vlogo *LINQ* v navezi z ogrodjem *Entity Framework*. Prvi služi za tvorbo poizvedb, drugi abstrahira podatkovno bazo. Vmesnik predstavlja uporabniško masko, preko katere se grafično prikažejo vsi podatki iz modela in se uporabniku omogoči interakcija z aplikacijo. Za definiranje strukture in splošnega videza uporabniške maske oz. spletne strani se uporabljajo *HTML* in *CSS* ter *JavaScript* za interakcijo in razne grafične prikaze. Krmilnik igra vlogo posrednika med modelom in vmesnikom, tako da izbere ustrezen vmesnik in ga poveže s pravim modelom, iz katerega se pridobijo podatki za prikaz.

Sledijo opisi vsake od uporabniških mask, v sklopu katerih je tudi opisana vsebina podatkov, ki jih prikazujejo.




Slika 4.7: Prikaz MVC-komponent.

### 4.4.1 Pregled celotne množice pogodb

Kot že omenjeno, ta uporabniška maska prikazuje podatke na ravni celotne množice pogodb. Skoraj vse kategorije podatkov so prikazane v obliki grafov, kot je razvidno s slike 4.8. Če sledimo vrstnemu redu grafov na sliki 4.8, potem je njihova vsebina naslednja:

- **Prihodki:** v obliki črtnega grafa sta prikazani dve časovni vrsti. Prva prikazuje skupno vsoto vseh prihodkov BPD za preteklo leto (to se vedno gleda od datuma zadnjega prenosa podatkov v živo podatkovno bazo). Pod prihodke spadajo vsi tipi provizij. Druga vrsta prikazuje razmerje med prihodki za preteklo leto in prihodki za leto pred tem.

  Razmik med posamezni točkami znotraj časovna vrste je en teden, torej vsaka od vrst vsebuje 52 točk. Posamezna točka sestoji iz datuma in pripadajoče vrednosti, pri čemer datumi določajo lego točke, vzdolž horizontalne osi grafa, in vrednosti, lego vzdolž vertikalne osi. Pri vsoti prihodkov se za vrednost posamezne točke vzame vsota prihodkov za pretekli teden, gledano od datuma točke. Pri vrsti razmerij se za vre-

| Datum | Prihodki | Od | Do |
|---|---|---|---|
| 30. 5. 2013 | 10.000 | 23. 5. 2013 | 30. 5. 2013 |
| 23. 5. 2013 | 13.000 | 16. 5. 2013 | 23. 5. 2013 |
| ... | ... | ... | ... |
| 30. 5. 2012 | 22.000 | 23. 5. 2012 | 30. 5. 2012 |

Tabela 4.4: Primer časovne vrste prihodkov. Prvi stolpec prikazuje datum točke, drugi njeno vrednost in zadnja dva stolpca obdobje, na katerega se nanaša izračunana vrednost.

dnost točke vzame razmerje med vsoto prihodkov na datum v preteklem letu in v letu pred tem. Vsota prihodkov je definirana enako kot pri prejšnji vrsti. Za jasnejšo predstavo tabela 4.4 prikazuje izsek iz primera časovne vrste z dodanim obdobjem, v katerem se računa vrednost točke.

Na levi strani grafa so trije semaforji, ki v odstotkih prikazujejo razmerje med prihodki v zadnjih 3 mesecih preteklega leta in prihodki v enakem obdobju za eno od prejšnjih let. Najvišji primerja s predpreteklim letom, srednji poveča zamik v preteklost za leto dni in najnižji še za dodatno leto. Na desni strani si lahko izberemo časovni razpon, ki naj ga graf prikazuje. Izbiramo med prikazom za zadnje 3, 6 ali 12 mesecev.

