UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

# ANALIZA UPORABNOSTI ODPRTOKODNIH REŠITEV ZA PODATKOVNO RUDARJENJE: PRIMER ANALIZE PREHAJANJA NAROČNIKOV V TELEKOMUNIKACIJSKEM PODJETJU

Ljubljana, december 2014 PETER MOČNIK

IZJAVA O AVTORSTVU

Spodaj podpisani Peter Močnik, študent Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor zaključne strokovne naloge magistrskega dela z naslovom Analiza uporabnosti odprtokodnih rešitev za podatkovno rudarjenje: Primer analize prehajanja naročnikov v telekomunikacijskem podjetju, pripravljenega v sodelovanju s svetovalcem prof. dr. Jurijem Jakličem.
Izrecno izjavljam, da v skladu z določili Zakona o avtorski in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo zaključne strokovne naloge magistrskega dela na fakultetnih spletnih straneh.
S svojim podpisom zagotavljam, da

- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem
    - poskrbel, da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v zaključni strokovni nalogi magistrskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobil vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v tekstu, in sem to v besedilu tudi jasno zapisal;
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predložene zaključne strokovne naloge magistrskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne 23.12.2014 Podpis avtorja:_____

# KAZALO

**UVOD** 1

**1 TRŽENJSKI IZZIV** 4

1.1 Zasičenost trga 6

1.2 Prehajanje strank h konkurenci 7

**2 POSLOVNA INTELIGENCA** 10

2.1 CRISP model podatkovnega rudarjenja 12

2.2 Management odnosov z odjemalci 14

2.3 Programske rešitve za management odnosov z odjemalci 15

2.4 Ocena uporabnikov 15

**3 MASOVNI PODATKI** 16

3.1 Podatki in čas hranjenja 16

3.2 Viri podatkov 16

3.3 Združevanje podatkov 17

**4 PROGRAMSKA OPREMA ZA PODATKOVNO RUDARJENJE** 18

4.1 Primerjava med lastniškimi programskimi rešitvami za podatkovno rudarjenje 21

4.2 Odprtokodna programska oprema za podatkovno rudarjenje 22

4.2.1 Kriteriji odprtokodne programske opreme 22

4.2.2 Prednosti odprtokodne programske opreme 23

4.3 Primerjava med odprtokodnimi programskimi rešitvami za podatkovno rudarjenje 24

**5 PODATKOVNO RUDARJENJE** 25

5.1 Klasifikacijske metode 25

5.2 Metode podatkovnega rudarjenja za napovedovanje prehoda strank h konkurenci 26

5.2.1 Odločitvena drevesa 27

5.2.2 Naivni Bayesov klasifikator 27

5.2.3 Nevronske mreže 27

5.2.4 Regresijski modeli 28

5.3 Logistična regresija 28

5.4 Logistična regresija kot napovedni model 31

5.5 Učni sistem in klasifikator 31

**6 IZDELAVA REŠITVE** 33

6.1 Pregled uporabljenih orodij 34

6.1.1 Programski jezik Python in grafične knjižnice Qt 35

6.1.2 Programska rešitev za podatkovno rudarjenje Orange 35
6.1.3 Spletno ogrodje Web2py 36
6.2 Poraba pomnilnika 37
6.3 Viri podatkov 38
6.4 Priprava podatkov za sistem Orange 39
6.5 Modeliranje v aplikaciji Orange 39
6.5.1 Razred logistične regresije v paketu Orange 42
6.5.2 Arhitektura rešitve 43
6.5.3 Uporaba programske kode 44
6.5.4 Web2Py koda s komentarji 46
6.6 Končni izgled aplikacije 48
**7 ANALIZA USTREZNOSTI REŠITVE 51**
**SKLEP 54**
**LITERATURA IN VIRI 57**

# KAZALO TABEL

Tabela 1: Lastniški in odprtokodni programi za podatkovno rudarjenje ..... 20
Tabela 2: Maksimalna količina razpoložljivega pomnilnika pri različnih tipih operacijskega sistema Windows..... 38

# KAZALO SLIK

Slika 1: Življenjski cikel novih tehnologij ..... 5
Slika 2: Penetracija aktivnih uporabnikov mobilne telefonije na prebivalstvo ..... 6
Slika 3: Število prenosov mobilnih telefonskih številk k operaterjem ..... 7
Slika 4: Razlogi za prehajanje h konkurenci ..... 8
Slika 5: Področja uporabe poslovne inteligence ..... 10
Slika 6: CRISP-DM model ..... 14
Slika 7: Proces združevanja več virov podatkov v eno podatkovno skladišče, ki se uporablja za podatkovno rudarjenje (ETL) ..... 18
Slika 8: Gartnerjev kvadrant za primerjavo lastniške programske opreme za podatkovno rudarjenje ..... 21
Slika 9: Algoritmi za podatkovno rudarjenje ..... 26
Slika 10: Funkcija logistične regresije ..... 29
Slika 11: Shema sodelujočih entitet pri klasifikaciji ..... 33
Slika 12: Shema delovanja MVC aplikacij ..... 37
Slika 13: Primer glave datoteke, ki je zapisana v prvih treh vrsticah datoteke ..... 39
Slika 14: Ocenjevanje kvalitete modelov v aplikaciji Orange ..... 40
Slika 15: Primerjava uspešnosti različnih napovednih modelov ..... 41
Slika 16: Matrika zmot za logistično regresijo ..... 41
Slika 17: Matrika zmot za odločitveno drevo ..... 41
Slika 18: ROC krivulja za primerjavo uspešnosti napovednih modelov ..... 42
Slika 19: Arhitektura celotne rešitve s sodelujočimi komponentami ..... 44
Slika 20: Okno z nastavitvami spletnega strežnika ..... 48
Slika 21: Spletna stran za uporabo celotne rešitve ..... 49

# UVOD

Količina podatkov na svetu se vsako leto podvoji (Bradicich, 2013). V letu 2011 je nastalo 1,8 zeta bajtov ali $10^{21}$ bajtov podatkov (Gantz & Reinsel, 2011). To je tako velika količina, da je ljudje sami ne bomo mogli nikoli obdelati. Zdaj ko živimo v tako imenovanem obdobju masovnih podatkov (angl. *big data*), se bolj kot kadarkoli prej poraja vprašanje, kaj z vsemi temi informacijami narediti. Ena od ved, ki se s tem ukvarja, se imenuje podatkovno rudarjenje (angl. *data mining*). Njena orodja so statistika, umetna inteligenca in strojno učenje, cilj pa iskanje vzorcev v podatkih (Chakrabarti et al., 2004).

Podatkovno rudarjenje je prisotno povsod, kjer si lahko zamislimo, od znanosti in raziskovanja do potrošništva in zabave. Kot zelo uporabno se je izkazalo tudi na področju trženja, kjer se osredotoča na obnašanje uporabnikov oz. potrošnikov. Zaradi velike količine podatkov, ki se zbirajo in nato obdelujejo, je podatkovno rudarjenje precej zahtevno. Za obdelovanje velikih količin podatkov obstajajo razne statistične metode, kar pa ni dovolj. Za uporabo teh metod nad velikimi količinami podatkov potrebujemo specializirano programsko opremo, s katero je te analize mogoče narediti. Večina programske opreme za podatkovno rudarjenje so razvili veliki proizvajalci programske opreme za prav tako velike uporabnike. Ta programska oprema je visoko specializirana za obdelavo velikih količin podatkov in temu primerno draga. Alternativa lastniškim rešitvam velikih proizvajalcev so odprtokodne rešitve. Večinoma so bile razvite v raziskovalne namene, vendar vsebujejo vse pomembne elemente potrebne za podatkovno rudarjenje nad velikimi količinami podatkov.

Odprtokodne rešitve so postale že del našega vsakdana. Pred leti smo na njih gledali z določeno mero nezaupanja, danes pa je realnost povsem drugačna. Kot dobro poznano odprtokodno rešitev velja omeniti operacijski sistem Linux, ki je med najbolj zaupanja vrednimi projekti, ki izhajajo iz odprtokodnega sveta.

Dobre prakse, ki so pripeljale Linux do trenutnega statusa, so se prijele in izkazale za zelo uspešne. Ne samo, da omogočajo dober, delujoč sistem, ampak so tudi brezplačne v smislu nabavne cene, kar pa ne pomeni, da z njihovo uporabo nimamo nobenih stroškov. V poročilu (Johnson, 2008) trdijo, da so uporabniki v globalnem smislu na ta način privarčevali 60 milijard ameriških dolarjev na leto. Danes je odprtokodnih rešitev toliko, da jih ni mogoče niti našteti in so alternativa na skoraj vseh področjih, kjer obstaja tudi lastniška programska oprema.

Ena izmed slabosti odprtokodne programske opreme je pomanjkanje celovite podpore in strukturiranega načina implementacije za rešitve določenega problema, ker je pač brezplačna, čeprav obstajajo tudi podjetja, ki se ukvarjajo z

implementacijami brezplačnih rešitev. Kot prednost pri uporabi odprtokodne programske opreme lahko sicer navedemo široko podporo skupnosti, ki se s tem ukvarja in spletne forume na katerih lahko najdemo ogromno informacij v zvezi s podporo rešitvam, vseeno pa moramo bistven del implementacije opraviti sami. Za uspešno implementacijo odprtokodnih rešitev je potrebno veliko znanja s področja integracije sistemov in tudi nekaj programiranja.

V tej nalogi se bomo osredotočili na raziskovanje obnašanja uporabnikov mobilne telefonije. Telekomunikacijski trgi so v Sloveniji in Evropi v letu 2014 že praktično popolnoma zasičeni. Zaradi tega lahko sklepamo, da nadaljnja rast, kateri smo bili priča vse od devetdesetih let prejšnjega stoletja, ni več mogoča oziroma, da naročnikov na mobilno telefonijo ni več mogoče pridobivati iz baze prebivalcev brez naročnine. Ponudniki mobilne telefonije so se za svoje stranke primorani boriti na drugačne načine in eden iz med njih je tendenca, da se stranko obdrži preden preide h konkurenčnemu operaterju, saj naj bi bilo pridobivanje nove stranke šestkrat dražje, kot pa zadržanje obstoječe (Pareek, 2007).

Za napovedovanje prehoda stranke h konkurenčnemu operaterju se uporabljajo statistične metode, s katerimi izračunamo, kakšna je verjetnost da bo stranka zamenjala operaterja. Ko enkrat ta podatek podjetje ima, se lahko odloči, na kakšen način bo poskušalo stranko zadržati, kar pa ne sodi več v okvir te naloge.
V nalogi smo si zastavili vprašanje, ali so odprtokodne rešitve že dovolj dodelane in uporabne, da bi jih bilo mogoče uporabiti za podatkovno rudarjenje nad velikimi količinami podatkov ter z njimi izvesti napoved prehoda stranke h konkurenčnemu operaterju v telekomunikacijskem podjetju.

Rešitev bomo sprogramirali in sestavili iz različnih odprtokodnih sistemov. Zaradi specifičnosti odprtokodnih sistemov in zaradi povezovanja različnih odprtokodnih rešitev bomo potrebovali tudi nekaj programerskega znanja, da rešitve povežemo ter izluščimo rezultate, ki sicer pritičejo specifično razvitim velikim lastniškim sistemom.

Da bi lahko analizirali primernost uporabe odprtokodne rešitve, bomo izdelali programsko rešitev, s katero bo mogoče opravljati statistične analize in napovedi. V konkretnem primeru bomo naredili prediktivni model za napovedovanje prehajanja uporabnikov h konkurenci, s katerim bomo lahko ocenili verjetnost, da bodo naročniki mobilne telefonije zaradi kakršnihkoli vzrokov prešli h konkurenčnemu operaterju.

Želeli smo preveriti, ali bi bilo mogoče izdelati takšno orodje za uporabo v profesionalne namene. Zato mora rešitev vsebovati način, kako priti do podatkov,

kako jih obdelati in kako prikazati rezultate. V nalogi bomo pokazali le enega izmed možnih načinov kako to narediti.

Takšna rešitev seveda ne more konkurirati komercialnim lastniškim rešitvam, ki poleg programske opreme ponujajo tudi izkušnje in svetovanje. Pokaže pa na to, da so odprtokodne rešitve možne in tudi dobre ter delujoče, kljub temu, da so cenejše ali celo brezplačne, kot v našem primeru. Odpira nam možnost, da bi jih bilo mogoče uporabiti tudi v profesionalne in ne zgolj v raziskovalne in izobraževalne namene. Prikazali bomo postopek, kako se napiše programska rešitev in sestavi potrebne komponente, da pridemo do delujočega sistema.

Ker je tudi odprtokodnih (Weber, 2004) rešitev precej, vse pa imajo svoje prednosti in slabosti, smo kot najoptimalnejšo kombinacijo izbrali Orange (Orange, 2014), Python (Python, 2014) in Web2py (Web2py, 2014), v nalogi pa bomo pojasnili, kako in zakaj smo prišli do te odločitve.

Orange je programski paket za strojno učenje in podatkovno rudarjenje, napisan v jeziku Python. Vključuje orodje za vizualno računalniško podprto načrtovanje analiz (Demšar et al., 2013). Ker pa to orodje zaradi svojih omejitev ne zmore obdelovati večjih količin podatkov, bomo uporabili pristop, pri katerem s programsko kodo napisano v jeziku Python uporabimo knjižnice paketa Orange in tako lahko obdelujemo bistveno večje količine podatkov.

Ker bomo zaradi tega izgubili uporabniški vmesnik, s katerim je delo relativno enostavno, smo se odločili, da napišemo spletno aplikacijo s katero bomo upravljali program in prikazali rezultate. Spletna aplikacija omogoča enostavnejšo uporabo od klasičnih aplikacij, saj omogoča dostop do aplikacije vsem uporabnikom kar preko spleta.

Ker je Orange večinoma napisan v Pythonu, je tudi za spletno aplikacijo edino smiselno uporabiti rešitev, ki bi temeljila na istem jeziku. Tako smo po pregledu mnogih rešitev izbrali spletno ogrodje Web2Py, ki je nastalo pod taktirko Massima di Pierra in prav tako temelji na jeziku Python (Massimo Di, 2011). Dodatna prednost takšne tehnologije je tudi prenosljivost med operacijskimi sistemi, saj je jezik Python neodvisen od platforme, na kateri teče.

Ta spletni vmesnik nam bo omogočil, da napišemo uporabniški vmesnik, ki omogoča nalaganje podatkov v model, izvajanje klasifikacije za napovedovanje in shranjevanje rezultatov. Poleg tega vsebuje tudi varnostne mehanizme za dostop. Tudi uporabniška izkušnja bo na zelo visokem nivoju, saj je pretežni način uporabe aplikacij postal svetovni splet.

Uporabili bomo podatke o strankah telekomunikacijskega podjetja in jih kategorizirali. Delali bomo na samo določenem segmentu uporabnikov in sicer na fizičnih osebah in to tistih, ki ob prehodu h konkurenčnemu operaterju, obdržijo svojo telefonsko številko.

Rezultati analize naročnikov bodo postali vir za kontaktiranje uporabnikov, tako da jih bomo lahko s pomočjo metod neposrednega trženja poskusili prepričati, da ne prekinjajo pogodbene vezave pri trenutnem operaterju.

Primernost rešitve bomo ovrednotili s količino dela in programerskega znanja, ki je potrebno, da programsko rešitev sestavimo. Predstavili bomo omejitve pri obdelovani količini podatkov ter omejitve strojne in programske opreme. Predstavili bomo spletno aplikacijo, kot prototip možne rešitve problema. Primernost bomo ugotavljali tudi z funkcionalnostjo sestavljene rešitve, ki bo produkt te naloge. S to nalogo bomo namreč pokazali način, kako to idejo uspešno realizirati.

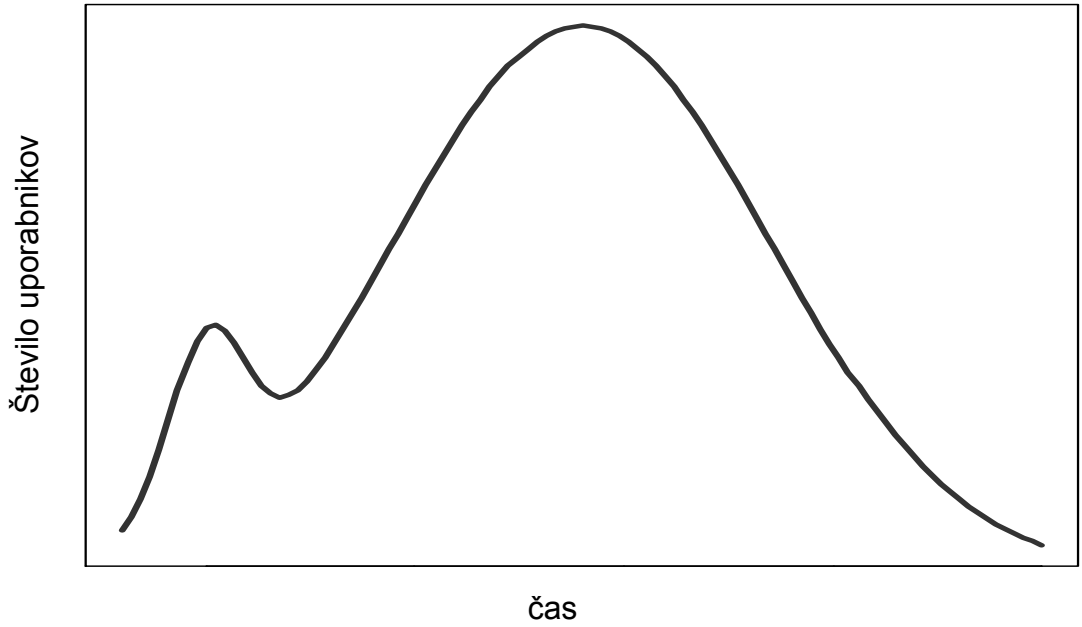
Statistične metode, ki so integrirane v programski opremi, so splošno znane ter morajo delovati enako ne glede na to v katerem programskem paketu so uporabljene. Rezultati napovednega modela, uporabljenega v aplikaciji, niso odvisni od uporabljene programske opreme ampak od vhodnih podatkov in izbire napovednega modela.

# 1 TRŽENJSKI IZZIV

Nove tehnologije imajo na trgu določen življenjski cikel. Ta prehaja skozi zgodnjo fazo rasti, preko brezna do širitve proti odrasli dobi, kjer ima največ uporabnikov, nato njena uporaba upade.

V zgodnji fazi se za produkt odloči nekaj tehnoloških navdušencev, ki preizkusijo novost na trgu. Sledi brezno. To je obdobje, preden je tehnologija na voljo za množično uporabo, tehnološki navdušenci pa so prešli na druge tehnologije. Sledi zlato obdobje rasti, kjer se prodaja eksponentno povečuje. V tem obdobju se produkt oziroma tehnologija praktično sama prodaja, saj se želja po izdelku širi sama med ljudmi. Lep primer takšne rasti so mobilni telefoni. Nato pride do odrasle dobe, kjer se trg zasiči. Vsi, ki želijo uporabljati tehnologijo jo že uporabljajo. Rast se ustavi. Uporabniki začnejo prehajati na nove tehnologije. Pride do upada, ki se konča tako, da ni več uporabnikov (Moore, 2001). Pojav je prikazan na sliki 1.

*Slika 1: Življenjski cikel novih tehnologij*




*Vir: R. Mattison, The telco churn management handbook, 2005, str. 22.*

Tudi telekomunikacijska podjetja se soočajo z življenjskim ciklom njihovih tehnologij. V odrasli dobi se morajo soočiti z novimi razmerami na trgu. Obstajajo tri osnovna področja, ki jih morajo telekomunikacijska podjetja v zreli dobi upoštevati, da lahko preživijo. Na področju delovanja organizacije je potrebna poenostavitev, torej ohranjanje najboljših rešitev in uvedbo teh na ravni celotne organizacije. Usklajevanje na ravni storitev in omrežij se nanaša na združevanje storitev mobilne in fiksne telefonije. Usmerjenost k uporabniku, pa pomeni izvajanje raznih pristopov, tako do stranke kot tudi na ravni organizacije za zadrževanje starih in pridobivanje novih strank. Takšno podjetje stalno razmišlja kako bi zadovoljilo sedanje in tudi prihodnje stranke (Wavelet, Trejo, Griffa, & Vallejo, 2012).