- **Sredstva in denarni tok:** v obliki črtnega grafa sta prikazani dve časovni vrsti. Prva prikazuje gibanje vsote sredstev za preteklo leto in druga gibanje vsote denarnega toka za enako obdobje. Razmik med točkami je pri obeh vrstah en teden. Pri obeh vrstah točka sestoji iz datuma in vrednosti. Pri vrsti sredstev vsaka vrednost točke prikazuje vsoto sredstev vseh pogodb za posredovanje in gospodarjenje, ki so jih pogodbe imele na datum točke. Pri vrsti denarnega toka vrednost točke predstavlja vsoto denarnih prilivov in odlivov za pretekli teden,

gledano od datuma točke.

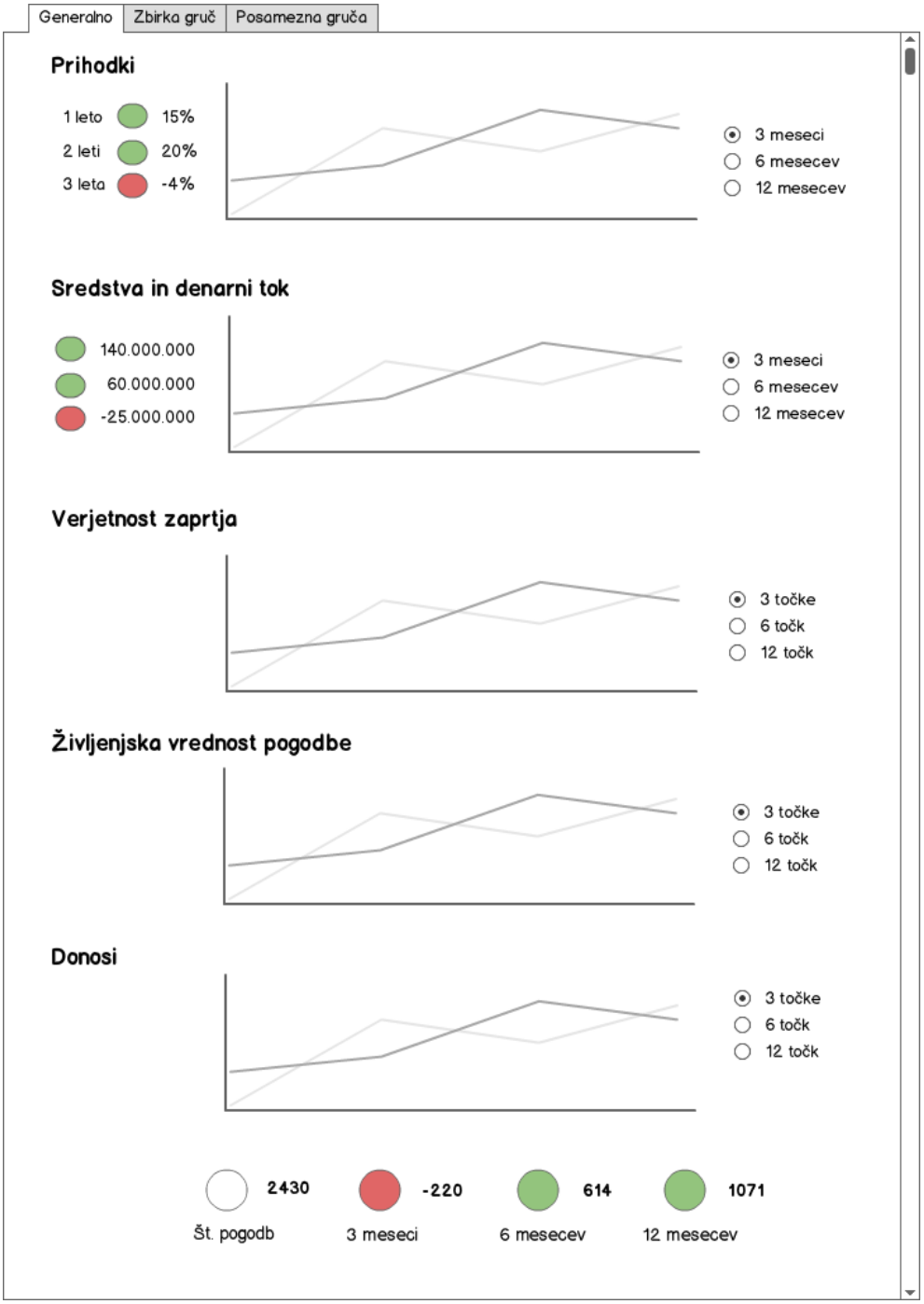
Na levi strani grafa so trije semaforji, ki prikazujejo spremembo vsote sredstev v različnih obdobjih. Najvišji prikazuje spremembo za zadnji mesec, semafor pod njim za zadnje 3 mesece in najnižji za zadnjih 6 mesecev. Na desni strani je opcija za izbiro časovnega razpona grafa, ki deluje enako kot pri prejšnjem grafu.

- **Verjetnosti zaprtja pogodbe:** v obliki črtnega grafa je prikazana časovna vrsta, ki vsebuje zadnjih 12 izračunov povprečne verjetnosti zaprtja pogodbe. Posamezna točka sestoji iz datuma, na katerega je bil izračun opravljen, in vrednosti, ki hrani izračunano povprečno verjetnost zaprtja pogodbe. Na desni strani se lahko določi časovni razpon grafa, kjer lahko izbiramo med prikazi zadnjih 3, 6 ali 12 izračunov.

- **Življenjska vrednost pogodbe:** v obliki črtnega grafa so prikazane časovne vrste povprečne življenjske vrednosti pogodbe za vse štiri variacije, od katerih vsaka vsebuje zadnjih 12 izračunov. Posamezna točka sestoji iz datuma izračuna in vrednosti, ki hrani izračunano življenjsko vrednost pogodbe. Na desni strani grafa imamo enake opcije kot pri prejšnjem grafu.

- **Donosi:** v obliki črtnega grafa so prikazane časovne vrste za izračunane donosnosti portfeljev za vsa štiri obdobja v preteklem letu. Posamezna točka sestoji iz datuma, na katerega je bila donosnost izračunana, in vrednosti, ki hrani donosnost. Razmik med točkami je en mesec, pri vseh vrstah. Za večjo jasnost, kako je videti posamezna časovna serija, tabela 4.5 prikazuje primer vrste 90-dnevnih donosnosti.

- **Število pogodb:** v tekstovni obliki so prikazani skupno število pogodb in trije semaforji za spremembo števila pogodb v določenem obdobju. Prvi semafor z leve prikazuje obdobje zadnjih 3 mesecev, naslednji zadnjih 6 mesecev in zadnji 12 mesecev.




Slika 4.8: Uporabniška maska za pregled celotne množice pogodb.

| Datum | Donosnost | Od | Do |
|---|---|---|---|
| 30.5.2013 | 6,2 % | 30.2.2013 | 30.5.2013 |
| 30.4.2013 | 7,1 % | 30.1.2013 | 30.4.2013 |
| ... | ... | ... | ... |
| 30.5.2012 | -2,7 % | 30.2.2012 | 30.5.2012 |

Tabela 4.5: Primer časovne vrste 90-dnevnih donosnosti. Prvi stolpec prikazuje datum izračuna, drugi izračunano donosnost in zadnja dva stolpca obdobje, na katerega se nanaša izračunana donosnost.

### 4.4.2 Pregled zbirke gruč

Ta uporabniška maska prikazuje podatke na nivoju zbirke gruč. Prikazani so agregirani podatki za vsako od gruč. Podobno kot na prejšnji maski je velika večina prikazov v obliki grafov, kot je razvidno s slike 4.9. Če sledimo vrstnemu redu grafov na sliki 4.9, potem je njihova vsebina naslednja:

- **Število pogodb:** v obliki stolpčnega grafa je za vsako gručo prikazano število pogodb, ki jih vsebuje. Vsak stolpec predstavlja svojo gručo in njegova višina označuje število pogodb.
- **Verjetnost zaprtja pogodbe:** v obliki stolpčnega grafa je za vsako gručo prikazana povprečna verjetnost zaprtja pogodbe za zadnje tri izračune. Vsako gručo predstavlja skupina treh stolpcev, od katerih vsak prikazuje povprečje za enega od izračunov. Višina stolpca označuje velikost verjetnosti.
- **Življenjska vrednost pogodbe:** v obliki stolpčnega grafa je za vsako gručo prikazana vsota zadnjih izračunanih življenjskih vrednosti vseh pogodb v njej. Vsak stolpec predstavlja svojo gručo in višina stolpca označuje velikost vrednosti.
- **Sredstva:** v obliki tortnega grafikona je prikazana vsota sredstev v vsaki gruči. Vsak del tortnega grafikona predstavlja eno gručo in nje-

gova velikost je sorazmerna z deležem celotnih sredstev, ki so v tej gruči.

- **Prihodki:** v obliki črtnega grafa so prikazane časovne vrste prihodkov, za vsako od gruč. Vsebinsko je posamezna časovna vrsta enaka kot pri časovni vrsti pri pregledu celotne množice pogodb, le da se tukaj upošteva zgolj prihodke, ki jih je ustvarila posamezna gruča. Na levi strani grafa je mogoče vklapljati in izklapljati prikaz prihodkov za posamezno gručo.

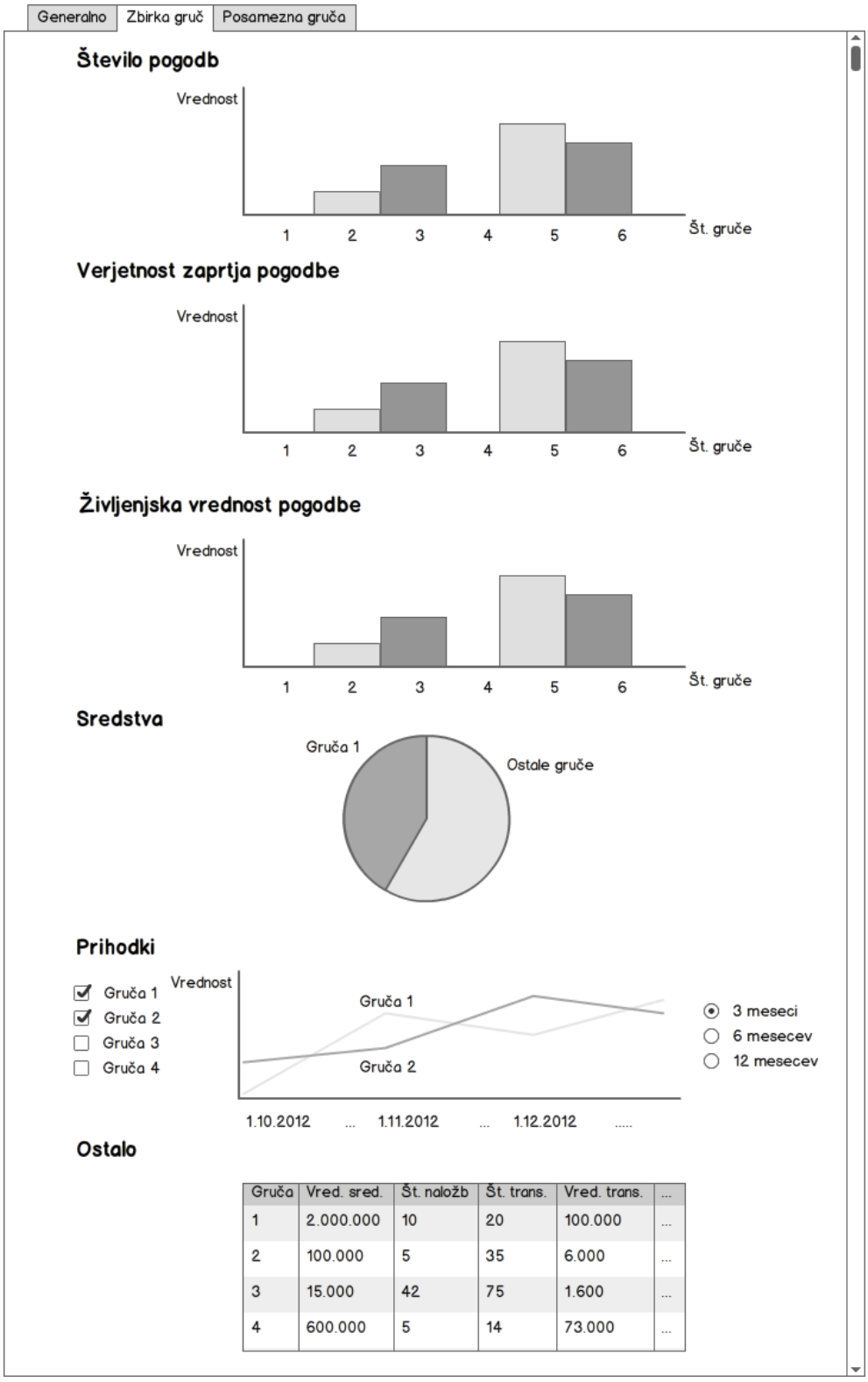
- **Ostalo:** v obliki tabele je prikazan izbran nabor preostalih podatkov. Vsaka vrstica predstavlja svojo gručo in v njej so prikazane povprečne vrednosti v gruči za sledeče podatke: vrednost sredstev v portfelju, število naložb, število transakcij, vrednost transakcij, delež spletnih naročil, vse štiri donosnosti, volatilnost in delež domačih naložb.