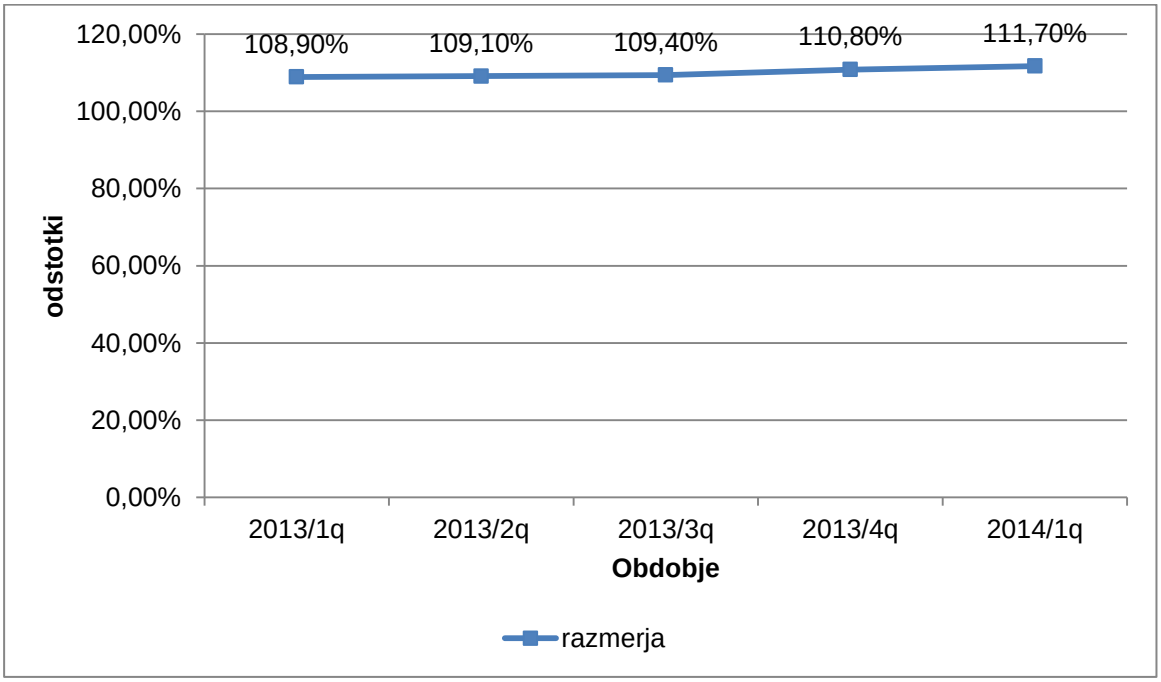
V obdobju rasti se telekomunikacijskim podjetjem ni bilo potrebno ukvarjati s prehodi strank h konkurenci, saj se jim je število novih uporabnikov stalno povečevalo. Poleg tega uporabniki niso veliko prehajali, saj se je večina šele prvič srečala z novo tehnologijo. Podjetja si razdelijo deleže na trgu, ki se z naraščanjem uporabnikov ne spreminja. Delež je odvisen od trženjskih sredstev, možnosti hitrega novačenja strank in postavljenega omrežja. Ker stranke ne vedo, kaj lahko pričakujejo od novega produkta, nimajo postavljenih standardov in se ne ozirajo h konkurenci. To obdobje je zlato obdobje za podjetje, saj uporabniki ostajajo in novi prihajajo ne glede na storitve, ki jim jih podjetje nudi. Šele ko

nastopi odrasla doba, se morajo podjetja soočiti z novimi načini pridobivanja strank oziroma se boriti za tiste, ki jih že imajo. Uporabniki si postavijo standarde in vedo kaj pričakujejo od storitve. Začnejo preverjati konkurenčne ponudbe. Delež podjetja na trgu se lahko drastično spremeni, saj se uporabniki odločajo za podjetja, ki zagotavljajo boljšo uporabniško izkušnjo in ki jim nudijo več storitev. Vsa podjetja se začnejo soočati s prehodi strank h konkurenci (Mattison, 2005).

## 1.1 Zasičenost trga

Po podatkih Statističnega urada Republike Slovenije (v nadaljevanju SURS) je bilo ob koncu leta 2013 Sloveniji skoraj 2.284.000 aktivnih uporabnikov mobilnega omrežja (SURS, 2014). Po podatkih Agencije za komunikacijska omrežja in storitve Republike Slovenije (v nadaljevanju AKOS) se penetracija trga v Sloveniji še rahlo povečuje in je bila v začetku leta 2014 že 111,7 odstotna (AKOS, 2014). Kljub temu lahko ugotovimo, da je trg mobilne telefonije v zreli fazi in nadaljnja večja rast števila uporabnikov ni več mogoča. Na sliki 2 vidimo grafičen prikaz rasti števila aktivnih uporabnikov v obdobju zadnjih dveh let.

*Slika 2: Penetracija aktivnih uporabnikov mobilne telefonije na prebivalstvo*




*Vir: AKOS, Poročilo o razvoju trga elektronskih komunikacij za prvo četrtletje 2014, 2014, str. 16.*

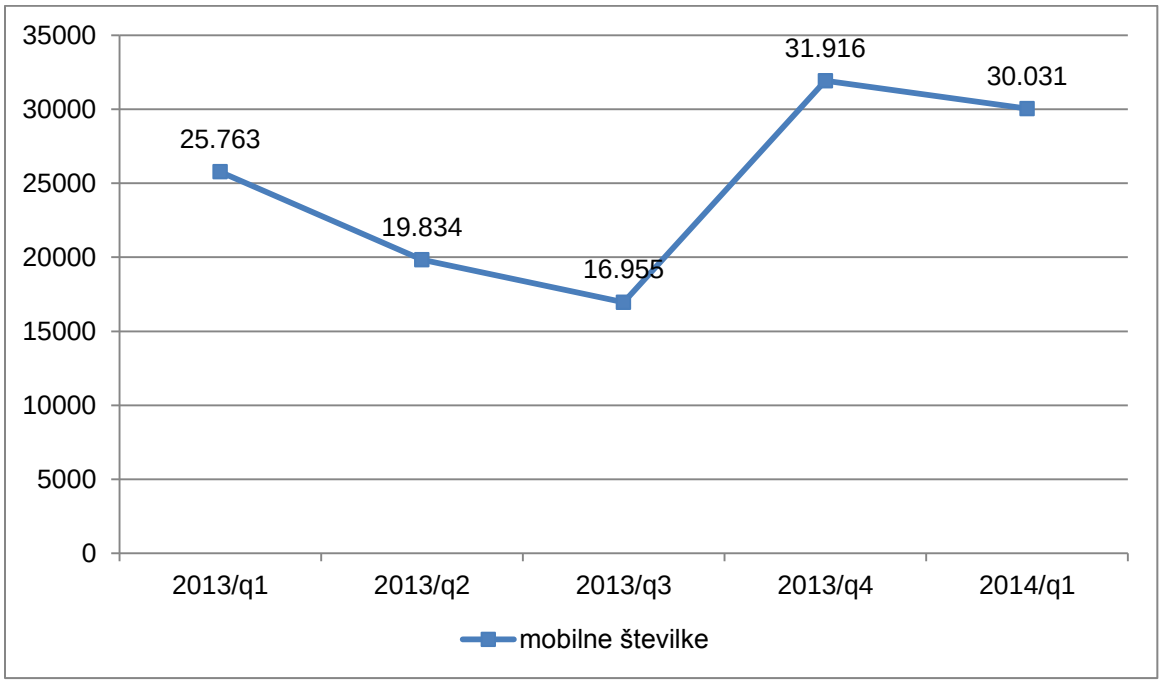
V Sloveniji na trgu posluje sedem operaterjev mobilne telefonije, od katerih sta dva virtualna, kar zagotavlja močno konkurenco med operaterji (AKOS, 2014).

## 1.2 Prehajanje strank h konkurenci

Kotler (1994) ocenjuje, da šestnajstkrat cenejše prepričati stranko, da ne preide h konkurenci, kot je poiskat novo stranko in cena za pridobitev nove stranke je petkrat do šestkrat večja od cena za zadržanje obstoječe. Reicheld in Sasser (1990) predvidevata, da lahko podjetje poveča dobiček do 100%, če uspe zmanjšati prehode strank h konkurenci za 5%. Ta podatek pokaže, kako velik vpliv ima prehajanje strank h konkurenci na poslovanje podjetja.

Po navedbah upravljalca centralne baze prenesenih številk je bilo 8 let po uvedbi možnosti prenosa mobilnih telefonskih številk v Sloveniji uspešno prenesenih že več kot 1 milijon mobilnih telefonskih številk. Na sliki 3 je prikazano število prenosov mobilnih telefonskih številk k operaterjem, pri čemer so upoštevani vsi prenosi telefonskih številk, torej tudi prenos k novemu operaterju in nato nazaj (AKOS, 2014).

*Slika 3: Število prenosov mobilnih telefonskih številk k operaterjem*

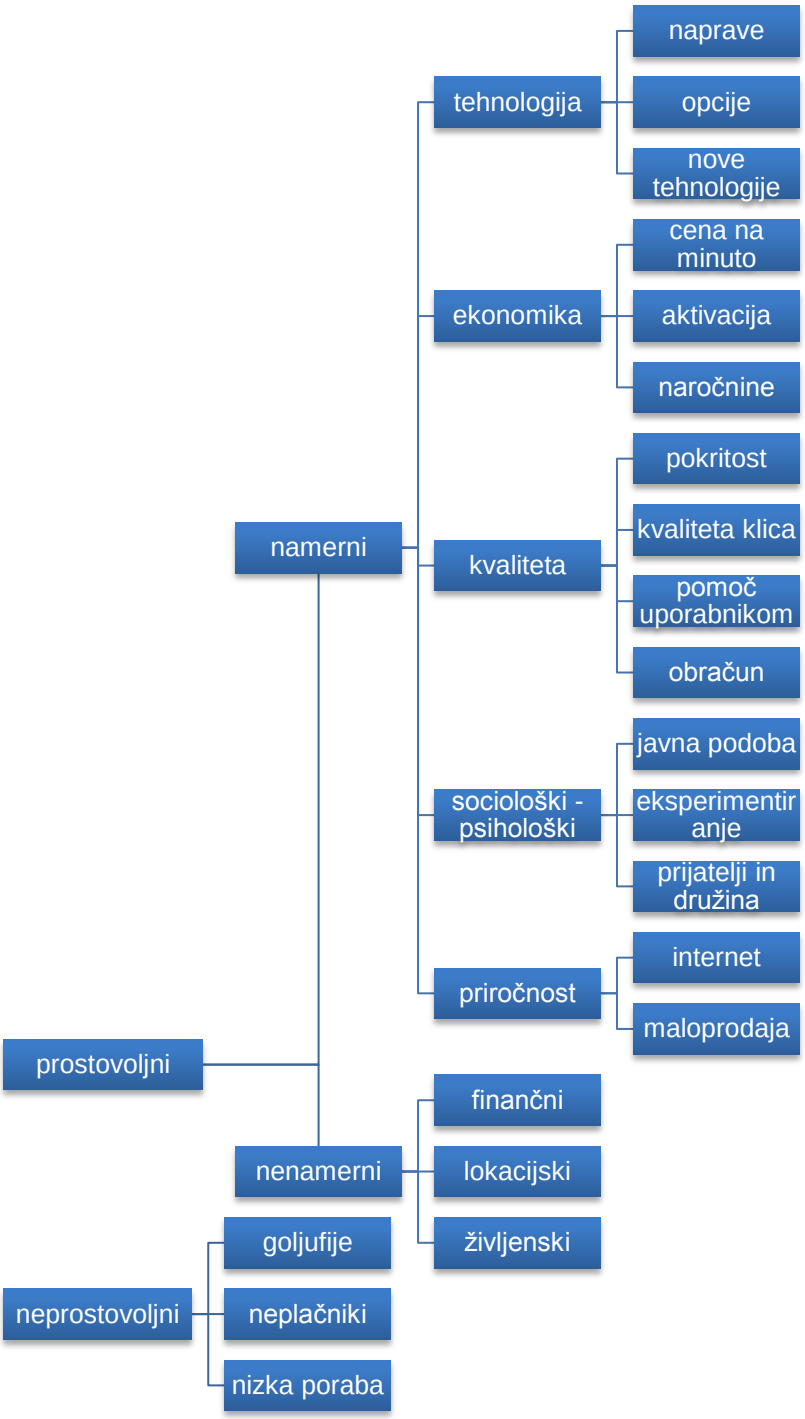



*Vir: AKOS, Poročilo o razvoju trga elektronskih komunikacij za prvo četrtletje 2014, povzeto po poročilu upravljavca CBP na dan 1.4.2014, 2014, str.24.*

V grobem ločimo prehajanje strank h konkurenci na prostovoljno in neprostovoljno. Med neprostovoljno prehajanje štejemo tiste stranke, ki jim podjetje sámo prekine pogodbo iz različnih razlogov. Najpogostejši razlogi pa so goljufije, neplačevanje, slaba kreditna sposobnost, neuporaba storitev in nizki prihodki od posameznih naročnikov (Mattison, 2005).

Zaradi preprečevanja izgube prihodkov pa podjetje v vsakem primeru bolj zanima prostovoljno prehajanje strank h konkurenci. Na sliki 4 je prikazana razdelitev razlogov za prehajanje h konkurenci.

*Slika 4: Razlogi za prehajanje h konkurenci*




*Vir: R. Mattison, The telco churn handbook, 2005, str. 35.*

Rezultati raznih tržnih raziskav so pokazali, navaja Mattison (2005), da se kot najpogostejši razlogi za prehajanje strank h konkurenci izkažejo sledeči:

- cena,
- kakovost storitev,
- pokritost s signalom,
- služba za pomoč uporabnikom,
- javna podoba.

Poleg prostovoljnega in neprostovoljnega prehajanja obstaja tudi slučajno prehajanje strank, ki je neodvisno od ponudnika in od stranke, do tega pa včasih vseeno pride npr. zaradi padca dohodkov, selitev in ostalih življenjskih sprememb (Mattison, 2005).

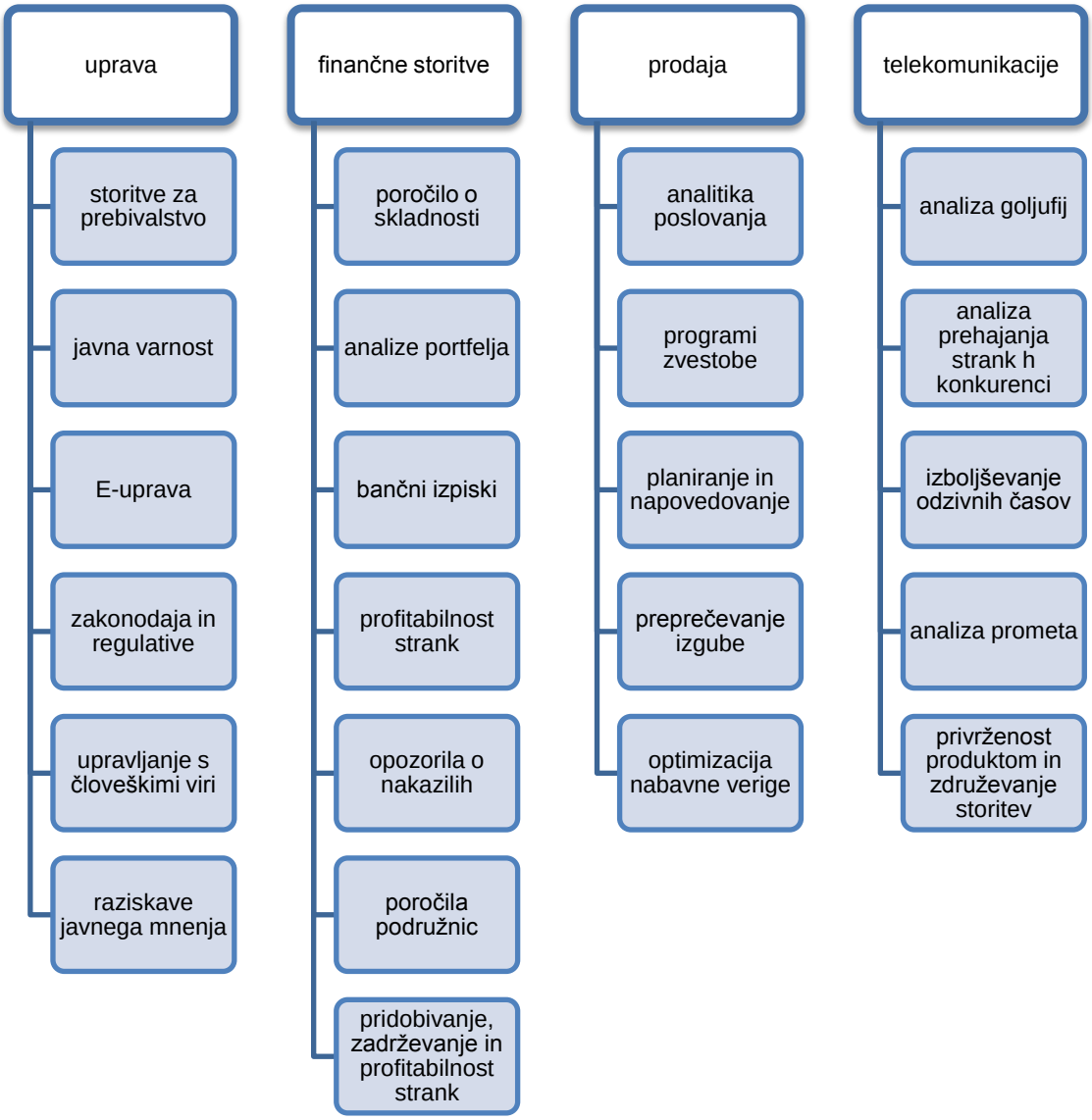
Zdaj ko je telekomunikacijski trg v zreli fazi, je postalo še pomembnejše upravljanje odnosov s strankami. Ker je pridobivanje novih strank nekajkrat dražje od zadrževanja trenutnih, dobiva vedno pomembnejšo nalogo področje preprečevanja prehodov strank h konkurenci (Gürsoy, 2010).

# 2 POSLOVNA INTELIGENCA

Poslovna inteligenca je veda, ki se ukvarja s preiskovanjem podatkov in iskanjem informacij, ki temeljijo na dokazih in so lahko podlaga za nadaljnje ukrepe. Poslovna inteligenca omogoča podjetjem odkrivanje in uporabo informacij, ki jih že poseduje, da jih preoblikuje v znanje, ki neposredno vpliva na delovanje podjetja (Pareek, 2007).

Področje uporabe poslovne inteligence je zelo široko in čeprav se bomo v tej nalogi ukvarjali z uporabo poslovne inteligence v telekomunikacijskem sektorju, velja omeniti še druga področja na katerih se uporablja. Prikazani so na sliki 5. Čeprav se okoliščine pojavov, ki jih raziskujemo med temi področji precej razlikujejo, so metode za preučevanje teh pojavov zelo podobne.

*Slika 5: Področja uporabe poslovne inteligence*




*Vir: D. Pareek, Bussines Intelligence for Telecommunications, 2007, str. 32.*

Poslovna inteligenca je krovni izraz, ki zaobjame aplikacije, infrastrukturo in orodja skupaj z najboljšimi praksami, ki omogočajo dostop in analizo informacij za izboljšanje in optimizacijo odločitev in delovanja (Business Intelligence (BI), 2013).

Poslovna inteligenca predstavlja okolje v katerem poslovni uporabniki pridobivajo podatke, ki so zanesljivi, konsistentni, razumljivi in pravočasni in se da z njimi preprosto upravljati. Poslovna inteligenca služi dvema osnovnima ciljema. Meri finančno in operativno stanje (poročila, opozorila, alarmi, analitična orodja, ključni kazalniki uspešnosti, grafični prikazi) in regulira organizacijske odločitve z dvosmerno integracijo operacijskih sistemov in analizo informacij o odgovorih (Business Intelligence, 2014).

Orodje poslovne inteligence je programska oprema, ki omogoča uporabnikom, da pregledajo in uporabijo velike količine kompleksnih podatkov. Pod te spadajo (Bussines Intelligence Tools, 2014):

- OLAP kocke (angl. *OnLine Analytical Pocessing*), ki omogočajo večdimenzionalne poglede na podatke;
- orodja za poizvedbe – programska oprema, ki omogoča uporabniku, da poizveduje po vzorcih ali podrobnostih znotraj podatkov;
- orodja za podatkovno rudarjenje – programska oprema, ki samostojno preiskuje podatke in išče izstopajoče vzorce in povezave znotraj podatkov.

## 2.1 CRISP model podatkovnega rudarjenja

CRISP-DM model (angl. *CRoss-Industry Standard Process for Data Mining*) je model podatkovnega rudarjenja, ki je prosto dostopen in se lahko uporablja v katerikoli panogi in pri katerikoli aplikaciji. Organizacijam ponuja strukturo za hitrejše in boljše izvajanje podatkovnega rudarjenja, saj je nastal na podlagi najboljših praks. Model razdeli proces podatkovnega rudarjenja na šest faz (Shearer, 2000):

- razumevanje poslovnega okolja,
- poznavanje podatkov,
- priprava podatkov,
- modeliranje,
- ocena uspešnosti,
- postavitev v produkcijsko okolje.