### 4.4.3 Pregled posamezne gruče

Ta uporabniška maska prikazuje agregirane podatke za izbrano gručo. Podatki so prikazani s pomočjo grafov in tabel. Njihovo postavitev prikazuje slika 4.10. V zgornjem desnem kotu si lahko izberemo, za katero gručo si želimo ogledati podatke. Če sledimo prikazanemu vrstnemu redu, potem je vsebina sekcij sledeča:

- **Največ vredne pogodbe:** v obliki tabele je prikazanih pet pogodb iz množice največ vrednih pogodb. Ta množica sestoji iz 10 % največ vrednih pogodb, glede na vrednost sredstev v portfelju. Iz nje se periodično, na vsakih 30 sekund, naključno izbere in prikaže pet pogodb. V tabeli so za vsako pogodbo prikazani: ime imetnika pogodbe, trenutna vrednost sredstev v portfelju in vrednost vseh prihodkov, ki so bili ustvarjeni na račun pogodbe do sedaj.

- **Prihodki:** v obliki črtnega grafa je prikazana časovna vrsta prihodkov za izbrano gručo. Časovna vrsta zaobjema prihodke za preteklo




Slika 4.9: Uporabniška maska za pregled zbirke gruč.

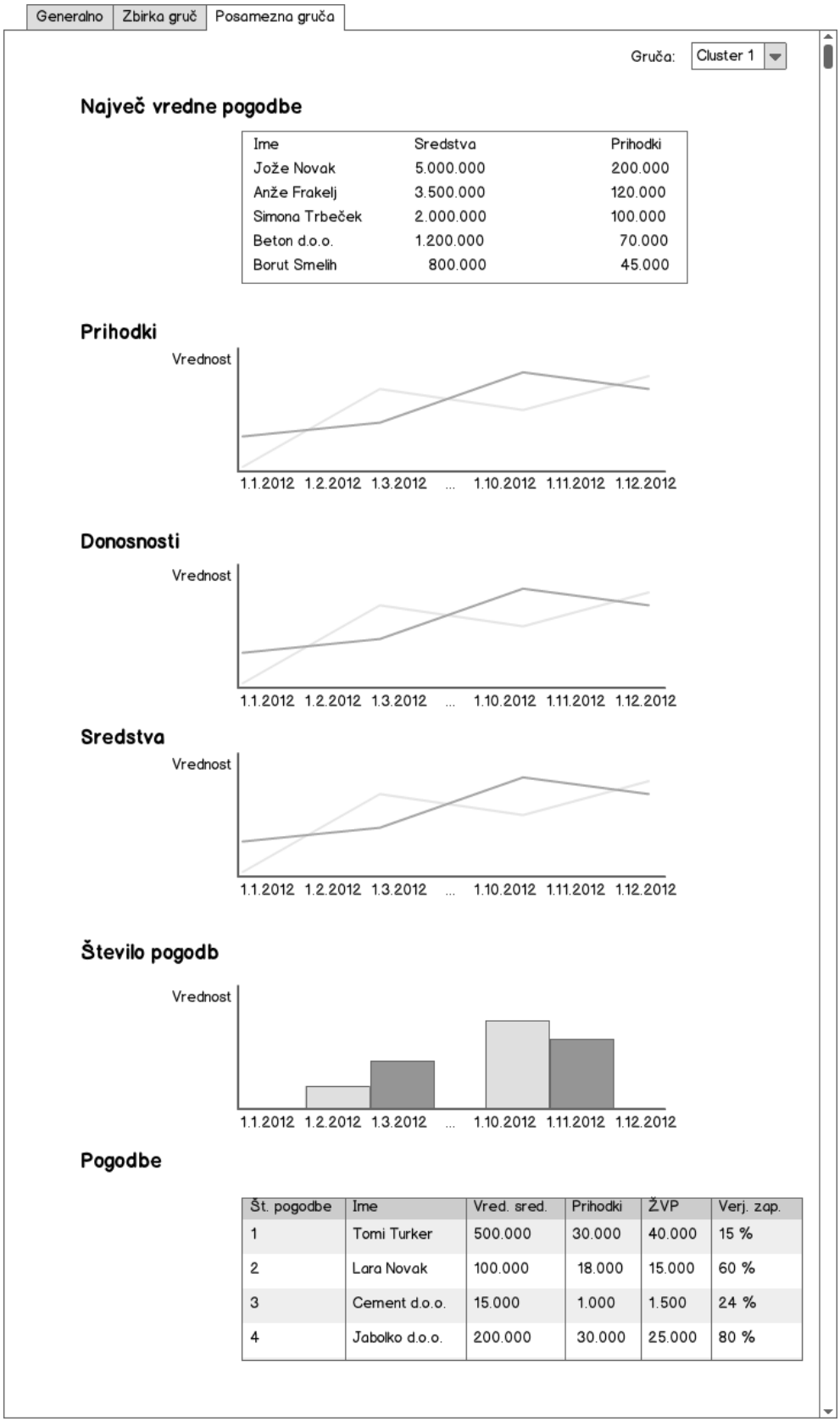
leto in sestoji iz 12 točk, z enomesečnim medsebojnim razmikom. Posamezna točka sestoji iz datuma in pripadajoče vrednosti prihodkov. Datumi določajo lego točke, vzdolž horizontalne osi grafa, in vrednost lego, vzdolž vertikalne osi. Za vrednost prihodkov se vzame vsoto vseh prihodkov, ki so jih ustvarile pogodbe v gruči za pretekli mesec dni, gledano od datuma točke.

- **Donosnosti:** v obliki črtnega grafa so prikazane časovne vrste povprečnih donosnosti za vsa štiri obdobja, za vsako od gruč. Vsebinsko je posamezna časovna vrsta enaka časovni vrsti pri pregledu celotne množice pogodb, le da se tukaj povprečne donosnosti računa za izbrano gručo.

- **Sredstva:** v obliki črtnega grafa je prikazana časovna vrsta sredstev za izbrano gručo. Časovna vrsta je enaka tisti pri prihodkih, s to razliko, da tukaj pripadajočo vrednost predstavlja vsota sredstev v gruči na datum od točke.

- **Število pogodb:** v obliki stolpčnega grafa je prikazana časovna vrsta števila pogodb za izbrano gručo. Časovna vrsta je enaka tisti pri prihodkih, s to razliko, da tukaj pripadajočo vrednost predstavlja število pogodb v gruči na dani datum. To določa višino posameznega stolpca.

- **Pogodbe:** v obliki tabele so prikazane vse pogodbe za izbrano gručo. Za posamezno pogodbo so navedeni naslednji podatki: ime imetnika pogodbe, trenutna vrednost sredstev v portfelju, vrednost vseh ustvarjenih prihodkov, življenjska vrednost pogodbe in verjetnost zaprtja pogodbe. Tabela tudi omogoča sortiranje pogodb po kateremkoli od stolpcev.




Slika 4.10: Uporabniška maska za pregled posamezne gruče.

# Poglavje 5

# Sklepne ugotovitve

Cilj diplomske naloge je bila izdelava načrta IS za analizo podatkov BPD in izdelava prototipa. Najprej je bilo v uvodnem opisu zaobjeto bistvo našega IS. Nato smo si ogledali primera rešitev, ki se po svojih funkcionalnostih lahko štejeta kot sorodna naši, in problemsko domeno, v kateri so podatki, nad katerimi izvajamo analize. V tretjem poglavju je sledil kratek pregled tehnologij, ki so bile uporabljene pri gradnji IS. V tem poglavju smo si tudi podrobneje ogledali teoretično ozadje algoritmov strojnega učenja, ki igrajo ključno vlogo v našem IS. Na koncu je sledil še pregled same zgradbe in implementacije IS.

Ključni problem so predstavljali podatki, saj so ti vsebovali kar nekaj anomalij in pomanjkljivosti, ki so nam iznakazile rezultate oz. onemogočile njihovo pridobivanje pri preizkušanju nekaterih funkcionalnosti. Kot je bilo opisano v prejšnjem poglavju, smo se z anomalijami srečevali pri izračunih donosnosti in gručenju portfeljev, kjer je bil problem z napačnimi vrednostmi portfeljev in pomanjkljivo beleženimi denarnimi tokovi. Ta problem smo sicer lahko v večjem delu sanirali s filtri, ki so ujeli problematične primere, vendar jih žal ni bilo mogoče zastaviti tako, da bi bili sposobni ujeti vse primere s temi anomalijami. Največjo pomanjkljivost podatkov so predstavljale časovne vrste tečajev. Le za zelo majhen delež finančnih instrumentov so obstajale dovolj dolge časovne vrste tečajev, da so ustrezale potrebam izračunov

volatilnosti portfeljev in donosnosti finančnih instrumentov pred izvršitvijo transakcije. Iz tega razloga pri teh dveh funkcionalnostih ni bilo mogoče pridobiti smiselnih rezultatov. Na drugi strani smo dobili dobre rezultate pri napovedovanju zaprtij in gručenju pogodb. Pri prvem se je izkazalo, da klasifikator deluje z visoko natančnostjo, pri gručenju pa smo dobili gruče, ki dobro odražajo strukturo podatkov.

Zaradi svoje modularne zgradbe ima IS še kar nekaj potenciala za nadaljnje razširitve. Kot primer lahko navedemo generiranje priporočil za nakup in prodajo finančnih instrumentov. Ta bi se ustvarjala za posamezen portfelj, s ciljem optimizacije določenih kriterijev kot npr. donosnosti in volatilnosti portfelja. Izbrani finančni instrumenti bi se potem v obliki priporočila za nakup oz. prodajo dotičnega finančnega instrumenta poslali imetniku portfelja v obliki elektronske pošte.