Kot pravi Shearer (2000) je razumevanje poslovnega okolja verjetno najpomembnejša faza pri podatkovnem rudarjenju. Osredotoča se na razumevanje ciljev projekta s poslovnega vidika. V tem koraku pretvorimo razumevanje problema v definicijo problema podatkovnega rudarjenja. Da bi lahko razumeli kateri podatki bodo analizirani je razumevanje poslovnega okolja ključnega pomena. To fazo lahko razdelimo na več korakov, ki obsegajo določitev poslovnih ciljev, oceno stanja, določitev ciljev podatkovnega rudarjenja in izdelavo projektnega načrta.

Drugo fazo začnemo s preliminarno izbiro podatkov. Potrebno je spoznati podatke, identificirati potencialne kvalitativne probleme, prepoznati zanimive vzorce, da bi lahko oblikovali hipotezo o vsebovanih informacijah. To fazo lahko razdelimo na več korakov, ki obsegajo izbor preliminarnih podatkov, opis podatkov, raziskovanje podatkov in preverjanje kvalitete podatkov (Olson & Delen, 2008).

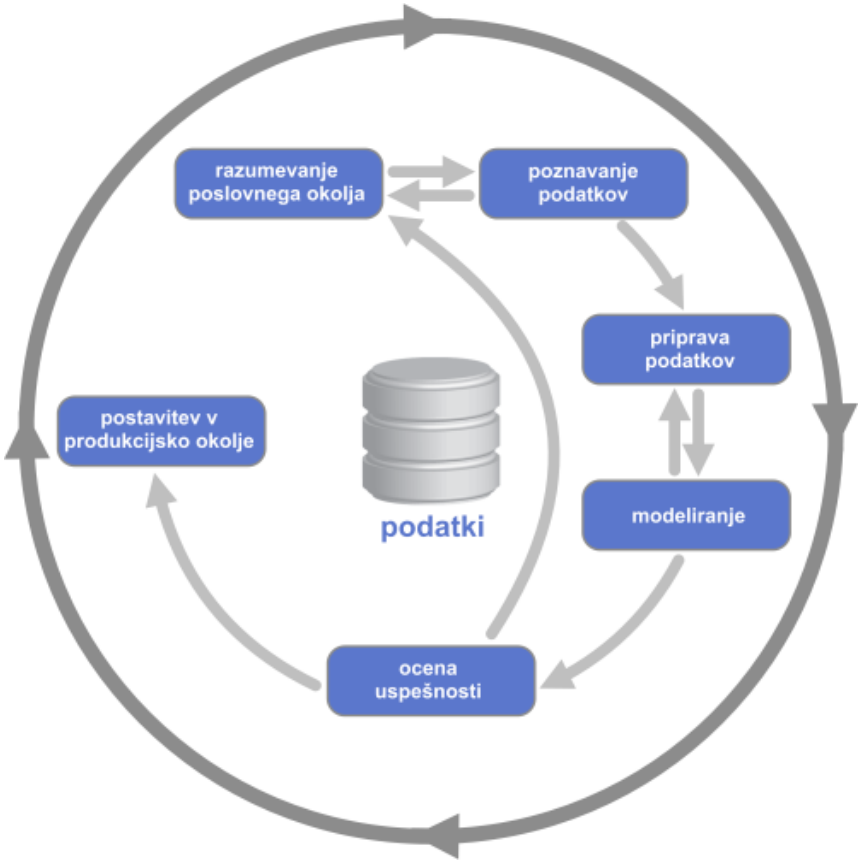
Priprava podatkov pokriva vse aktivnosti potrebne za izgradnjo podatkovne zbirke, ki bodo služili, kot vhod v model. Naloge vključujejo tako izbor tabel, zapisov in atributov kakor tudi transformacijo in čiščenje podatkov. To fazo delimo na pet korakov. To so izbor podatkov, čiščenje podatkov, izgradnja podatkov, integracija podatkov in oblikovanje podatkov (Shearer, 2000).

V četrti fazi izbiramo model in parametrom modela določamo optimalne vrednosti. Reševanja problema v podatkovnem rudarjenju se lahko lotimo z različnimi tehnikami od katerih imajo nekatere specifične zahteve za obliko podatkov. Zato se v tej fazi včasih vrnemo v fazo priprave podatkov. Faza je razdeljena na štiri korake. To so: izbor modela, izdelava testnega okolja, izdelava modela in ocenjevanje modela (Olson & Delen, 2008).

Preden model umestimo v produkcijsko okolje ga je potrebno oceniti, da bi bili prepričani, da dosega zastavljene cilje. V tem koraku je potrebno preveriti ali so bile izpuščene kakšne pomembne poslovne zahteve. Na koncu te faze se projektni vodja odloči kako bo uporabil rezultate podatkovnega rudarjenja. Glavni koraki v tej fazi so ocenjevanje rezultatov, revidiranje procesa in določitev naslednjih korakov. Ta faza velikokrat vodi nazaj k prvi fazi, saj so pridobljeni podatki pokazali nova razmerja v poslovnem okolju (Olson & Delen, 2008).

Postavitev v produkcijsko okolje še ne pomeni, da je projekt zaključen. Pridobljeni rezultati morajo biti predstavljeni na takšen način, da jih uporabnik razume in da jih lahko uporabi. Postavitev v produkcijsko okolje lahko pomeni samo izdelavo poročila, lahko pa je kompleksnejša, kot na primer implementacija postopka, ki se v podjetju večkrat uporablja, po navadi v enakomernih časovnih presledkih (Shearer, 2000). Celoten proces podatkovnega rudarjenja po metodi CRISP je prikazan na sliki 6.

*Slika 6: CRISP-DM model*




*Vir: C. Shearer, The CRISP-DM Model: The New Blueprint for Data Mining, 2000.*

Šest faz predstavlja osnovne korake, ki jim lahko sledimo. Izkušen analitik morda ne bo uporabil vseh korakov v vsaki študiji. Model CRISP-DM podaja dobro osnovo za projekt podatkovnega rudarjenja. Poleg CRISP-DM modela obstaja tudi SEMMA model, ki pa je specifičen za SAS (Olson & Delen, 2008).

## 2.2 Management odnosov z odjemalci

Management odnosov z odjemalci (angl. *Customer Relationship Management*) (v nadaljevanju CRM) je strateški pristop, ki naj bi zagotavljal večji dobiček, skozi vzpostavljanje primernega odnosa s ključnimi strankami in uporabniškimi segmenti. Združuje tako trženjske strategije kakor informacijsko komunikacijske strategije, da ustvari dobičkonosno, dolgoročno razmerje s strankami in ostalimi uporabniki. Management odnosov z odjemalci daje veliko priložnost, da se skozi pridobljene podatke in informacije bolje razume uporabnike in njihovo vrednost. Za ta proces je potrebno navzkrižno povezovanje procesov, ljudi, operacij in tržnih zmožnosti (Payne & Frow, 2005).

Management odnosov z odjemalci je gradnja dolgoročnega močnega in lojalnega odnosa s strankami. Pristop je osredotočen na uporabnika in zgrajen na vpogledu v obnašanje uporabnika. Z upoštevanjem različnosti uporabnikov, različnega obnašanja ter posebnih potreb in zahtev pridemo do osebno usmerjenega upravljanja z uporabniki (Tsiptsis & Chorianopoulos, 2009).

## 2.3 Programske rešitve za management odnosov z odjemalci

Obstaja veliko programske opreme za učinkovit management odnosov z odjemalci. Največkrat se uporablja pri osebnih kontaktih s stranko, preko prodaje, trženja ali centra za pomoč uporabnikom. Ti beležijo strankino zgodovino, pomembne informacije o stranki in nudijo pomoč pri zagotavljanju dobrega odnosa s strankami (Tsiptsis & Chorianopoulos, 2009).

Management podatkov o odjemalcih predstavlja zbiranje, združevanje in uporabo podatkov o odjemalcih z vseh kontaktnih točk, da bi dobili pravo sliko in pravilne marketinške odzive za določeno stranko. Tako pridobljeni podatki prikažejo podjetju zgodovinski odtis stranke in njenih interakcij z podjetjem. Podatki se stekajo v podjetje preko raznih kanalov, preko pravega stika s stranko ali preko beleženja strankinega delovanja (Greenberg, 2010).

## 2.4 Ocena uporabnikov

Da bi podjetja zares dobila vpogled v potrebe in želje strank, morajo narediti analizo podatkov. Analitični pristop k managementu odnosa z odjemalci omogoča boljši vpogled v uporabnika. Uporabijo lahko modele podatkovnega rudarjenja, da pridobijo oceno stranke in razumejo ter predvidijo strankino vedenje. S podatkovnim rudarjenjem se iščejo vzorci v podatkih, iz katerih se lahko pridobi znanje o uporabnikih, na podlagi katerih se nato optimizira odnose s strankami (Tsiptsis & Chorianopoulos, 2009).

Izdelajo se ocene o uporabniku (Tsiptsis & Chorianopoulos, 2009):

- ocena trenutne in bodoče profitabilnosti,
- tip stranke, demografski podatki, podatki o obnašanju in potrebah pridobljeni z analizo podatkov in segmentacijo,
- potencial rasti preko navzkrižne prodaje ali nadgradnje,
- plačila in ocena zmožnosti plačevanja,
- ocena verjetnosti prehoda stranke h konkurenci.

Telekomunikacijska podjetja potrebujejo programe za izvajanje poslovne inteligence, da bi lahko izvajale podatkovne analize masovnih podatkov v realnem

času. Z analizo podatkov o klicih in ostalih podatkov o uporabniku lahko določijo oceno uporabnika na podlagi katere nato določijo strategijo kako obdržati uporabnika in s tem tudi svoj dobiček (Ishaya & Folarin, 2012).

# 3 MASOVNI PODATKI

Prva pisna omemba izraza masovni podatki (angl. *big data*) naj bi se po poročanju Forbesa pojavila leta 1997 v dokumentu pri Ameriški NASI, ob opisu problema z vizualizacijo. Bolj popularen je postal po letu 2008, z napovedjo da bodo masovni podatki preoblikovali računalništvo (Press, 2014).

Količina podatkov se povečuje v treh smereh. Povečuje se količina, hitrost ter raznolikost podatkov (Laney, 2001). Količina podatkov je že tako velika, da marsikatero podjetje ne uporablja vseh podatkov, ki jih ima na voljo. Takšnim podatkom se reče temni podatki (angl. *dark data*), saj predstavljajo zamujeno priložnost, da bi pripomogli k boljšemu poznavanju uporabnikov (Burgelman, 2013).

Uspešen projekt podatkovnega rudarjenja je močno odvisen od širine in kvalitete pridobljenih podatkov. Zato je faza pridobivanja in priprave podatkov zelo pomembna in tudi zahteva veliko časa (Tsiptsis & Chorianopoulos, 2009).

## 3.1 Podatki in čas hranjenja

Tako kot v vseh transakcijskih sistemih tudi v telekomunikacijskem sektorju nenehno nastaja ogromna količina podatkov. Nekateri podatki so pretežno statični, to so recimo demografski podatki z izjemo starosti, ki se izračunava sproti, medtem, ko se drugi podatki spreminjajo iz sekunde v sekundo, to so podatki o opravljenih klicih in preneseni količini podatkov.

Podatki o porabi storitev se shranjujejo v obliki CDR zapisov (angl. *Call Detail Record*) (v nadaljevanju CDR) v tabelah podatkovnih baz. Ta zapis vsebuje podrobne informacije o klicu ali prenosu podatkov na omrežju povezanih s samo stranko. V prvi vrsti so te informacije namenjene obračunavanju storitev (Whetten, Smith, Pearce, & Alexander, 2005). Koliko časa se lahko hranijo pa je v Evropi odvisno od direktiv Evropske unije in zakonodaje posameznih držav (Frost & Sullivan, 2010).

## 3.2 Viri podatkov

Podatki, ki se običajno uporabljajo v napovednih in analitičnih modelih se lahko v grobem delijo na prometne podatke in na podatke o uporabniku. Ti podatki so (Kraljević & Gotavec, 2010):

- prometni podatki: telefonska številka, čas pogovora, tip, omrežje klicanega in drugi; ti podatki so shranjeni kot zapis CDR in prikazujejo natančne podatke posameznega klica za vsakega uporabnika;
- podatki o uporabniku:
    - registracija in demografski podatki (spol, starost);
    - osnovna segmentacija (naročniki, predplačniki);
- prehodi med segmenti.

Poleg zgoraj naštetih pa tudi (Tsiptsis & Chorianopoulos, 2009):

- odhodni podatki za določanje količine uporabe (vrsta, številka, število in omrežje (lastno, tuje, tujina)) skozi čas;
- dohodni podatki za določanje uporabe (vrsta, število in omrežje (lastno, tuje, tujina)) skozi čas;
- količina prenesenih podatkov;
- finančni podatki, kot so prihodki in cena (plačani prihodki, stroški medomrežnega povezovanja in drugi);
- nakupi mobilnih naprav in dodatne opreme;
- dodatne storitve (glasba, novice in drugo);
- akcijske ponudbe in odziv;
- zabeležene pritožbe in število klicev na center za podporo uporabnikom;
- obračuni, plačila in podatki o kreditih (povprečno število dni do plačila, število izklopov, število dni do ponovnega vklopa in podobni).

Ker so podatki o klicih, ki nastajajo kot zapisi CDR, za uporabo v analitične preveč podrobni, jih grupiramo že na nivoju podatkovnega skladišča na nivo mesečne porabe razdeljene na posamezne storitve (Tsiptsis & Chorianopoulos, 2009).

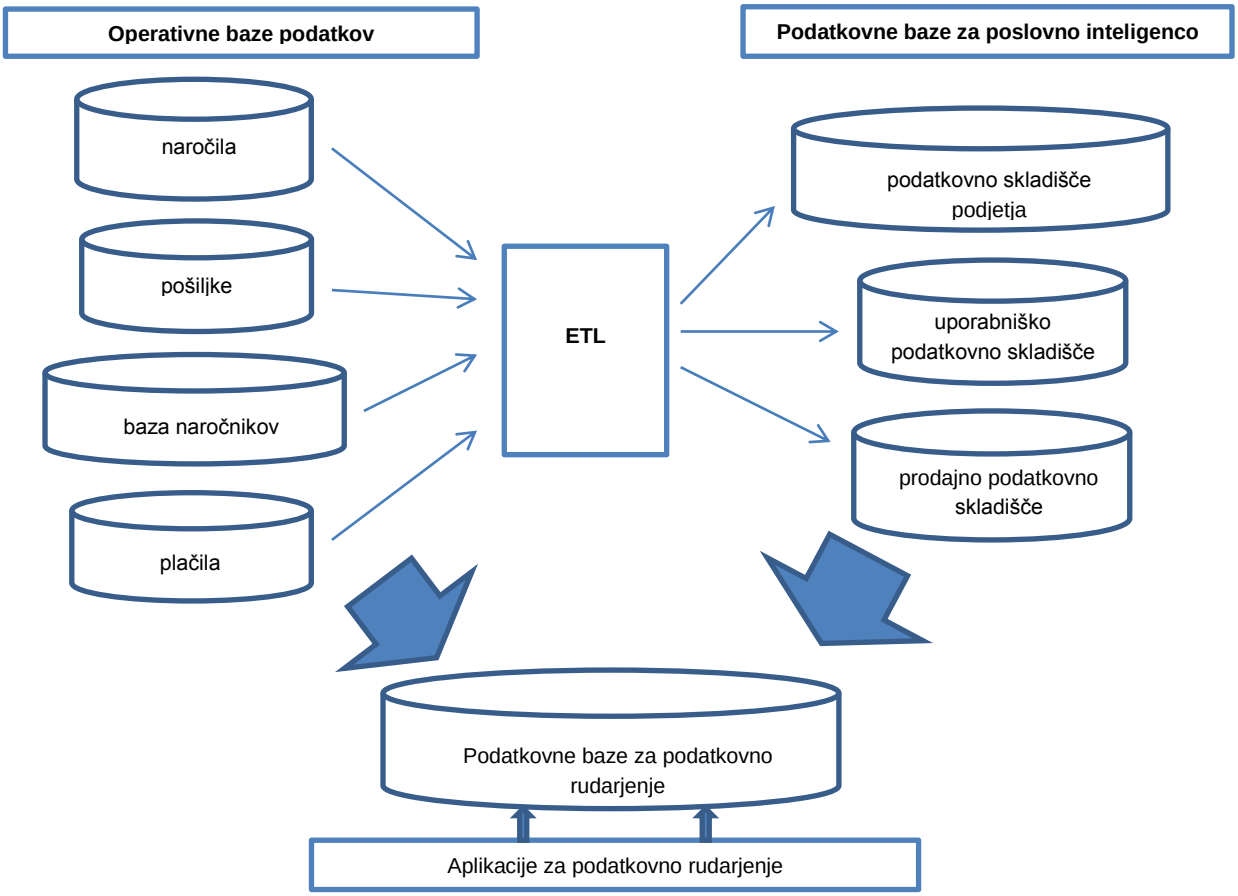
Tako se izgubijo podatki o posameznih klicih, ohrani pa se dovolj informacij, da se iz njih da relativno dobro predvideti nekatere vzorce obnašanja stranke, od katerih pa nas najbolj zanima verjetnost prehoda h konkurenci.

## 3.3 Združevanje podatkov

Da lahko podatke uporabimo v poslovni analitiki jih moramo preoblikovati, in združiti v razumljivo celoto, saj se običajno nahajajo v različnih zbirkah, podatkovnih bazah in med seboj nekompatibilnih sistemih, kar je prikazano na sliki 7. Postopek tega združevanja se imenuje ETL (angl. *Extract Transform Load*) (v

nadaljevanje ETL) in pomeni zbiranje, preoblikovanje in nalaganje podatkov (Oracle, 2014).

*Slika 7: Proces združevanja več virov podatkov v eno podatkovno skladišče, ki se uporablja za podatkovno rudarjenje (ETL)*




*Vir: D. Pareek, Bussines Intelligence for telecommunications, 2007, str. 85.*

Prečiščene informacije prirejene za analitično uporabo iz vseh relevantnih virov podatkov se nahajajo v podatkovnem skladišču za podatkovno rudarjenje (angl. *mining data mart*). Da bi bili uporabni za podatkovno rudarjenje, se morajo sproti posodabljati ter zajemati podatke za najmanj zadnji 2 leti (Tsiptsis & Chorianopoulos, 2009).

# 4 PROGRAMSKA OPREMA ZA PODATKOVNO RUDARJENJE

Na trgu je veliko programske opreme za podatkovno rudarjenje. Izbira prave programske opreme je za podjetje zato zelo pomembna in ravno tako tudi težka in časovno potratna. Za izbiro določene programske opreme je potrebno preučiti (Bhargava, Aziz, & Arya, 2013):

- delovanje: zanesljivost, učinkovitost;
- funkcionalnost: vključene funkcije, interoperabilnost, varnostni mehanizmi, tipi podatkov;
- dodatne opcije: čiščenje podatkov, iskanje praznih vrednosti, filtri;
- kvaliteta: spreminjanje prejšnjih projektov, prenosljivost in uporabniški vmesnik;
- izvajalec: priročniki za uporabo, podpora, nadgradnje;
- strojna in programska oprema: zahtevana strojna in programska oprema za delovanje ter odprtost programske kode.

Izsledki raziskave (Piatetsky, 2014) o uporabi programske opreme za podatkovno rudarjenje, kjer je sodelovalo preko 3000 uporabnikov, so pokazali, da je 71% uporabnikov, ki so glasovali, uporabljalo lastniško programsko opremo in 78% odprtokodno. 22% jih je uporabljalo le lastniško, kar je manj kot leto prej, ko je bilo takšnih uporabnikov 29%. Okrog 28,5% pa jih je uporabljalo le odprtokodno programsko opremo, kar je tudi manj kot leto prej, ko je bilo takšnih uporabnikov okrog 30%. Zgovoren podatek glede manjšanja razlike v uporabi med odprtokodno in lastniško programsko opremo za podatkovno rudarjenje je, da je kar 49% uporabnikov uporabljalo obe vrsti programske opreme. Leta 2013 le 41%. V tabeli 1 je prikazanih nekaj najbolj uporabljanih lastniških in odprtokodnih programov za podatkovno rudarjenje. Podatki so bili pridobljeni z odgovori na vprašanje: Katero programsko opremo ste v zadnjem letu uporabili v realnem projektu? Prikazani so tudi rezultati uporabe glede na prejšnje leto.

Glede na raziskave iz prejšnjih let lahko ugotovimo, da se uporaba programov za podatkovno rudarjenje počasi veča, a je še vedno omejena na ozke skupine analitikov, ki večinoma delujejo na področju svetovnega spleta, v vladnih ustanovah in velikih podjetjih, večina analiz pa je narejenih na srednje ali majhnih količinah podatkov (Piatetsky, 2014).