# Literatura


[1] A Relational Database Overview. Dostopno na: http://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

[2] ASP.NET MVC Overview. Dostopno na: http://msdn.microsoft.com/en-us/library/dd381412(VS.98).aspx

[3] Covariance. Dostopno na: http://www.math.uah.edu/stat/sample/Covariance.html

[4] CRAN: CARET. Dostopno na: http://cran.r-project.org/web/packages/caret/index.html

[5] Entity Framework Overview. Dostopno na: http://msdn.microsoft.com/en-us/library/bb399567.aspx

[6] Evolve. Dostopno na: http://www.evolve.si/

[7] Flot. Dostopno na: http://www.flotcharts.org/

[8] HTML, CSS. Dostopno na: http://www.w3.org/

[9] Javascript. Dostopno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[10] jQuery. Dostopno na: http://jquery.com/

[11] PortfolioMonkey. Dostopno na: https://www.portfoliomonkey.com/

[12] PRM Exam Preparation. Dostopno na:
https://www.riskprep.com/all-tutorials/36-exam-22/58-modeling-portfolio-variance

[13] Root mean square error. Dostopno na:
http://www-stat.stanford.edu/~susan/courses/s60/split/node60.html

[14] SAP: SAP Predictive Analysis. Dostopno na:
http://www54.sap.com/pc/analytics/business-intelligence/software/predictive-analysis/index.html

[15] Wikipedia: Hierarchical clustering. Dostopno na:
http://en.wikipedia.org/wiki/Hierarchical_clustering

[16] Wikipedia: Modified Dietz method. Dostopno na:
http://en.wikipedia.org/wiki/Modified_Dietz_method

[17] Wikipedia: Financial instrument. Dostopno na:
http://en.wikipedia.org/wiki/Financial_instrument

[18] Wikipedia: Accuracy and precision. Dostopno na:
http://en.wikipedia.org/wiki/Accuracy_and_precision

[19] Wolfram MathWorld: Least squares fitting. Dostopno na:
http://mathworld.wolfram.com/LeastSquaresFitting.html

[20] Wolfram MathWorld: z-Score. Dostopno na:
http://mathworld.wolfram.com/z-Score.html

[21] J. Albahari, B. Albahari, *C# 4.0 In A Nutshell: The Definitive Reference*, Sebastopol, California: O'Reilly, 2010, pogl. 8.

[22] L. Breiman, "Random Forests," *Machine Learning*, št. 45, zv. 1, str. 5-32, 2001.

[23] L. Breiman, A. Cutler. *Random forests*. Dostopno na: `http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`

[24] L. Breiman: Manual on Setting up, Using and Understanding Random Forests v4.0. Dostopno na: `http://oz.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf`

[25] B. Davis, J. Menchero. Risk Contribution is Exposure times Volatility times Correlation. Dostopno na: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.7503&rep=rep1&type=pdf`

[26] (2001) G. Fung. A Comprehensive Overview Of Basic Clustering Algorithms. Dostopno na: `http://pages.cs.wisc.edu/~gfung/clustering.pdf`

[27] S. Gupta et al., "Modeling Customer Lifetime Value," *Journal of Service Research*, št. 9, zv. 2, str. 139-155, 2006.

[28] A.K. Jain et al., "Data Clustering: A Review," *ACM Computing Surveys*, št. 31, zv. 3, str. 264-323, 1999.

[29] S.B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Frontiers in Artificial Intelligence and Applications*, št. 160, zv. 1, str. 3-24, 2007.

[30] A.G. Lalkhen, A. McCluskey, "Clinical Tests: Sensitivity and Specificity," *CEACCP*, št. 8, zv. 6, str. 221-223, 2008.

[31] H.M. Markowitz, *Portfolio Selection Efficient Diversification of Investments*, New York: John Wiley & Sons Inc., 1959, pogl. 4

[32] S. Salvador, P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms", v zborniku *Proc. 16th IEEE Intl. Conf. on Tools with AI*, Boca Raton, Florida, ZDA, nov. 2004, str. 576-584.

[33] (2009) C. Shalizi: Distances between clustering, hierarchical clustering. Dostopno na: http://www.stat.cmu.edu/~cshalizi/350/lectures/08/lecture-08.pdf

[34] P.N. Tan et al., *Introduction to Data Mining*, Boston: Addison-Wesley, 2006, pogl. 4.