*Tabela 1: Lastniški in odprtokodni programi za podatkovno rudarjenje*

| Odprtokodni ali lastniški | Program | Samo to orodje v % | 2013 v % | 2014 v % |
|---|---|---|---|---|
| odprtokodni | RapidMiner | 35,0 | 39,2 | 44,2 |
| odprtokodni | R | 2,1 | 37,4 | 38,5 |
| lastniški | Excel | 0,1 | 28,0 | 25,8 |
| lastniški | SQL | 0,1 | Ni podatka | 25,3 |
| odprtokodni | Python | 0,9 | 13,3 | 19,5 |
| odprtokodni | Weka | 0,4 | 14,3 | 17,0 |
| odprtokodni | KNIME | 10,6 | 5,9 | 15,0 |
| odprtokodni | Hadoop | 0,0 | 9,3 | 12,7 |
| lastniški | SAS Base | 0,0 | 10,7 | 10,9 |
| lastniški | MS SQL Server | 0,0 | 7,0 | 10,5 |
| lastniški | Revolution Analytics R | 13,3 | 4,5 | 9,1 |
| lastniški | Tableau | 1,3 | 6,3 | 9,1 |
| lastniški | MATLAB | 0,0 | 9,9 | 8,4 |
| lastniški | IBM SPSS Statistics | 0,4 | 8,7 | 7,7 |
| lastniški | SAS (tudi BussinesObjects/Sybase/Hana) | 0,0 | 1,4 | 6,8 |
| lastniški | IBM SPSS Modeler | 3,2 | 6,1 | 5,7 |
| odprtokodni | druga brezplačna analitična orodja | 1,8 | 3,4 | 5,1 |
| odprtokodni | Rattle | 0,0 | 4,5 | 4,9 |
| lastniški | BayesiaLab | 23,5 | 1,0 | 4,1 |
| odprtokodni | GNU Octave | 0,0 | 2,9 | 3,9 |
| lastniški | JMP | 3,2 | 4,1 | 3,8 |
| lastniški | Prediction Software | 47,5 | 1,9 | 3,8 |
| lastniški | Salford SPM/CART/Random Forests/MARS/TreeNet | 31.4 | 2,2 | 3,6 |
| odprtokodni | Orange | 0,0 | 3,6 | 3,4 |

*Opomba: izpustili smo ostale manj uporabljene produkte*

*Vir: G. Piatetsky, KDnuggets 15th Annual Analytics, Data Mining, Data Science Software Poll, 2014.*

## 4.1 Primerjava med lastniškimi programskimi rešitvami za podatkovno rudarjenje

Primerjava programskih rešitev za podatkovno rudarjenje, ki jo izdela Gartner vsako leto, se osredotoča na možnosti uporabe programske opreme in zastavljeno vizijo. Voditelji so proizvajalci, ki so že utečeni in so globoko prisotni na trgu in po vsej verjetnosti vplivajo tudi na samo uporabo orodij in s tem na rast trga (Jeremy, 2014).

Na sliki 8 je prikazan magični kvadrant primerjave lastniških sistemov za podatkovno rudarjenje.

*Slika 8: Gartnerjev kvadrant za primerjavo lastniške programske opreme za podatkovno rudarjenje*


*Vir: G. Herschel, A. Linden & L. Kart, Magic Quadrant for Advanced Anaytics Platforms, 2014.*

Cena lastniških programov je večinoma odvisna od cene razvoja programske opreme in števila inovacij in prilagoditev, ki jih stranka zahteva. Kljub vedno več zahtevam za spremembe in prilagoditve ter zmogljivejšim programom in dodanim novim funkcionalnostim, cena lastniških programov za podatkovno rudarjenje stalno pada, predvsem zaradi razvoja odprtokodne programske opreme. Kljub temu, da je odprtokodna programska oprema načeloma brezplačna, je potrebno upoštevati ostale stroške, ki jih prinese. Takšni programi so visoko prilagodljivi a so odlični le za podjetja, ki imajo znanje iz programiranja. Lastniška programska oprema je opremljena s preprostim uporabniškim vmesnikom in navodili za uporabo za osnovne ter napredne funkcije. Kljub temu se vse več podjetij odloča za odprtokodne programske rešitve (Schmidt, 2014).

## 4.2 Odprtokodna programska oprema za podatkovno rudarjenje

Po navedbah Centra odprte kode Slovenije (v nadaljevanju COKS) so odprtokodni programi (angl. *open source software*) programi, za katere ne veljajo tako stroge licenčne omejitve glede načina uporabe, kopiranja, spreminjanja kode in distribucije, kot veljajo za večino lastniške programske opreme. Programska koda odprtokodnega programja je prosto dostopna vsakomur, da jo lahko ureja, spreminja, popravlja, izboljšuje in dograjuje. Posebnost odprtokodne licence pa je v tem, da se tako spremenjene kode ne sme izdati pod strožjimi licenčnimi pogoji kot so tisti, pod katerimi je izdana začetna koda. Ponujajo odlično alternativo (plačljivi) lastniški programski opremi (COKS, 2011).

### 4.2.1 Kriteriji odprtokodne programske opreme

Uradna definicija odprte kode pravi, da je to tisto programje, ki je izdano pod licenco, ki ustreza vsem desetim kriterijem Open Source Initiative. Ti kriteriji so (COKS, 2011):

- odprtokodno programsko opremo je mogoče svobodno redistribuirati; lahko jo redistribuira kdorkoli brezplačno ali proti plačilu;
- izvorna programska koda je dostopna uporabniku; licenca mora dovoljevati distribucijo v prevedeni kakor tudi v izvorni obliki;
- licenca mora dovoljevati spremembe osnovne kode in izvedene oblike nove kode;
- kljub temu, da mora biti izvorna koda dostopna, lahko izvorni avtorji zahtevajo, da se morebitne spremembe jasno ločijo od originalne kode in tako ohranijo ločnico med prvotno in modificirano kodo (npr. v obliki popravkov ali različnih verzij);
- licenca ne sme omejevati katerekoli osebe ali skupine;

- licenca ne sme biti omejevalna glede na področje dela, v okviru katerega se programska koda uporablja;
- distribucija licenc mora biti enakovredna za vse uporabnike, brez dodatnih omejitev;
- licenca za isto programsko kodo se ne sme razlikovati, če se jo uporablja v kombinaciji z drugo programsko opremo;
- licenca ne sme omejevati uporabe druge programske opreme;
- licenca mora biti tehnološko nevtralna.

Lahko jo opišemo tudi s naslednjimi besedami: odprtokodna programska oprema je naziv za programsko opremo, katere izvorna koda je prosto dostopna in jo je mogoče prosto uporabljati, raziskovati njeno delovanje, spreminjati in razširjati tako originalne kot dopolnjene in spremenjene kopije tega programja. (COKS, 2011).

### 4.2.2 Prednosti odprtokodne programske opreme

Odprtokodna programska oprema je postala komplementarna metoda tradicionalnemu razvoju lastniške programske kode. Prispevanje skupnosti in brezplačna uporaba sta postali nekaj običajnega. Pomembno vlogo pri razvoju so odigrali tudi odprti standardi (angl. *open standards*), ki so omogočili, da so standardi enaki ne glede na to ali gre za lastniško ali odprtokodno programsko opremo. Šele odprti standardi so povzročili razvoj in inovacije na mnogih področjih (Friedrich, 2014).

Ekonomsko vprašanje, ki se pojavlja pri odprti kodi je, zakaj bi bila koda brezplačna. Odgovor na to vprašanje je v tem, da se kljub temu, da je sama koda odprta, pojavljajo nove priložnosti pri izvajanju storitev ter pomoči uporabnikom. Odprtokodna programska oprema večinoma nastaja z uporabo te kode in z nadgradnjo oziroma popravkom kode, ki jo uporabljajo podjetja za svoje delovanje. Zato splošno prepričanje, da takšno kodo sestavljajo navdušenci v prostem času in brezplačno ne velja. Mnoga podjetja uporabljajo odprtokodne sisteme, ki jih nato nadgradijo in spremenijo, da ustrezajo njihovim potrebam. (Adams, 2014).

Prednost v razvoju odprtokodnih sistemov je v tem, da razvijalci osnovne kode ne ustvarjajo na novo, vendar jo le prilagajajo svojim potrebam. Ker jo prilagaja več razvijalcev in ti nato prispevajo svoje inovacije skupnosti se koda hitreje razvija in ponuja več opcij, kot bi jo lahko razvilo katerokoli samostojno podjetje. Namesto da se podjetja borijo, da bi uporabniki prešli na njihovo zaprto platformo in plačevali za nadgradnje, vzdrževanje in podporo uporabnikom, se raje borijo za nudenje dodatnih storitev (Adams, 2014).

Prednosti odprtokodnega sistema se kažejo tudi v (Lerner & Tirole, 2002):

- alumni učinek: ker so odprtokodni sistemi velikokrat uporabljeni v izobraževalne namene, jih zaposleni dobro poznajo in se jim jih ni potrebno na novo učiti;
- prilagajanje in odpravljanje napak: zaradi močne podporne skupnosti se napake hitreje odpravijo ali pa je rešitev že na voljo;
- boljša merljivost rezultatov: zaprti sistemi pri nadgradnji dodajo nove funkcije vendar uporabniki ne morejo pregledati rešitve; pri odprtokodnih sistemih uporabniki lahko pregledajo rešitev in celo določijo ali je bila dobro izvedena;
- zavzetost: pri odprtokodnih sistemih je za dobro rešitev odgovoren programer, v zaprtih sistemih pa prihajajo ukazi od zgoraj;
- večja pretočnost tako znanja, kot človeškega kapitala: zaprti sistemi postavijo programerja v zaprto okolje in tako težje prehaja v druge sisteme, pri odprtokodnih rešitvah tega problema ni;
- priznanje: odprtokodni sistemi omogočajo priznanje za dobro delo in za prispevek skupnosti. V zaprtih sistemih se ne ve kdo je kaj prispeval;
- organizacija in upravljanje: prednost odprtokodnih sistemov je v njegovi modularnosti in zabavi ob reševanju problemov; kljub temu je potreben dober vodja, ki zna držati skupaj celoten projekt.

## 4.3 Primerjava med odprtokodnimi programskimi rešitvami za podatkovno rudarjenje

Raziskava, narejena na primerjavi šestih odprtokodnih programskih orodjih za podatkovno rudarjenje, je pokazala, da ne obstaja najboljša odprtokodna rešitev za podatkovno rudarjenje. Vsako orodje ima svoje prednosti in slabosti. Odločiti se je potrebno glede na specifične naloge, ki jih želimo izvajati (Jović, Brkić, & Bogunović, 2014).

Druga raziskava med šestimi odprtokodnimi orodji za podatkovno rudarjenje je pokazala, da so določena orodja bolj primerna za začetnike, druga, med njimi tudi Orange, pa zahtevajo že nekaj programerskega znanja in so bolj primerna za napredne uporabnike (Rangra & Bansal, 2014).

Odprtokodna orodja za podatkovno rudarjenje ponujajo dobre grafične vmesnike, ki so osredotočeni na uporabnost. So fleksibilna preko vizualnega programiranja ali preko uporabe skriptnega jezika. Ker imajo vedno več uporabnikov, se lahko pričakuje, da se bodo še naprej tehnično razvijala in ponujala vse učinkovitejšo

funkcionalnost. Namesto specifičnega področja bodo pokrivala več problematik in ponujala možnosti za reševanje problemov različnih področjih (Rangra & Bansal, 2014).

# 5 PODATKOVNO RUDARJENJE

Podatkovno rudarjenje spada v področje strojnega učenja. To je veja raziskav umetne inteligence. Pri strojnem učenju poskušamo računalnik naučiti obnašanja na podlagi podatkov iz preteklosti, za napovedovanje rezultatov v prihodnosti (Kononenko & Robnik Šikonja, 2010).

Kot pravita Kononenko in Robnik Šikonja (2010), je ena najpogostejših uporab stojnega učenja uvrščanje ali klasifikacija. Za klasifikacijo lahko uporabljamo razne metode, med katerimi so najbolj pogosto uporabljana odločitvena drevesa, Bayesov klasifikator, klasifikator z najbližjimi sosedi, diskriminantne funkcije, nevronske mreže, linearna regresija in druge.

## 5.1 Klasifikacijske metode

Preizkušanje različnih modelov je bistvo podatkovnega rudarjenja. Model, ki bi zagotavljal nadpovprečne rezultate v različnih situacijah pač ne obstaja (Tsiptsis & Chorianopoulos, 2009).

*Slika 9: Algoritmi za podatkovno rudarjenje*


*Vir: A. Orozco Zuluaga, Data Mining, a useful tool in Bussines Intelligence, 2011.*

## 5.2 Metode podatkovnega rudarjenja za napovedovanje prehoda strank h konkurenci

Za izbiro najprimernejše metode za napovedovanja prehoda strank h konkurenci je potrebno poleg same točnosti napovedi upoštevati še hitrost, interoperabilnost, robustnost in preprostost uporabe. Zelo pogosto se za napovedovanje prehoda strank h konkurenci uporabljajo logistična regresija, odločitvena drevesa in nevronske mreže, ki vse spadajo med individualne klasifikatorje (Clemente, Ginger Bosch, & San Matias, 2014).

### 5.2.1 Odločitvena drevesa

Odločitvena drevesa nastanejo s pomočjo algoritma, ki glede na oceno informativnosti spremenljivk izbira atribute in ustrezne podmnožice vrednosti teh atributov. Iz teh informacij nastane drevesna struktura, s pomočjo katere lahko uvrščamo podatke, za katere ne vemo kateremu razredu pripadajo in jim na ta način določimo razred. Podatki morajo biti za uporabo v odločitvenih drevesih diskretizirani, npr. dodeljeni v razrede, saj metode odločitvenih dreves ne znajo delati z zveznimi podatki (Kononenko & Robnik Šikonja, 2010).

### 5.2.2 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator izračuna pogojne verjetnosti za vsak razred pri danih vrednostih vseh atributov za vsak novi primer, ki ga želimo klasificirati. (Kononenko & Robnik Šikonja, 2010).

Ta metoda predpostavi pogojno neodvisnost atributov pri danem razredu. Za učno množico nato uporabi oceno vseh potrebnih verjetnosti za izračun pogojne verjetnosti posameznega razreda. Ker implementacije naivnega Bayesovega klasifikatorja po navadi predpostavljajo samo diskretne atribute je potrebno zvezne pred tem diskretizirati. (Kononenko & Robnik Šikonja, 2010).

### 5.2.3 Nevronske mreže

Algoritmi umetnih nevronskih mrež delujejo tako, da poskušajo oponašati biološke nevronske mreže. Nevron je predstavljen z elementom, ki zna seštevati vhodne signale, ki so bili predhodno uteženi, rezultat pa normalizira s pragovno funkcijo. Te nevrone lahko povezujemo v kompleksne nevronske mreže. Za klasifikacijo se pogosto uporabljajo usmerjene več nivojske umetne nevronske mreže, ki so kaskadno povezane v več nivojev. Vhodni nevroni, k ustrezajo atributom, za tem en ali več nivojev skritih nevronov in na koncu izhodni nevroni, ki ustrezajo razredom. Učni algoritem v procesu učenja nastavlja uteži na povezavah med nevroni, da je na koncu klasifikacijska napaka čim manjša. Nevronska mreža med klasificiranjem dobi na vhodu vrednosti atributov za vsak novi primer. Na naslednjem nivoju vsak nevron izračuna svoj izhod, kot uteženo vsoto svojih

vhodnih signalov, izračun zadnjega nevrona pa vrne razred (Kononenko & Robnik Šikonja, 2010).

### 5.2.4 Regresijski modeli

Regresijski modeli po navadi zagotavljajo neposredno povezavo med pojasnjevalnimi spremenljivkami in napovedano vrednostjo. Če kot uporabniki razumemo model in relacije, ki pripeljejo do rezultata, potem lažje sprejmemo model kot primeren za uporabo (Yang & Chiu, 2014).

## 5.3 Logistična regresija

Ker bomo v nalogi za napovedovanje uporabili logistično regresijo, jo bomo na tem mestu podrobneje pojasnili.

V statistiki se logistična regresija ali logit regresija oziroma logit model uporablja za napovedovanje verjetnosti (Bishop, 2006). Logistična regresija je postala široko uporabljana metoda za analizo dihotomnih izhodnih spremenljivk. Pogostost uporabe izhaja iz prisotnosti te metode v mnogih statističnih paketih ter preprostosti uporabe in razlage rezultatov (Hosmer, Hosmer, Le Cessie, & Lemeshow, 1997).

Logistična regresija sledi splošnim načelom, ki veljajo za linearno regresijo. Razlika med logistično regresijo in linearno regresijo je v tem, da je pri logistični regresiji izhodna spremenljivka binarna oziroma dihotomna. Logistična regresija meri razmerja med diskretno odvisno spremenljivko in eno ali več neodvisnih spremenljivk, ki so največkrat zvezne, z uporabo ocene verjetnosti kot predvidenih vrednosti odvisne spremenljivke (Hosmer et al., 1997).

Druga večja razlika med linearnim in logističnim modelom je v pogojni distribuciji izhodne spremenljivke. V linearnem modelu se predpostavlja normalna porazdelitev okrog srednje vrednosti s konstantno varianco, kar pa ne drži pri dihotomnih izhodnih spremenljivkah, kjer lahko izrazimo izhodno vrednost z eno ali drugo vrednostjo. Logistična regresija distribuira po distribuciji Bernoullija in ne po distribuciji Gaussa. (Hosmer Jr., Lemeshow, & Sturdivant, 2013)

Bernoullijeva distribucija je diskretna oz. binarna. Tako omogoča samo dve končni vrednosti in sicer 1 ali 0. Uporablja se za napovedovanje verjetnosti, kjer 1 pomeni uspeh in 0 neuspeh (Evans, Hastings, & Peacock, 2000). Enačba verjetnosti je predstavljena v enačbi 1.

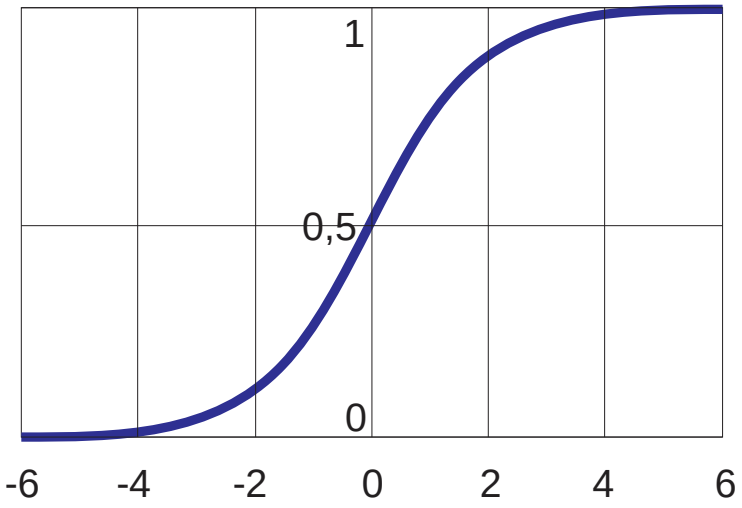
$$P_{(n)} = \begin{cases} 1-p & n=0 \\ p & n=1 \end{cases} \quad (1)$$

Logistična regresija napoveduje verjetnost, da se bo dogodek zgodil (Hosmer Jr. et al., 2013). Formula logistične regresije je zapisana v enačbi 2.

$$P(x) = \frac{e^{\beta_0 + x*\beta}}{1+e^{\beta_0 + x*\beta}} = \frac{1}{1+e^{-(\beta_0 + x*\beta)}} \quad (2)$$

Logistična regresija prikaže možnost in ne-nemožnost določenega dogodka (Simeunović & Milosavljević, 2009).

*Slika 10: Funkcija logistične regresije*




Iz osnovne predpostavke je razvidno, da se v primeru, ko je $x = -\infty$ dobi vrednost 0 in v primeru, da je $x = \infty$ dobi vrednost 1. Kadar je x = 0, pa dobimo vrednost 0,5 (Simeunović & Milosavljević, 2010). Funkcijo logistične regresije smo prikazali na sliki 10.

Kadar nas zanimajo verjetnosti dogodka bi želeli, da se funkcija $P(x)$ obnaša linearno. Vsak prirastek od x bi dodal ali odvzel določen del verjetnosti. Vendar smo omejeni s tem, da se mora P nahajati med 0 in 1. Z $\log P(x)$, kot linearno funkcijo spremenljivke x, bi se za vsako odvisno spremenljivko pomnožila verjetnost za določen del. Vendar so logaritmi neskončni le v eni smeri. Ta problem rešimo z logistično transformacijo, ki nam omogoča neskončen razpon in se obnaša kot linearna funkcija (Shalizi, 2014).

Inverzna oblika logistične funkcije je logaritmična funkcija, ki je prikazana z enačbo 3 in predstavlja logit transformacijo.

$$\log \frac{P}{1-P} = \log it(P) \quad (3)$$

Enačba za logit transformacijo oziroma naravni logaritem obetov je prikazana z enačbo 4.

$$\mathrm{g}(x) = \ln\left[\frac{P_{(x)}}{1-P_{(x)}}\right] = \beta_0 + \beta x \quad (4)$$

Ali drugače povedano z enačbo 5.

$$\frac{P_{(x)}}{1-P_{(x)}} = e^{\beta_0+\beta x} \quad (5)$$

$\mathrm{g}(x)$ predstavlja logit funkcijo določenih linearnih vrednosti x. $P_{(x)}$ je verjetnost, da odvisna spremenljivka vrne pozitiven rezultat. $\beta_0$ predstavlja linearni del enačbe in je verjetnost, da vrne odvisna spremenljivka negativen rezultat. $\beta x$ je regresijski koeficient pomnožen z napovednimi vrednosti. Eksponentno funkcijo dobimo z osnovo *e*.

Enačba logit transformacije lahko vzame katerokoli število med $-\infty$ in $\infty$ in vrne vrednost med 0 in 1, kar razumemo kot verjetnost. (Simeunović & Milosavljević, 2010).

Logit transformacija je linearna v parametrih, lahko je zvezna in ima meje med $-\infty$ in $+\infty$ odvisno od razpona x. Logit transformacija omogoča povezavo med linearno regresijo in funkcijo verjetja. Logit je preprosto spremeniti nazaj v obete (Hosmer Jr. et al., 2013).

Formula za razmerje obetov (angl. *Odds Ratio*) (v nadaljevanju OR) je prikazana v enačbi 6.

$$OR = obet\big(P(x+1)\big)/obet\big(P(x)\big) = \frac{e^{\beta_0+\beta_1(x+1)}}{e^{\beta_0+\beta_1 x}} = e^{\beta} \quad (6)$$

β predstavlja povečanje logaritma razmerja obetov, če se x poveča za 1 enoto. Bolj pomembno pa je, da je $e^{\beta}$ razmerje obetov med dvema skupinama, ki se v x ločita za 1 (Hosmer Jr. et al., 2013)

Iz tega pridemo do naravnega logaritma, prikazanega z enačbo 7.

$$\ln(OR) = \beta \quad (7)$$

## 5.4 Logistična regresija kot napovedni model

Za zmanjšanje napake je potrebno določiti Y=1 kadar je $p \geq 0{,}5$ in Y=0, kadar je $p < 0{,}5$. To pomeni, da kadar je $\beta_0 + x * \beta$ pozitivna iščemo 1 ter 0, kadar ni.

Logistična regresija nam da linearni klasifikator. Meja odločitve, ki ločuje predvidena razreda, je rezultat enačbe, ki je točka, kadar je x enodimenzionalen, črta, kadar je dvodimenzionalen itd. Tako logistična regresija ne pokaže samo meje med razredoma ampak tudi pravi, da so verjetnosti razreda odvisne od razdalje od meje v določenem smislu in da se bližajo ekstremoma (0, 1) tem hitreje, kadar je $\beta$ večji. To so razlogi zakaj je logistična regresija v napovednih modelih tako močna in predstavlja več kot le klasifikator. Daje dobre, bolj podrobne napovedi in jo lahko uporabimo v različnih variantah (Shalizi, 2014). Enačba linearnega klasifikatorja je prikazana v enačbi 10.

$$\beta_0 + x * \beta = 0 \qquad (10)$$

Če združimo obe enačbi dobimo drugo obliko logistične regresije, prikazano v enačbi 11.

$$\log it\left(P\left\{y = \frac{1}{x}\right\}\right) = x\beta = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \quad (11)$$

Ker logistična regresija napoveduje verjetnosti in ne samo razredov jo lahko uporabimo kot funkcijo verjetja. Za vsako testno točko podatkov imamo vektor lastnosti, $x_i$, in razred $y_i$. Verjetnost za ta razred je lahko p, če je $y_i$=1 ali 1-p, če je $y_i$=0 (Shalizi, 2014). Enačba verjetja je prikazana v enačbi 12.

$$V(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i)^{1-y_i}) \qquad (12)$$

## 5.5 Učni sistem in klasifikator

Do sedaj smo v nalogi opisali problemsko domeno in okolje v katerem se pojavlja. Pojasnili smo delovanje metod s katerimi bomo poskušali priti do rešitve. Opisali smo tehnično delovanje posameznih komponent in kako jih lahko ločeno uporabljamo. Vse to pa moramo povezati v sistem, ki bo deloval kot rešitev zastavljenega problema. Ker je jedro naše rešitve klasificiranje strank, bomo na tem mestu podrobneje opisali postopek klasifikacije in sodelujoče entitete v tem postopku.

Da bi lahko izdelali rešitev moramo najprej določiti problem. V našem primeru je problem določitev strank, ki bodo v bližnji prihodnosti menjale telekomunikacijskega operaterja. Rešitev problema bo predstavljal matematični model, s katerim bomo iskali podobnosti v obnašanju strank, ki so že menjale operaterja, s tistimi, ki ga še niso. Za modeliranje obnašanja strank bomo uporabili model logistične regresije, s katerim bomo iskali podobnosti v obnašanju.

Matematični model, ki se nauči znanja o obnašanju strank imenujemo učni sistem. Učni sistem (angl. *learner*) je sistem, ki kot vhod sprejme niz primerov ($x_i$, $y_i$), kjer velja, da so $x_i=(x_{i,1}…x_{id})$ vhodne spremenljivke, $y_i$ pa je pripadajoči rezultat in kot rezultat vrne klasifikator.

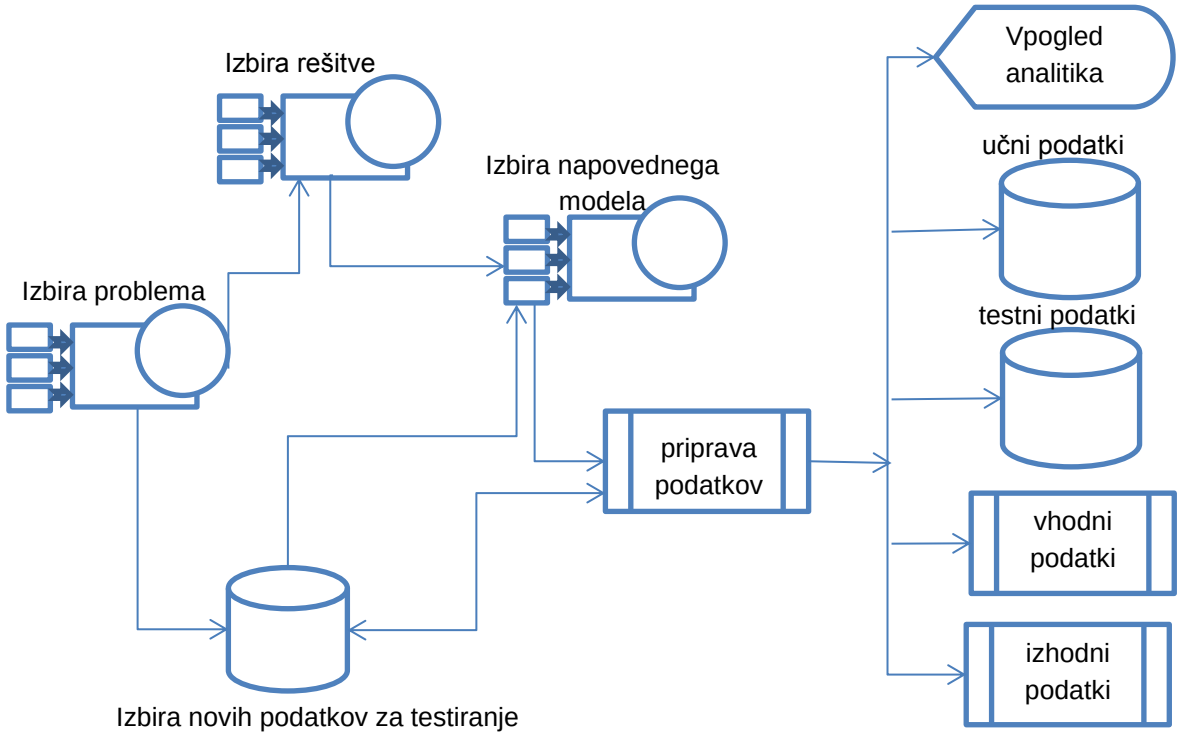
Klasifikator (angl. *classifier*) pa je sistem, ki sprejme vektor diskretnih ali zveznih spremenljivk, kot rezultat pa vrne eno samo diskretno vrednost imenovano razred. (Domingos, 2012).

Učni sistem se torej nauči nekega pravila na podlagi niza testnih podatkov z znanim rezultatom in vrne sistem imenovan klasifikator. Na podlagi pravil iz učnega sistema nato ustrezno razporedi niz podatkov, za katerega razred ni znan, tako da mu razred določi. Klasifikator v nekaterih tujih virih včasih poimenujejo z angleško besedo *predictor*.

V naslednjem koraku se moramo odločiti kateri nabor podatkov bomo uporabljali za učni sistem in kateri nabor podatkov za testni sistem, s katerim bomo poskrbeli, da se podatki ne bodo preveč prilegali učni množici.

Pogosto v tem postopku opravimo še evalvacijo rezultatov na nizu testnih podatkov, da lahko s tem ocenimo kvaliteto modela (Vuk & Curk, 2006). Ker popolnih modelov ni mogoče narediti, njihovo kvaliteto ocenimo s pomočjo različnih meril. Med te sodijo tabela napačnih klasifikacij (angl. *Confusion Matrix*), ROC krivulja (angl. *Receiver Operating Characteristic*) (v nadaljevanju ROC krivulja), klasifikacijska točnost (angl. *Classification Accuracy*) in druga.

*Slika 11: Shema sodelujočih entitet pri klasifikaciji*




*Vir: D. Pyle, Data preparation for Data mining, 1999, str. 88, slika 3.1.*

Za pravilno delovanje celotnega sistema potrebujemo še dve entiteti in sicer funkcijo za sprejem vhodnih podatkov, ki jih želimo klasificirati in funkcijo, ki vrača rezultate klasifikacije v nam berljivi obliki. Vse opisane entitete vidimo na sliki 11.

Tukaj smo govorili samo o jedru naše aplikacije, to je o konceptu sodelovanja podatkov in uporabljenega modela. V nadaljevanju pa bomo opisali še tehnične korake, ki so potrebni, da celotna rešitev sploh deluje.

# 6 IZDELAVA REŠITVE

Ugotovili smo, da na trgu obstaja veliko profesionalnih, lastniških programskih paketov za reševanje problematike prehajanja strank h konkurenci. Zanimalo nas

je, ali lahko pridemo do enakega napovednega modela tudi z odprtokodnimi programi za podatkovno rudarjenje.

Izmed več odprtokodnih rešitev smo izbrali Orange, saj se je odlično odrezal na že omenjenih primerjavah. Omogoča nam, da implementirane funkcije in metode uporabimo v lastnih programskih rešitvah, ker programska koda ni skrita.

Pokazali bomo, da je programsko rešitev mogoče sestaviti iz že narejenih komponent, ki same po sebi odlično delujejo, manjka pa jim uporabnost in enostavnost.

Izdelali bomo programsko rešitev za iskanje naročnikov, katerih obnašanje kaže na veliko verjetnost, da bodo prestopili h konkurenci. Z analitičnim napovednim modelom, bomo za vsakega posameznika izračunali, ali je zanj verjetno, da se nagiba k menjavi operaterja in te rezultate vrnili uporabniku aplikacije, ki dela na področju neposrednega trženja, da bo s tem dobil informacijo, katere stranke lahko kontaktira z največjo verjetnostjo uspeha.

Ker se različni segmenti uporabnikov precej razlikujejo v obnašanju, jih je smiselno obravnavati ločeno. V našem primeru se bomo osredotočili na naročnike storitev, to so stranke, ki imajo s podjetjem sklenjeno pogodbo u uporabi storitev in poravnavajo račun enkrat mesečno in so hkrati samo fizične osebe.

Aplikacija bo izdelana v obliki spletne strani, s pomočjo katere bo uporabnik lahko naložil dva različna seznama s podatki o strankah. Prvi seznam mora vsebovati niz strank, ki so že menjale operaterja in v približno enakem številu niz strank, ki ga niso. To imenujemo učni seznam (angl. *learn data*). Manjši del učnega seznama bo uporabljen kot testni seznam (angl. *test data*) in ga bo program uporabil za evalvacijo uspešnosti modela. Ta seznam program sam sestavi tako, da izbere naključne primere iz učnega seznama. Drugi seznam pa je niz podatkov, ki jih želimo klasificirati in za njih ne poznamo razreda. Ta seznam je lahko poljubno velik, omejeni smo le s količino pomnilnika, ki ga lahko porabimo.

Uporabnik aplikacije bo nato lahko z gumbom sprožil postopek računanja, ki bo po končanem delu vrnil seznam z rezultati klasifikacije. Ta seznam si bo uporabnik lahko naložil nazaj na svoj računalnik s pritiskom na primeren gumb.

## 6.1 Pregled uporabljenih orodij

Ker so knjižnice paketa Orange napisane v jeziku Python, smo poiskali tudi spletni strežnik, ki je napisan v istem jeziku, s katerim smo povezali vse komponente.

### 6.1.1 Programski jezik Python in grafične knjižnice Qt

Python je dinamično tipiziran visoko nivojski programski jezik, ki lahko sam upravlja s pomnilnikom, podpira objektno, funkcionalno, proceduralno in strukturirano programiranje. Licenciran je kot odprtokodni program (Python, 2014). Eno izmed glavnih vodil je berljivost, zaradi svoje enostavnosti pa omogoča pisanje programov z uporabo manj vrstic kode, kot v nekaterih drugih jezikih (Summerfield, 2007). Prevajalniki za Python delujejo na mnogih operacijskih sistemih. Obstajajo tudi orodja s pomočjo katerih lahko programe napisane v Pythonu prevedemo v samostojne aplikacije, ki ne potrebujejo namestitve Python prevajalnika (Pyinstaller, 2014).

Qt je odprtokodno ogrodje za razvoj aplikacij z grafičnim uporabniškim vmesnikom, ki lahko teče na različnih okoljih. Uporablja jezik C++, kompatibilno pa je tudi z drugimi jeziki in sistemi. V paketu Orange se uporablja kot ogrodje za izdelavo grafičnega dela aplikacije z uporabniškim vmesnikom, ter grafičnimi ikonami in za grafično prikazovanje rezultatov analiz, ki jih lahko naredimo v programu Orange, to so grafikoni, tridimenzionalni koordinatni sistemi.

### 6.1.2 Programska rešitev za podatkovno rudarjenje Orange

Orange je nastal na Fakulteti za računalništvo in informatiko Univerze v Ljubljani (Orange, 2014) in je napisan v C++ in Pythonu (Demšar et al., 2013), njegovi začetki segajo v telo 1997 in je eno najstarejših orodij tega tipa.

Njegove glavne lastnosti so enostavnost, interaktivnost preko pisanja skript in razvoj z uporabo komponent. Uporabljamo ga lahko na dva načina. Eden je s pomočjo grafičnega uporabniškega vmesnika, ki omogoča delo z ikonami, drugi način pa je uporaba skript v programski kodi napisani v jeziku Python.

Vsebuje komponente za (Demšar et al., 2013):

- upravljanje s podatki in pred procesiranje (branje in pisanje podatkov, filtriranje, vzorčenje, vnašanje, normalizacijo in druge);
- klasifikacijo (različni algoritmi nadzorovanega strojnega učenja – drevesa, gozdove, Bayesove klasifikatorje in pravila);
- regresijo (linearno, logistično, Lasso, metodo najmanjših kvadratov in druge);
- asociacije (za iskanje asociativnih pravil);
- združevanja (za ojačitve in druge);
- gručenje (k-means in hierarhične metode);
- evalvacije (navzkrižno validiranje, ocenjevanje in drugo) in
- projekcije (metodo osnovnih komponent in podobne).

Odlikuje ga enostavnost dela s podatki in preprostost izdelave delovnih tokov iz obstoječih komponent. Uporablja in razvija se že več kot 15 let in ima veliko število uporabnikov ter močno podporo na spletnih forumih. Obstaja tudi precej dokumentacije, navodil za uporabo in praktičnih primerov uporabe. Poleg naštetega omogoča še grafične prikaze rezultatov analiz, kot so distribucije, razni diagrami in razpršilni diagrami. Žal ne podpira PMML (angl. *Predictive Model Markup Language*) oblike za izmenjavo podatkov in ima relativno omejen vmesnik za neposredno delo s podatkovnimi bazami.

### 6.1.3 Spletno ogrodje Web2py

Web2Py (Web2py, 2014) je brezplačno odprtokodno ogrodje za hitro izdelovanje razširljivih, varnih in podatkovno podprtih spletnih aplikacij v programskem jeziku Python (Massimo Di, 2011).

Zasnovano je tako, da olajša pisanje enoličnih opravil, kot so npr. spletni obrazci. Njegove razvojne paradigme vključujejo hiter razvoj, konvencijo pred konfiguracijo in deluje na principu MVC (angl. *Model View Controller*) (v nadaljevanju MVC), kar bi lahko prevedli kot princip modela, pogleda in krmilnika.

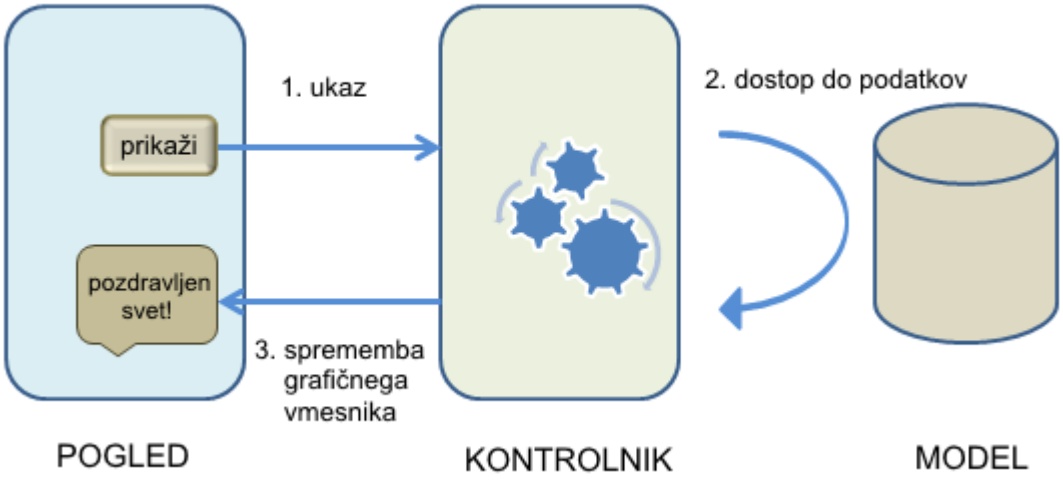
Podpira protokol http, spletne seje in piškotke, vsebuje module za nadzor varnosti, podpira abstrakcijo podatkovnih baz, internacionalizacijo, jQuery in Ajax in še veliko dodatnih funkcionalnosti (Massimo Di, 2011).

Ker je ogrodje Web2Py napisano v jeziku Python, kakor tudi knjižnice Orange, bo souporaba obeh sistemov v tej aplikaciji poenostavila delo, ker bomo celo rešitev napisali v enem samem jeziku.

Delovanje Web2py je neodvisno od operacijskega sistema, saj deluje na vseh sistemih na katerih je nameščen prevajalnik za jezik Python.

MVC je programersko arhitekturno ogrodje za izdelavo računalniških aplikacij. Predstavlja miselno ločitev med objekti, ki sestavljajo celoten sistem, za lažje razumevanje in delo z njimi. Predstavljen je na sliki 12.

*Slika 12: Shema delovanja MVC aplikacij*




*Vir: MVC, 2014.*

Krmilnik (angl. *controller*) je tisti del sistema, ki vsebuje jedro programske logike in skrbi za interakcijo med modelom in pogledom. Med drugim pošilja ukaze v model, da se osveži stanje modela. Ukaze lahko pošilja tu v pogled, da se osveži pogled na podatke

Model (angl. *model*) predstavlja nivo podatkov in podatkovnih baz. Na tem nivoju se napiše vse procedure, ki jih potrebujemo za delo s podatki. Model sporoči povezanim pogledom in kontrolnikom, da je prišlo do spremembe stanja. To omogoči pogledu, da osveži vsebino in kontrolniku, da osveži nabor ukazov.

Pogled (angl. *view*) se uporablja za prikaz vsebine oz podatkov. Po navadi je to spletna stran, čeprav je lahko tudi katerikoli drug uporabniški vmesnik.

## 6.2 Poraba pomnilnika

S stališča računalniške arhitekture lahko uporabniško programsko opremo delimo dve pomembni skupini in sicer na 32 bitne in 64 bitne aplikacije in operacijske sisteme. Podrobnejša razlaga bi presegala okvire te naloge. Razlika, ki je pomembno vplivala na izdelavo naše aplikacije, je v tem, da z 32 bitnimi Okni

(angl. *MS Windows*) teoretično ne moremo naslavljati več kot 4 gigabajte (4 GB) pomnilnika, v praksi pa zgolj približno 3 in pol gigabajte (3,5 GB). Okna narejena v 64 bitni arhitekturi teh omejitev nimajo, zato z njimi brez težav naslavljamo vse tja do 512 gigabajtov (512 GB) pomnilnika.

V spodnji tabeli vidimo omejitve porabe delovnega pomnilnika za različne verzije Microsoftovih Oken:

*Tabela 2: Maksimalna količina razpoložljivega pomnilnika pri različnih tipih operacijskega sistema Windows*

| Verzija MS Oken | Meja na X86 | Meja na X64 |
|---|---|---|
| Windows XP | 4 GB | 128 GB |
| Windows 7 Home Premium | 4 GB | 16 GB |
| Windows 7 Professional | 4 GB | 192 GB |
| Windows 8 | 4 GB | 128 GB |
| Windows 8 Professional | 4 GB | 512 GB |
| Windows 8 Enterprise | 4 GB | 512 GB |

*Vir: Memory Limits for Windows and Windows Server Releases, Microsoft Developer Network, 2014.*

Zaradi teh omejitev smo morali uporabiti 64 bitno različico jezika Python, knjižnic za Orange in ostalih potrebnih modulov. Na ta način smo lahko izkoristili skoraj vseh 16 gigabajtov (16 GB) pomnilnika, ki smo ga imeli na voljo ter naenkrat obdelali 350 megabajtov veliko datoteko s podatki, ki je vsebovala približno 300.000 vrstic in približno 90 stolpcev.

## 6.3 Viri podatkov

Podatke smo pridobili iz dveh virov. En vir je podatkovna baza CRM, ki vsebuje podatke o strankah kot so naročniški paket, spol, starost in podobno.

Drugi vir so podatki o prometu, torej o klicih in porabi paketnih storitev, kar smo pridobili iz podatkovnega skladišča, ki ga sicer podjetje potrebuje za obračunavanje storitev. Vsi podatki o porabi so grupirani na raven enega meseca.

Za vse numerične podatke, kot so npr. število klicev, število poslanih kratkih sporočil, minute porabljene v lastnem omrežju, število različnih klicanih številk in podobo, smo predhodno na nivoju podatkovne baze izračunali trend. Trend smo dobili tako, da smo izračunali povprečje za zadnje tri mesece in še povprečje za tri mesece pred tem. Iz tega bomo lahko sklepali o večanju ali manjšanju porabe določenih storitev.

Te podatke smo pripravili kot poizvedbo v podatkovni bazi, in pripravili opravilo, ki bo za vsak mesec obdelave podatkov izvedlo to poizvedbo in shranilo rezultate v tekstovno datoteko na neko določeno omrežno mesto, kjer bodo dostopni uporabniku naše aplikacije.

Podatki morajo biti zapisani v obliki, ki jo zahteva sistem Orange, kar vključuje tri vrstično glavo datoteke z opisi podatkovnih tipov in razreda, ter podatke ločene s tabulatorji.

## 6.4 Priprava podatkov za sistem Orange

Poizvedbe v podatkovnih bazah so bile narejene v jeziku SQL, od tam pa je bil narejen izvoz v besedilne datoteke, narejene po specifikaciji za Orange. Zraven spada še najpomembnejši meta podatek, to je enolična identifikacijska številka stranke. Ker pa ima vsaka stranka lahko sklenjenih več razmerij, analitiko pa izvajamo na nivoju posameznih razmerij, je glavni ključ vseh podatkov enolična številka razmerja, ki je prav tako meta podatek v isti datoteki.

Datoteka s podatki za Orange mora biti prirejena na poseben način (Orange, 2014). Za ločitev stolpcev v tabeli se uporablja tabulatorski znak, prve tri vrstice v datoteki pa so rezervirane. V prvi vrstici se nahajajo nazivi spremenljivk, v drugi vrstici so označbe za opise tipov teh spremenljivk in v tretji vrstici so zapisane dodatne zahteve, s katerimi povemo ali bomo določen podatek upoštevali ali ne. Primer prvih treh vrstic je prikazan na sliki 13.

*Slika 13: Primer glave datoteke, ki je zapisana v prvih treh vrsticah datoteke*

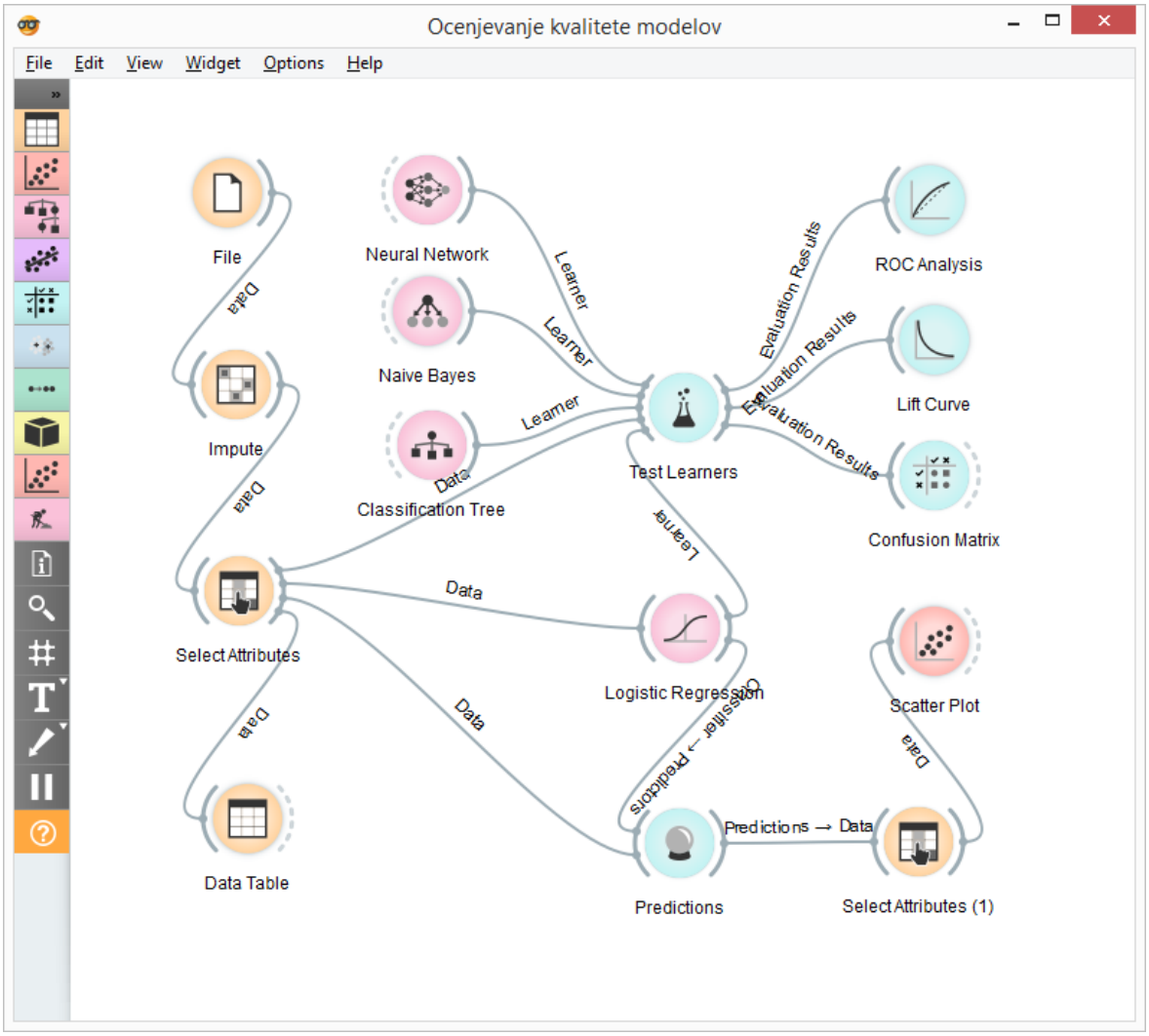
| ID | podatek1 | podatek2 | podatek3 | podatek4 | razred |
|---|---|---|---|---|---|
| d | d | c | d | c | d |
| m | | i | | | class |

V našem primeru mora biti naziv prvega stolpca ID in mora vsebovati m kot dodatno zahtevo, s katero določimo da se podatek v analizi ne upošteva, se pa vseeno ohrani. V zadnjem stolpcu z dodatno zahtevo določimo da gre za razred z uporabo možnosti class. V drugi vrstici so zapisani podatkovni tipi in sicer uporabimo možnost c za zvezne podatke in možnost d za diskretne. Če podatka ne želimo uporabiti, ga lahko v tretji vrstici označimo s črko i. Od četrte vrstice naprej so zapisani podatki, ločeni z vrsticami. Datoteka mora biti zapisana kot besedilna (tekstovna) datoteka s končnico tab.

## 6.5 Modeliranje v aplikaciji Orange

Problem smo najprej analizirali z manjšim številom strank v tabeli, kar v aplikaciji Orange, saj zaradi enostavnega dela omogoča hitro analitiko in preliminarni pregled podatkov. V aplikaciji Orange lahko na vizualen in intuitiven način brez predhodnega znanja programiranja in z nekaj osnovnega znanja statistike enostavno raziskujemo podatke in primerjamo uspešnost modelov.

*Slika 14: Ocenjevanje kvalitete modelov v aplikaciji Orange*




Na sliki 14 je prikazan delovni tok za primerjavo uspešnosti napovednih modelov. Jedro te primerjave je testni učni sistem (angl. *Test Learners*), ki smo ga povezali s podatki in s štirimi različnimi napovednimi modeli, to so logistična regresija, klasifikacijsko drevo, Bayesov klasifikator in nevronske mreže, kar je prikazano na sliki 15.

*Slika 15: Primerjava uspešnosti različnih napovednih modelov*

Evaluation Results

| | Method | CA | Sens | Spec | AUC | Brier |
|---|---|---|---|---|---|---|
| 1 | Logistic regression | 0.9703 | 0.9817 | 0.9190 | 0.9839 | 0.0456 |
| 2 | Classification Tree | 0.9834 | 0.9874 | 0.9655 | 0.9800 | 0.0308 |
| 3 | Naive Bayes | 0.9284 | 0.9664 | 0.7569 | 0.8770 | 0.1368 |
| 4 | Neural Network | 0.9606 | 0.9767 | 0.8879 | 0.9817 | 0.0575 |

V matrikah zmot na sliki 16 in 17 lahko vidimo, da je število napačnih klasifikacij pri odločitvenih drevesih nekoliko nižje, kot pri logistični regresiji.

*Slika 16: Matrika zmot za logistično regresijo*

Prediction

| | 0 | 1 | |
|---|---|---|---|
| **0** | 533 | 47 | **580** |
| **1** | 48 | 2572 | **2620** |
| | **581** | **2619** | **3200** |

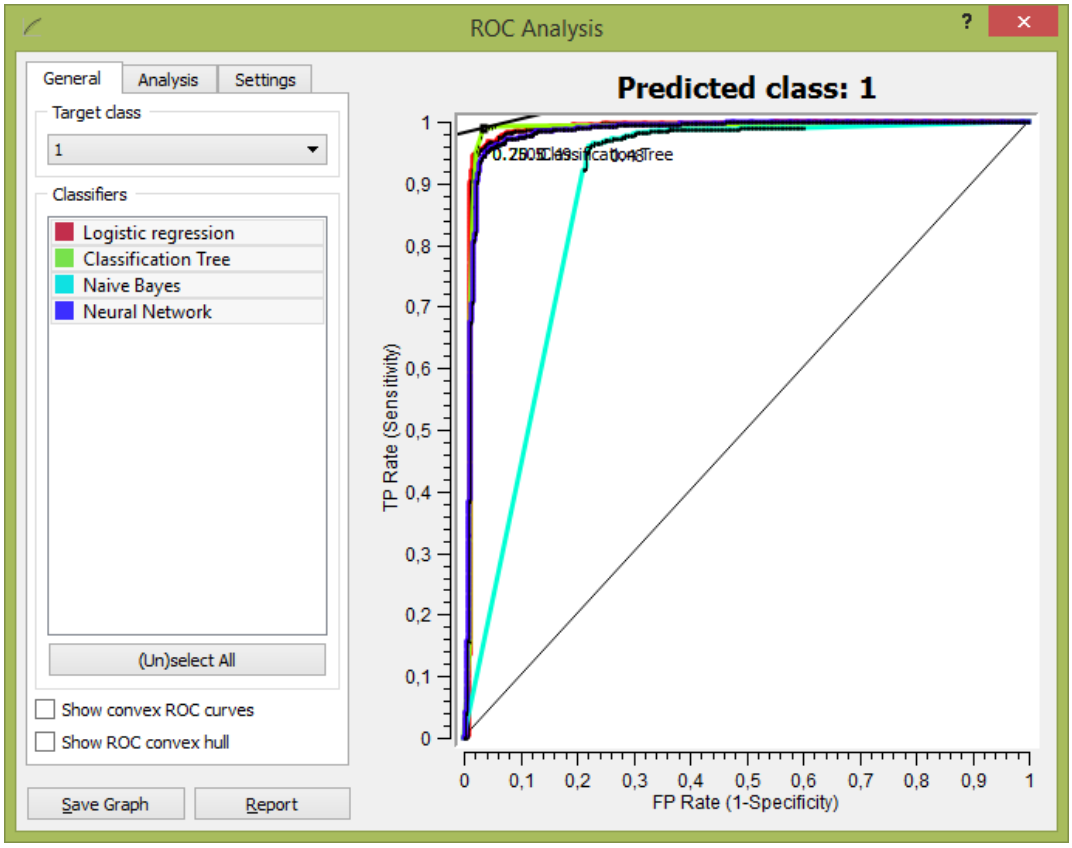
*Slika 17: Matrika zmot za odločitveno drevo*

Prediction

| | 0 | 1 | |
|---|---|---|---|
| **0** | 560 | 20 | **580** |
| **1** | 33 | 2587 | **2620** |
| | **593** | **2607** | **3200** |

Površina pod krivuljo (angl. *Area Under Curve*) pa je malenkost večja pri logistični regresiji, kot je prikazano na sliki 18. Modela nevronskih mrež in Bayesovega klasifikatorja sta se v obeh primerih izkazala za slabša.

*Slika 18: ROC krivulja za primerjavo uspešnosti napovednih modelov*




V tej nalogi smo kot napovedovalno metodo izbrali logistično regresijo. To je zelo pogosto uporabljana metoda za napovedovanje rezultatov, pri kateri lahko izhodna spremenljivka zavzame samo dve vrednosti, to je 0 ali 1.

### 6.5.1 Razred logistične regresije v paketu Orange

Konstruktor logistične regresije v paketu Orange (Orange, 2014):

*class* `Orange.classification.logreg.LogRegLearner` (*`remove_singular=0, fitter=None, **kwds`*)

Učni sistem za logistično regresijo – vrne učni algoritem (instanco od `LogRegLearner`) ali, če so podani podatki, pripadajoči (angl. *fitted*) model (instanco od `LogRegClassifier`).

Parametri:

- data (Orange.data.Table) – podatkovna tabela; lahko vsebuje diskretne ali zvezne podatke;
- weight_id (*int*) –ID uteži meta atributa;
- remove_singular (*bool*) – samodejno odstranjevanje konstantnih vrednosti in singularnosti (privzeto: *False*);
- fitter – uporabljena metoda (privzeto: LogRegFitter_Cholesky);
- stepwise_lr (*bool*) – omoči izbor koeficientov po metodi stepwise (privzeto: *False*);
- add_crit (*float*) – prag za dodajanje koeficientov pri metodi stepwise (privzeto: 0.2);
- delete_crit (*float*) – prag za odvzemanje koeficientov pri metodi stepwise (privzeto: 0.3);
- num_features (*int*) – število koeficientov pri metodi stepwise (privzeto: -1, brez omejitev).

Vrača: klasifikator za logistično regresijo. Izvede oblikovanje modela na podlagi danih podatkov: .`__call__`(*data*, *weight=0*)

Parametri:

- data (Table) – podatkovne instance;
- weight (*int*) – identifikator meta podatka z utežmi.

Vrača: LogRegClassifier
*class* `Orange.classification.logreg.LogRegClassifier`

To je klasifikacijski model za logistično regresijo. Shrani ocenjene vrednosti regresijskih koeficientov in njihovih p-vrednosti, ki jih uporabi za napovedovanje razredov in njihovih verjetnosti. Vrača naslednje vrednosti:

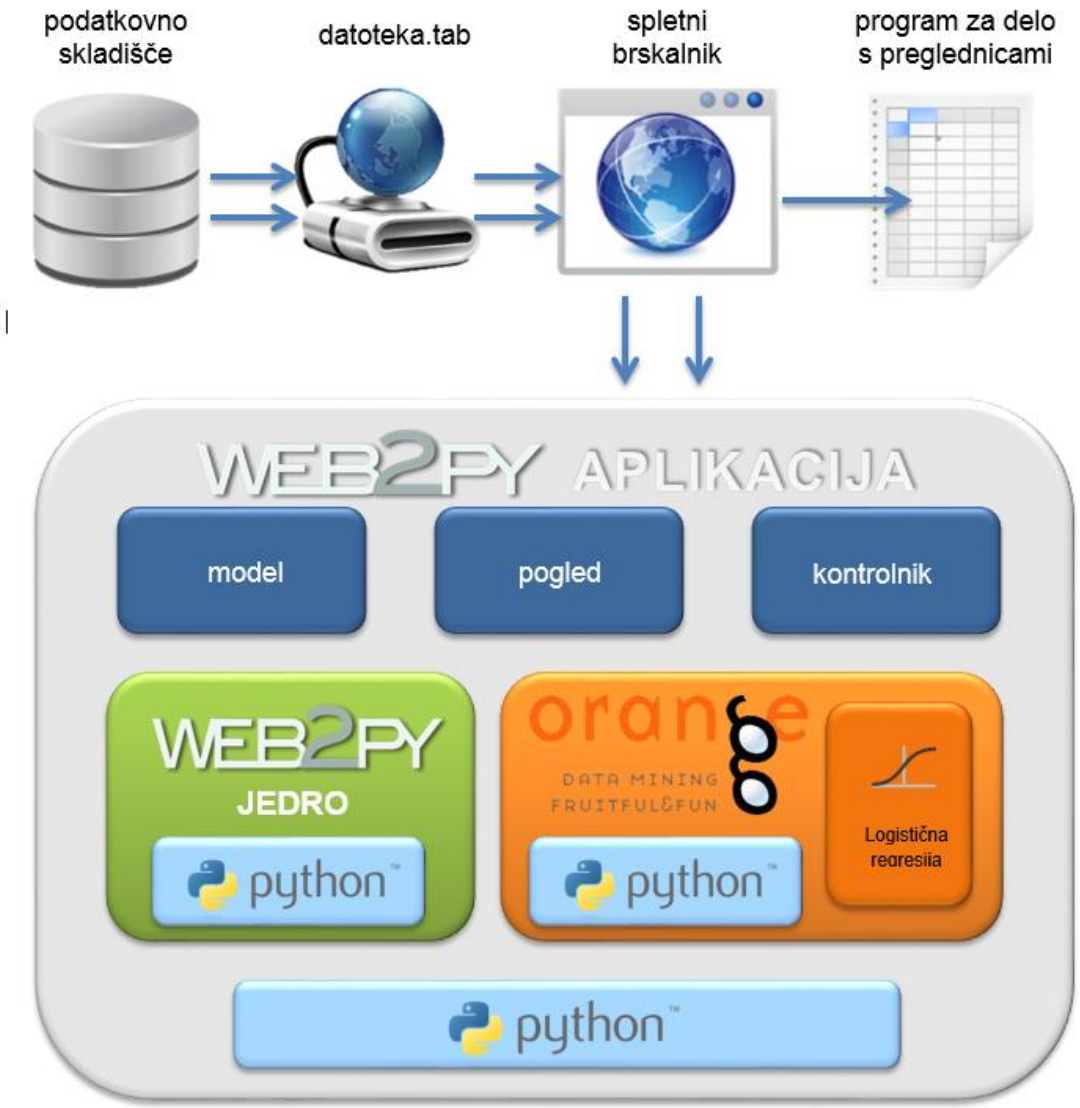
- `beta`: ocenjene vrednosti regresijskih koeficientov;
- `beta_se`: ocenjene standardne napake regresijskih koeficientov;
- `wald_Z`: Wald Z statistika za ocenjene vrednosti regresijskih koeficientov;
- `P`: seznam p-vrednosti beta koeficientov, to je verjetnost da se koeficient beta razlikuje od 0.

### 6.5.2 Arhitektura rešitve

Kot smo že omenili bomo delovanje aplikacije nadzorovali s pomočjo spletne aplikacije ki bo povezana z ostalimi deli kot je prikazano na sliki 19. Na spletni

strani bomo določili katere datoteke se preberejo iz spletnega mesta kamor so bile odložene po pripravi v podatkovnem skladišču. Proces bo te datoteke poslal v jedro sistema, ki ga nadzoruje spletni in aplikacijski strežnik Web2py. Za to skrbi del, ki ga imenujemo pogled. Od tam bodo podatki poslani v del imenovan kontrolnik, ki bo uporabil funkcijo logistične regresije, ki je sicer dela paketa Orange. Nato bo kontrolnik shranil rezultate na disk, objektu pogled pa bo vrnil sporočilo, da je operacija končana. Uporabnik bo nato imel možnost izpisa rezultatov na spletni strani ali pa jih bo lahko shranil, za uporabo v programu za delo s preglednicami. Arhitektura celotne rešitve je prikazana na sliki 19.

*Slika 19: Arhitektura celotne rešitve s sodelujočimi komponentami*




### 6.5.3 Uporaba programske kode

Za uporabo knjižnic paketa Orange najprej uporabimo ukaz `import Orange` v naslednjem koraku pa naložimo podatke v dve posebni tabeli definirani v paketu Orange. Podatki iz ene tabele bodo namenjeni učenju algoritma, del teh podatkov pa bo namenjen za testiranje uspešnosti modela. V drugo tabelo shranimo podatke, ki jih želimo klasificirati.

Nato priredimo objektu `logreg` učni model z nastavitvami. Nastavitev remove_singular na 1 pomeni, da model izloči morebitne singularnosti, nastavitev stepwise_lr na 0, pa pomeni da ne bomo uporabljali učenja stepwise. Shranimo še vrednost spremenljivke target, s katero bomo določili katera vrednost razreda nas zanima kot rezultat napovedi.

```
logreg = Orange.classification.logreg.LogRegLearner(train,
remove_singular = 1, stepwise_lr = 0)
target = 1
```

Določiti je potrebno tudi tip, ki ga vrača učni model in sicer nas zanimajo verjetnosti:

```
return_type = Orange.classification.Classifier. GetProbabilities
```

Če nas zanimajo regresijski koeficienti jih lahko izpišemo s pomočjo funkcije:

```
print Orange.classification.logreg.dump(logreg)
```

Model, ki smo ga naučili, je zdaj shranjen v objektu logreg, če pa želimo klasificirati podatke, katerih razreda ne poznamo in jih izpisati, naredimo to s spodnjo proceduro:

```
print 'Napovedi:'
print "ID" + "\t" + "Razred"
name = "ID "
x=0
for e in data_to_classify[:50]:
 print data_to_classify[x][name], "\t", logreg(e)
 x+=1
```

Dodatno lahko izpišemo še vrednosti znanih razredov iz učnih podatkov s proceduro:

```
print "Testni podatki:\n"
print "ID" + "\t" + "Verjetnost" + "\t" + "Razred"
name = "ID "
x=0
for d in test[:25]:
```

```
 print data[x][name], "\t", "%5.3f" % logreg(d, return_type)[target],
"\t", d.getclass()
 x+=1
```

Če pa nas zanima, kako natančna je bila klasifikacija, jo izračunamo in izpišemo tako:

```
correct = 1.0
for x in data:
 if logreg(x) == x.getclass():
 correct += 1
print "Natančnost klasifikacije:", correct / len(data)
```

Sestavljen model, ki ga apliciramo nad podatki, za katere ne vemo, v kateri razred spadajo, nam še ne ponuja celotne uporabnosti. Da bi dosegli cilj, uporabno aplikacijo, s katero bo mogoče klasificirati odjemalce ali bodo prešli h konkurenčnemu operaterju ali ne, je potrebno izdelati še uporabniški vmesnik.

### 6.5.4 Web2Py koda s komentarji

Odločili smo se, da za uporabniški vmesnik izberemo spletno stran, ker smo prepričani, da je uporaba spletnih aplikacij postala edini primeren način interakcije uporabnikov z računalniki. Uporaba spletnih strani že v osnovi odpravlja potrebo po nameščanju aplikacij na lokalni računalnik, omogoča delo iz katerekoli lokacije in omogoča spreminjanje izvorne kode programa na enem samem mestu.

Spletno stran smo zasnovali na spletnem strežniku Web2py. Ker ne uporabljamo podatkovne baze, saj bi s tem rešitev zakomplicirali, delamo kar z besedilnimi datotekami, ki smo jih pridobili iz podatkovnega skladišča v fazi priprave podatkov.

Aplikacijo smo izdelali kot nadgradnjo osnovnega ogrodja, ki ga ustvarimo v administrativnem vmesniku. S tem dobimo prazno aplikacijo, ki vsebuje vse datoteke, potrebne za delovanje, in osnovni, precej preprosti oblikovni slog. V to ogrodje smo dodali gumbe in spletne obrazce, ki jih potrebujemo za delovanje aplikacije.

Večina kode, ki jo bomo potrebovali za ustvarjanje modela in klasifikacijo podatkov, se nahaja v kontrolni datoteki default.py, nekaj kode je v datoteki model.py, medtem ko se koda za proženje procedur in prikaz rezultatov nahaja v datoteki index.html.

Vsebina datoteke index.html, ki je potrebna za prikaz podatkov in interakcijo z uporabnikom:

```
{{extend 'layout.html'}}

<form action="" enctype="multipart/form-data" method="post">
 <input name="_formname" type="hidden" value="form">
 <h4>Naloži datoteko za učenje modela:</h4>
 <input class="upload" name="file1" size="60" type="file">
 <h4>Naloži datoteko za klasifikacijo:</h4>
 <input class="upload1" name="file2" size="60" type="file">
 <input type="submit" value="Naloži">
</form>
<br/>
<h3>
{{=DIV(A('Zaženi
analizo',callback=URL('racunaj'),target="me"),_id="me1")}}
</h3>
<br/>
<h3>
<a href="/churn/static/rezultati.tab" target="_blank">Klikni tukaj za
rezultate</a>
</h3>
<br/><br/>
```

Vsebina datoteke default.py, v kateri je glavni del kode, ki opravi klasifikacijo:

```
import Orange
import random
import os
import shutil
import sys

def index():
 save_path = os.getcwd() + "\\applications\\churn\\static"
 form = FORM(
 INPUT(_name='file1',_type='file'),
 INPUT(_name='file2',_type='file')
 )
 if form.accepts(request.vars, formname='form'):
 print "OK!"
 stream1 = request.vars.file1.file
 stream2 = request.vars.file2.file
 name_of_file1 = 'podatki1.tab'
 name_of_file2 = 'podatki2.tab'
 complete_name1 = os.path.join(save_path, name_of_file1)
 complete_name2 = os.path.join(save_path, name_of_file2)
 shutil.copyfileobj(stream1, open(complete_name1, 'wb'))
 shutil.copyfileobj(stream2, open(complete_name2, 'wb'))
 else:
 print "Napaka!"
 return dict(form=form)

def racunaj():
 file_path = os.getcwd() + "\\applications\\churn\\static"
 data = Orange.data.Table(file_path + "\\podatki1.tab")
 data_to_classify = Orange.data.Table(file_path + "\\podatki2.tab")
 test = Orange.data.Table(random.sample(data, 50))
 train = Orange.data.Table([d for d in data if d not in test])
 logreg = Orange.classification.logreg.LogRegLearner(train,
remove_singular = 1, stepwise_lr = 0)
 target = 1
```

```
 return_type = Orange.classification.Classifier.GetProbabilities

 name = "ID"
 x=0
 text=""
 for e in data_to_classify:
 text+= str(data_to_classify[x][name]) + "\t" + str(logreg(e)) + "\n"
 x+=1

 shrani_rezultate(text, 'rezultati.tab')
 shrani_rezultate(dump, 'regkoef.txt')
 return text


def rezultati():
 myurl = URL('static', 'rezultati.tab')
 mylink = XML(A('Klikni tukaj za rezultate',_href=myurl))
 print mylink
 return mylink
```

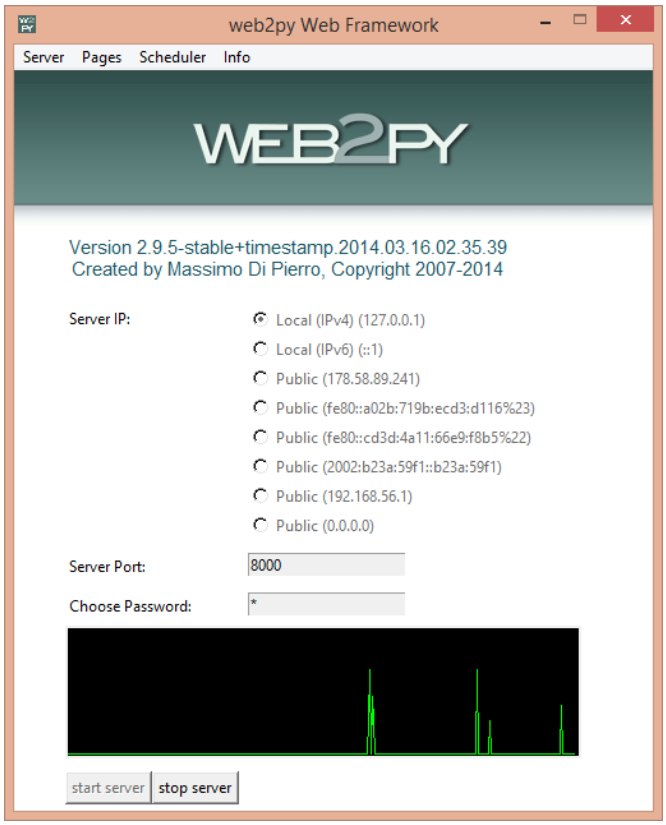
## 6.6 Končni izgled aplikacije

Aplikacija, ki smo jo izdelali, deluje po načelu odjemalec-strežnik. Ker je odjemalec katerikoli sodobni spletni brskalnik, ki podpira JavaScript, se bomo v osredotočili samo na delovanje strežnika.

Strežnik zaženemo iz ukazne vrstice v Oknih:

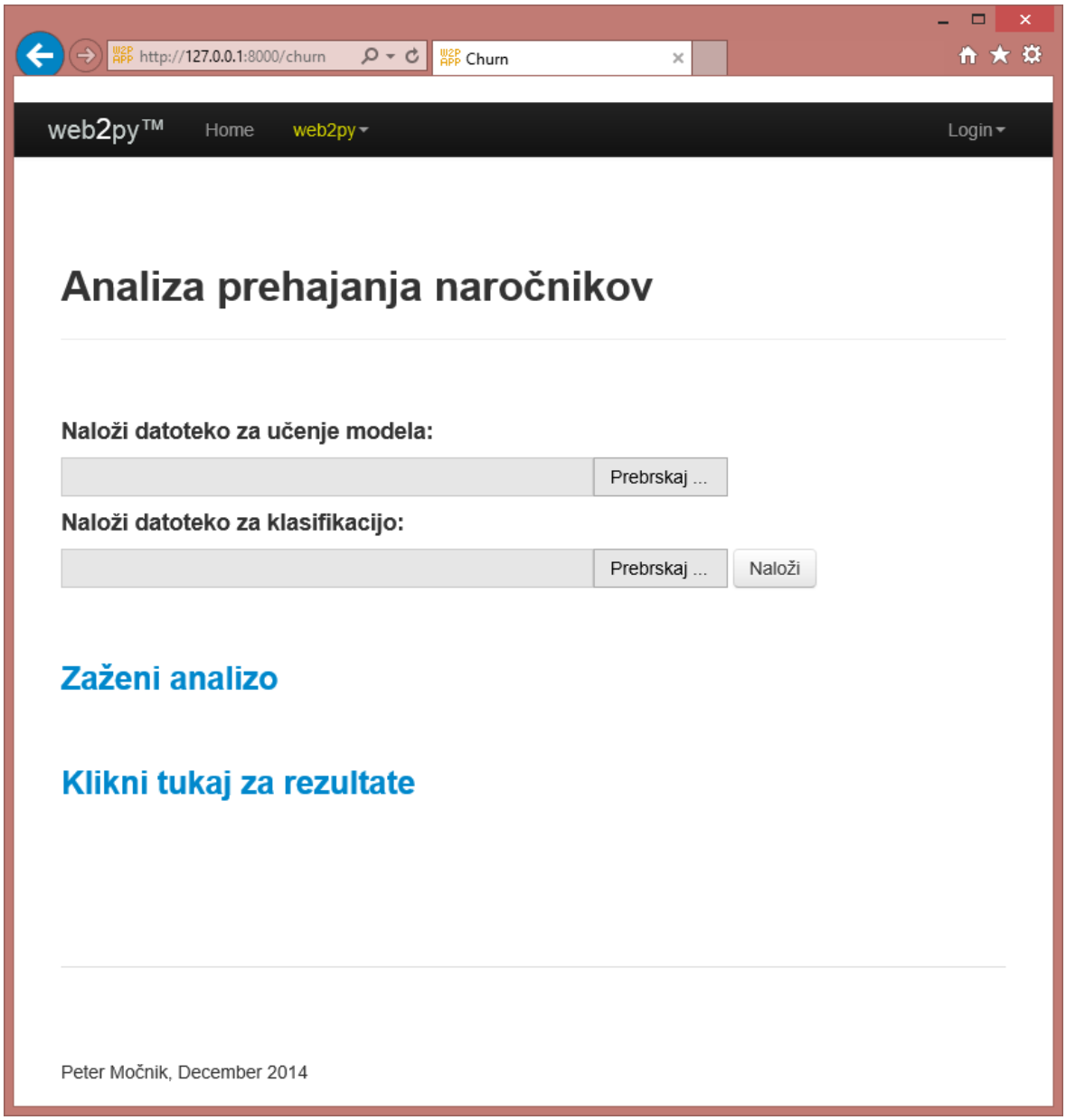
```
CD c:\Program Files (x86)\Python27\web2py\ <Enter>
Python.exe web2py.py <Enter>
```

*Slika 20: Okno z nastavitvami spletnega strežnika*




Na liki 20 smo prikazali nastavitveno okno spletnega strežnika. Ob kliku na ukaz zaženi strežnik (angl. *start server*) se odpre privzeta spletna stran, zato v brskalnik vnesemo spletni naslov naše aplikacije: http://127.0.0.1:8000/churn.

*Slika 21: Spletna stran za uporabo celotne rešitve*




Na sliki 21 smo prikazali končni izgled spletne aplikacije. V prvem okencu izberemo datoteko, v kateri se nahajajo podatki z znanim razredom, v drugem okencu pa podatke, ki jih želimo klasificirati in zanje razreda ne poznamo. Ko so podatki naloženi na strežnik lahko poženemo analizo s klikom na ustrezno povezavo. To požene prej opisano proceduro, ki klasificira podatke in rezultate shrani na strežnik. Zaradi enostavnosti prikaza delovanja smo rezultate napovedi shranili v besedilno datoteko, ločeno s tabulatorskimi znaki, ki je primerna za uporabo v večini programov za obdelavo tabel.

S tem je opravilo, ki je cilj te naloge končano, podatki o klasifikaciji pa so od tu naprej na voljo za nadaljnje korake, ki že sodijo v področje prodaje in trženja. Iz te datoteke lahko zaposleni v trženjskem oddelku izberejo tiste stranke, za katere

smo jim z klasifikacijo določili, da za njih obstaja možnost, da bodo prešli h konkurenci, to so stranke označene z številko 1 in na njih izvedejo željene akcije.

# 7 ANALIZA USTREZNOSTI REŠITVE

Izdelali smo odprtokodno rešitev, s pomočjo katere bomo lahko napovedovali prehajanje strank h konkurenci in bi lahko bila uporabna v realnem podjetju.

Ker je v podjetjih običajno, da so podatki o strankah v domeni nekih drugih in ne analitičnih oddelkov, se nismo ukvarjali s pripravo podatkov, ampak smo za analizo oz. napovedovanje pričakovali že pripravljene podatke iz podatkovnega skladišča. Podobno pripravljene podatke bi potrebovali, tudi če bi uporabili katerega od velikih lastniških sistemov.

Osredotočili smo se na izbor primerne metode za napovedovanje oz. klasifikacijo in na izbor odprtokodne programske opreme, s katero je vse skupaj mogoče izvesti. Pokazalo se je, da so odprtokodne rešitve dosegle že precejšnjo stopnjo zrelosti, saj so tudi uporabljene rešitve na trgu že več kot deset let.

Za uporabo paketa Orange smo se odločili zaradi tega, ker vsebuje zelo veliko statističnih in napovednih funkcij, ker je odprtokoden, med drugim pa tudi za to, ker je plod raziskovalnega dela Fakultete za računalništvo in informatiko Univerze v Ljubljani. Paket Orange kot aplikacija podpira delo s pomočjo grafičnega uporabniškega vmesnika. Ker pa ta po našem testiranju v 32 bitni različici na operacijskem sistemu Windows kmalu neha delovati zaradi pomanjkanja pomnilnika, je bilo potrebno uporabiti 64 bitne knjižnice, ki omogočajo uporabo več kot 4 GB pomnilnika. Te, 64 bitne knjižnice, ki smo jih uporabili v pisanju programske kode oz. v programskih skriptah, so nam omogočile izdelavo nove aplikacije z lastnim uporabniškim vmesnikom, ki je prilagojen reševanju konkretnega problema in omogoča delo z večjimi količinami pomnilnika. Grafični uporabniški vmesnik smo v nalogi uporabili le kot orodje za primerjavo kvalitete napovednih modelov, v aplikaciji, ki smo jo izdelali pa ga ne uporabljamo.

Namesto paketa Orange, bi za dosego podobne funkcionalnosti, lahko uporabili tudi kateri drugi odprtokodni program, npr. program Weka, vendar bi zaradi tega morali zamenjati celotno tehnologijo, ker je Weka napisana v programskem jeziku Java. Ugotovili smo tudi, da je Weka že nekoliko bolj uporabljan program, mi pa smo želeli raziskati možnosti uporabe paketa Orange.

Čeprav smo v tej nalogi prikazali samo uporabo logistične regresije, kot metode za napovedovanje, bi z relativno malo dela lahko to metodo zamenjali za odločitvena

drevesa, metodo naivnega Bayesa, nevronske mreže ali katero drugo, saj paket Orange vsebuje implementacije vseh teh metod.

Učni model je smiselno s časom posodabljati, saj se s časom spreminja tudi obnašanje uporabnikov. Zato programska rešitev omogoča uporabo vedno svežih podatkov za učenje modela. Po našem mnenju bi bilo za učenje smiselno jemati vzorec podatkov star med enim in tremi meseci.

Programski jezik Python smo izbrali zato, ker menimo, da je to aplikacijski jezik prihodnosti, saj je visokonivojski programski jezik in ker je njegova programska koda zelo berljiva. Količina potrebnega znanja za izdelavo takšne aplikacije v vsakem primeru zahteva znanje programiranja v jeziku Python, vendar je na spletu precej primerov uporabe in relativno dobra dokumentacija, ki pa sicer zahteva še dovolj dobro poznavanje stojnega učenja in statistike. Prednost so tudi strokovni forumi na spletu, kjer je bilo mogoče pridobiti veliko informacij in tudi pomoči sodelujočih.

Na spletu je na voljo zelo veliko dokumentacije in praktičnih primerov uporabe za spletni strežnik Web2Py. Pri paketu Orange je dokumentacije nekoliko manj, želeli pa bi si tudi več primerov uporabe. Osnovne metode so sicer dovolj dobro dokumentirane. V veliko pomoč so bili tudi avtorji in sodelujoči pri projektu Orange s sodelovanjem na spletnih forumih.

Uporaba spletnega strežnika se je zdela edina primerna rešitev, saj tako odpade potreba po distribuciji aplikacije uporabnikom in omogoča popravke in razširitve na enem mestu. Ker verjamemo, da bo v prihodnosti večina aplikacij napisanih v obliki spletnih aplikacij, smo se tudi mi odločili za ta pristop. Varnostni mehanizmi so bili v tej nalogi izpuščeni, čeprav uporabljeno spletno ogrodje z vgrajenimi metodami to omogoča.

Odločitev, da smo za operacijski sistem uporabili Microsoftova Okna temelji na tem, da je to med uporabniki še vedno najbolj razširjen operacijski sistem. Programsko rešitev bi lahko naredili tudi na operacijskem sistemu Linux ali OS X, saj sta tako Orange, kot Web2Py večinoma napisana v jeziku Python in zaradi tega brez težav delujeta tudi na drugih operacijskih sistemih.

Največja velikost datoteke, ki smo jo lahko obdelali s tem programom, se je gibala stolpcev. Analizo smo izvedli na prenosnem računalniku s 16 gigabajti delovnega pomnilnika in štiri jedrnim procesorjem Intel i7, kar je, za leto 2014, srednje zmogljiv računalnik. To pomeni, da lahko na za leto 2014 povprečnem računalniku naredimo klasifikacijo, primerno za število strank na telekomunikacijskem trgu v Sloveniji, če stranke predhodno segmentiramo.

V primerjavi z lastniškimi programskimi paketi za podatkovno rudarjenje imajo odprtokodni programi nekatere prednosti in tudi nekatere slabosti. Ena izmed prednosti je gotovo nizka cena, saj so odprtokodne rešitve večinoma brezplačne, če govorimo o nabavni ceni, nikakor pa niso več brezplačne, ko govorimo o implementaciji. Pri slednji je potrebno upoštevati še stroške dela, ki jih predstavlja izdelava celotne rešitve in prilagoditev podjetju ter stroški, ki bi jih podjetje imelo s pripravo podatkov, cena strojne in druge nujno potrebne programske opreme.

Profesionalna lastniška programska oprema za podatkovno rudarjenje običajno ponuja zelo širok nabor orodij, med katerimi nekatera morda ne bodo nikoli uporabljena, ker pa so vključena v paket, se to odraža tudi v ceni.

Prednost odprtokodnih programov je tudi zmožnost vpogleda v programsko kodo in možnosti spreminjanja le te. Pomembna je tudi prilagodljivost, saj smo s tem, da odprtokodna programska oprema omogoča vpogled v kodo in njeno uporabo na poljuben način, dobili možnost, da uporabimo samo tiste dele kode, ki jih potrebujemo za naš specifičen problem, vse ostalo pa zanemarimo.

Velika prednost celotne rešitve je v tem, da je bilo potrebno napisati relativno malo kode, kar pa je odlika vseh treh sodelujočih sistemov, torej jezika Python, knjižnic Orange in spletnega ogrodja Web2Py.

Glavna pomanjkljivost odprtokodne programske opreme je pomanjkanje strukturirane podpore in dokumentacije, kar se pri lastniški programski opremi odraža v precej visokih cenah.

Ena od možnosti razvoja takšne rešitve, ki je do zdaj nismo omenjali, bi bilo tudi naročilo razvoja odprtokodne rešitve pri zunanjih izvajalcih (angl. *outsourcing*), vendar bi s tem le delno zmanjšali količino dela v podjetju, saj bi nam v podjetju še vedno ostala priprava podatkov, priprava infrastrukture, implementacija v delovno okolje ter izobraževanje in podpora.

Naloga ni bila zastavljena kot izdelava rešitve za profesionalno uporabo v podjetjih, kaže pa na možnost razvoja aplikacij v prihodnosti.

Z uporabniškega vidika smo dosegli visoko uporabnost programske rešitve, ker je namenjena reševanju točno določenega problema in jo lahko uporabljajo uporabniki brez potrebe po intenzivnem izobraževanju. Napoved prehajanja naročnikov k drugemu operaterju bi lahko uporabniki naredili tudi v aplikaciji Orange, ki že ima vgrajen grafični uporabniški vmesnik, vendar bi za isto opravilo

potrebovali bistveno več znanja. Na operacijskem sistemu Windows, pa bi bili omejeni še z bistveno manjšo možno količino obdelovanih podatkov.
Ker smo uporabili 64 bitne knjižnice paketa Orange, smo s tem dosegli možnost obdelave bistveno večjih količin podatkov, kot pa v osnovni aplikaciji orange, ki uporablja 32 bitne knjižnice.

Za cilj smo si zastavili reševanje problema prehajanja strank h konkurenci, z narejeno aplikacijo pa smo rešili ta problem, saj si lahko z njeno uporabo pomagamo pri iskanju prehajanja strank. Aplikacija bi bila primerna za profesionalno uporabo, vendar pa bi jo bilo v tem primeru potrebno razširiti še z dodatnimi nastavitvami. Za njeno uporabo ne potrebujemo posebnega predznanja in bi bila zato primerna za uporabo v trženjskih oddelkih brez potrebe po podatkovnih analitikih.

Celotno rešitev je možno napisati v enem mesecu, dodajanje funkcionalnosti pa bi bilo od tu naprej hitrejše, saj je osnovna struktura že postavljena. Aplikacija deluje na vsaj treh najpogosteje uporabljanih operacijskih sistemih in preko spleta, hkrati pa jo lahko uporablja tudi več uporabnikov.

Menimo, da bi za resno uporabo programsko rešitev še razširiti. Dodati bi bilo potrebno možnosti za nastavitve delovanja modela, grafični prikaz uspešnosti napovedi, kot je npr. ROC krivulja ali krivulja dviga, dodati varnostne mehanizme in prikaz vhodnih podatkov. Aplikacijo bi lahko razširili še z uporabo dodatnih napovednih modelov, ki smo jih opisali v nalogi.

V nalogi smo govorili samo o uporabi aplikacije v telekomunikacijskem podjetju, lahko pa bi jo uporabili tudi drugje, kjer bi želeli klasificirati podatke z logistično regresijo in bi kot rezultat pričakovali dihotomno spremenljivko.

## SKLEP

Količina podatkov, ki se shranjuje v digitalni obliki eksponentno narašča. Da lahko iz te velike količine podatkov izluščimo neko znanje, ki nam bo pomagalo pri poslovnih odločitvah, potrebujemo računalnike in programsko opremo s katero se to lahko naredi. V tej nalogi smo se osredotočili na poslovni problem, ki ga lahko uvrstimo v področje poslovne inteligence.

Želeli smo rešiti problematike prehajanja strank h konkurenci pri operaterjih mobilne telefonije. Do prehajanja strank h konkurenci pride zaradi raznolikosti ponudbe in osveščenosti strank. Ker zaradi zasičenosti trga nadaljnja rast števila naročnikov mobilnih storitev ni več mogoča, se ta osredotočajo na preprečevanje

tega pojava. Da bi lahko poiskali stranke, za katere obstaja velika verjetnost prehoda h konkurenci, se poslužujejo metod statistike in stojnega učenja.

Na trgu obstaja mnogo ponudnikov ustreznih rešitev, ki pa zaradi svoje kompleksnosti predstavljajo precejšnje investicijske vložke. Da bi raziskali alternativo profesionalnim rešitvam, smo izdelali lastno rešitev, ki temelji na odprtokodnih orodjih.

Ker je odprtokodni sistem Orange plod dela raziskovalcev na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, smo uporabili knjižnice tega sistema. Te knjižnice smo lahko uporabili v programski kodi. Za izdelavo uporabniku prijazne rešitve, smo potrebovali še grafični vmesnik. Odločili smo se za spletno ogrodje Web2Py. Tako smo dobili robustno aplikacijo, ki je enostavna za uporabo in rešuje problem iskanja morebitnih prehodov strank h konkurenci.

Z izdelavo rešitve smo dokazali, da je odprtokodna rešitev mogoča, vendar še vedno ne čisto enostavna. Da lahko takšno rešitev izdelamo, potrebujemo znanje iz področja trženja za identifikacijo problema, znanje iz programiranja za izdelavo programske rešitve ter poznavanje mnogih sistemov in principov sodobnega razvoja aplikacij.

Načeloma bi vse korake lahko izvajali v ločenih programskih paketih, vendar potem ne bi dosegli enostavnosti uporabe. Zato smo vse korake združili v eno aplikacijo. Uporabna aplikacija zagotavljati dobro uporabniško izkušnjo. Bistvo izdelane rešitve je v tem, da ne zahteva poglobljenega znanja od uporabnikov ter omogoča enostavno uporabo sicer relativno zahtevnih postopkov. Enostavna je tudi zato, ker je strogo namenska in ne omogoča drugačne uporabe. Z razvito aplikacijo je mogoče opravljati resno naloge na področju trženja.

Z izdelavo samostojne programske rešitev za analizo verjetnosti prehoda strank h konkurenci na osnovi odprtokodnih sistemov smo želeli predstaviti enega od možnih načinov uporabe aplikacije. Prikaz delovanja s podatki o naročnikih na mobilno telefonijo nam pove, da aplikacija zmore obdelovati velike količine podatkov.

Bistvo aplikacije je izdelava statističnega modela, ki nam, po predhodnem učenju, klasificira stranke po verjetnosti s katero bodo menjale operaterja. Metoda, ki smo jo uporabili se imenuje logistična regresija in je ena izmed najpogostejših metod, ki se uporablja za reševanje tovrstnih problemov.

Ta rešitev, kot prototip, omogoča reševanje enega samega problema. Relativno preprosto bi lahko v uporabo dodali več metod za klasifikacijo. S tem bi postala

uporabna še za mnoga področja, kjer se uporablja podatkovno rudarjenje, saj uporabniku omogoča enostavno uporabo ter hkrati v ozadju uporablja kompleksne metode statistike in stojnega učenja.

Prednost odprtokodne rešitve je prav v možnosti vpogleda v jedro programske kode posameznih metod, saj programska koda ni skrita. Še več, omogoča nam tudi spreminjanje že napisane kode, da lahko posamezne dele popolnoma prilagodimo lastnim, specifičnim zahtevam.

Odločitev, da uporabimo odprtokodne sisteme, se je izkazala za dobro, sploh ker so odprti in brezplačni. Vendar pa s seboj nosijo tudi nekatere pomanjkljivosti. Kot največjo pomanjkljivost ocenjujemo količino potrebnega znanja in izkušenj, tako iz področja trženja, kot tudi analitike in programiranja, ki je potrebno za dosego cilja. Težava je tudi pomanjkanje literature in podpore ter poslovnih referenc. Pokazalo se je, da je pri razvoju takšne rešitve potrebnega precej lasnega znanja, da pa je rešitev vsekakor mogoče doseči v doglednem času.

Ker pa se odprtokodna programska oprema zelo hitro razvija in pokriva praktično vsa področja, ki jih sicer pokriva lastniška programska oprema, vidimo v tej smeri še ogromno prostora za razvoj podobnih aplikacij. Prednost je tudi v tem, da jedro sistema Orange ostaja razvoj analitičnih orodij in funkcij in se ne osredotoča na orodja za poročanje, spletni stražnik Web2Py pa omogoča ravno ta del. Zato smo lahko združili najboljši funkciji obeh sistemov v preprosto in praktično rešitev, ki je na trgu nismo našli.

V prihodnosti bo prišlo do razvoja mnogih podobnih aplikacij. Že sedaj lahko vidimo podobne integracije lastniških in odprtokodnih sistemov. Na ta račun se bo zmanjšala vloga velikih proizvajalcev programske opreme, povečala pa se bo vloga integratorjev znanja.

Tudi podatkovna analitika vse bolj pridobiva na pomenu, pa naj gre za iskanje poslovnih učinkov ali za bazične raziskave. Zato se kaže v razvoju analitičnih orodij in rešitev še ogromen razvojni potencial.

Rešitev, opisana v tej nalogi, je prav tako odprtokodnega tipa, saj to sledi že iz pogojev uporabe licenc odprtokodne programske opreme in s tem smo tudi mi dodali majhen prispevek k skupnosti in k nadaljnjemu razvoju podobnih rešitev.

## LITERATURA IN VIRI


1. Adams, A. A. (2014). Research On Open Innovation - A collection of papers on Open Innovation from leading researchers in the field. V *The open Versus Closed Debate* (str. 161 - 198). Brussels: OpenForum Europe LTD.
2. Agencija za komunikaciska omrežja in storitve Republike Slovenije. (2014). *Poročilo o razvoju trga elektronskih komunikacij za prvo četrtletje 2014.* Ljubljana: Agencija za komunikacijska omrežja in storitve Republike Slovenije.
3. Bhargava, N., Aziz, A., & Arya, R. (2013). Selection Criteria for Data Mining Software: A Study. *IJCSI International Journal of Computer Science Issues*, *10*, 308 - 312.
4. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Cambridge: Springer.
5. Bradicich, D. T. (15. januar 2013). The Moore's Law of Big Data. *National Instruments*. Najdeno 11. novembra 2014 na spletnem naslovu http://www.ni.com/newsletter/51649/en/
6. Burgelman, L. (8. februar 2013). Attention, Big Data Enthusiasts: Here's What You Shouldn't Ignore. *Wired Inovation Insights*. Najdeno 10. decembra 2014 na spletnem naslovu http://insights.wired.com/profiles/blogs/attention-big-data-enthusiasts-here-s-what-you-shouldn-t-ignore#axzz2YA8N0ne7
7. Business Intelligence. (b.l.). V *Market Intelligence & Marketing Glossary*. Najdeno 12. decembra 2014 na sletnem naslovu http://www.markintell.com/market-intelligence-glossary-a
8. Business Intelligence (BI). (2013). V *Gartner IT Glosary*. Najdeno 10. decembra 2014 na spletnem naslovu http://www.gartner.com/it-glossary/business-intelligence-bi/
9. Business Intelligence Tools. (b.l.). V *Market Intelligence & Marketing Glossary*. Najdeno 12. decembra 2014 na sletnem naslovu http://www.markintell.com/market-intelligence-glossary-a
10. Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., Piatetsky Shapiro, G., Wang, W., Bishop, C. M. (5. avgust 2004). *Data Mining Curriculum: A Proposal (Version 0.91).* Seattle: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
11. Clemente, M., Ginger Bosch, V., & San Matias, S. (b.l.). *Assessing classification methods for churn prediction by composite indicators*. Valencia:

Deparment of Applied Statistics, Operational Research and Quality, Universitat Politecnica de Valencia.

12. Center odprte kode Slovenije. (17. noveber 2011). Kaj je odprtokodna programska oprema? Najdeno 10. oktobra 2014 na spletnem naslovu http://www.coks.si/index.php5/Vse_o_Odprti_kodi

13. Demšar, j., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M. Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, L., Žitnik, M., Zupan, B. (2013). Orange: Data Mining Toolbox in Python. *Journal of machine learning Research, 14*, 2349-2353.

14. Domingos, P. (2012). *A Few Useful Things to Know about Machine Learning.* Seattle: Department of Computer Science and Engineering, University of Washington.

15. Evans, M., Hastings, N., & Peacock, B. (2000). *Statistical Distributions.* New York: Wiley.

16. Friedrich, J. (2014). Research On Open Innovation - A collection of papers on Open Innovation from leading researchers in the field. V *Facilitating Innovation: The Role of Standards and Openness in the Broader Innovation Ecosystem* (str. 31 - 55). Brussels: OpenForum Europe LTD.

17. Frost & Sullivan. (2010). *Meeting the Challenges of Data Retention: Now and in the Future.* London: Frost & Sullivan.

18. Gantz, J., & Reinsel, D. (2011). *Extracting Value from Chaos.* Framingham: IDC.

19. Greenberg, P. (2010). *CRM at the Speed of Light.* New York: McGraw-Hill.

20. Gürsoy, U. T. (2010). Customer churn analysis in telecommunication sector. *Istanbul University Journal of the School of Business Administration*, *39*(1), 35 - 49.

21. Herschel, G., Linden, A., & Kart, L. (2014). *Magic Quadrant for Advanced Anaytics Platforms.* Stamford: Gartner.

22. Hosmer Jr., D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied Logistic Regression* (3th ed.). Hoboken: John Wiley & Sons.

23. Hosmer, D., Hosmer, T., Le Cessie, S., & Lemeshow, S. (1997). A Comparison of Goodnes-of-fit Tests for the Logistic Regression Model. *Statistics in medicine*, *16*, 965 - 980.

24. Ishaya, T., & Folarin, M. (2012). Business Intelligence in Telecoms Industry: A Service Oriented Approach. V D. Catalan Matamoros (ur.), *Customer Relationship Management* (str. 125 - 146). Shanghai: InTech.

25. Jeremy. (2014). New Gartner Magic Quadrant: Advanced Analytics Platforms. *Applied Data Labs*. Najdeno 22. novembra 2014 na spletnem naslovu http://direct.applieddatalabs.com/content/new-gartner-magic-quadrant-advanced-analytics-platforms

26. Johnson, J. (16. april 2008). Free Open Source Software Is Costing Vendors $60 Billion, New Standish Group International Study Finds. *Standish Group International*. Najdeno 20. novembra 2014 na spletnem mestu http://www.marketwired.com/press-release/free-open-source-software-is-costing-vendors-60-billion-new-standish-group-international-844462.htm

27. Jović, A., Brkić, K., & Bogunović, N. (2014). *An overview of free software tools for general data mining.* Zagreb: Faculty of Electrical Engineering and Computing, University of Zagreb / Department of Electronics, Microelectronics, Computer and Intelligent Systems.

28. Kononenko, I., & Robnik Šikonja, M. (2010). *Inteligentni sistemi.* Ljubljana: Fakulteta za računalništvo in informatiko.

29. Kotler, P. (1994). *Marketing Management: Analysis, Planning, Implementation, and Control.* New Jersey: Prentice-Hal.

30. Kraljević, G., & Gotavec, S. (2010). Modeling Data Mining Applications for Prediction of Prepaid Churn in Telecommunication Services. *Automatika, 51*, 275 - 283.

31. Laney, D. (2001). *Application Delivery Strategies.* Stamford: Meta group.

32. Lerner, J., & Tirole, J. (2002). Some Simple Economics of Open Source. *The Journal of Industrial Economics, 50*(2), 197 - 234.

33. Massimo Di, P. (2011). *Web2py, Full-stack Web Framework* (4th ed). Chicago: Experts4solutions.

34. Mattison, R. (2005). *The Telco Churn Management Handbook.* Oakwood Hills: XiT Press.

35. Memory Limits for Windows and Windows Server Releases. (b.l.). Microsoft Developer Network. Najdeno 22. novebra 2014 na spletnem mestu http://msdn.microsoft.com/en-us/library/windows/desktop/aa366778%28v=vs.85%29.aspx

36. Moore, G. A. (2001). *Crossing the chasm.* New York: HarperCollins Publishers.

37. *MVC*. (27. januar 2014). Najdeno 21. novembra 2014 na spletnem mestu http://books.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC

38. Olson, D. L., & Delen, D. (2008). *Advanced Data Mining Techniques.* Berlin Heidelberg: Springer.

39. Oracle. (b.l.). *Overview of Extraction, Transformation, and Loading.* Najdeno 10. oktobra 2014 na spletnem naslovu https://docs.oracle.com/cd/B19306_01/server.102/b14223/ettover.htm

40. *Orange*. (b.l.). Najdeno 10. oktobra 2014 na spletnem mestu http://orange.biolab.si/

41. Orozco Zuluaga, A. (6. oktober 2011). Data Mining, a useful tool in Business Intelligence. *BI, BizTalk*. Najdeno 15. decembra 2014 na spletnem naslovu http://blogs.itsynergy.co/aorozcoz/2011/10/06/data-mining-algorithms/

42. Pareek, D. (2007). *Bussiness Intelligence for Telecommunications.* New York: Auerbach Publications.

43. Payne, A., & Frow, P. (2005). A Strategic Framework for Customer Relationship Management. *American Marketing Association, 69*, 167 - 176.

44. Piatetsky, G. (7. junij 2014). KDnuggets 15th Annual Analytics, Data Mining, Data Science Software Poll. *Kdnuggets*. Najdeno 22. novembra 2014 na spletnem naslovu http://www.kdnuggets.com/2014/06/kdnuggets-annual-software-poll-rapidminer-continues-lead.html

45. Press, G. (9. marec 2014). 12 Big Data Definitions: What's Yours? *Forbes*. Najdeno 10. oktobra 2014 na spletnem naslovu http://www.forbes.com/sites/gilpress/2014/09/03/12-big-data-definitions-whats-yours/

46. *Pyinstaller*. (b.l.). Najdeno 21. novembra 2014 na spletnem mestu http://www.pyinstaller.org/

47. Pyle, D. (1999). *Data preparation for Data Mining.* San Francisco: Morgan Kaufmann Publishers.

48. *Python*. (b.l.). Najdeno 10. oktobra 2014 na spletnem mestu https://www.python.org/

49. Rangra, K., & Bansal, D. (2014). Comparative Study of Data Mining Tools. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(6), 216 - 223.

50. Reicheld, F., & Sasser, W. E. (1990). Zero Defections: Quality Comes to Services. *Harvard Bussines Review*, *9*, 105 - 111.

51. Schmidt, D. (18. junij 2014). Benefiting from the Disruption in Data Mining Software*. CFO*. Najdeno 12. decembra 2014 na spletnem mestu http://ww2.cfo.com/analytics/2014/06/benefiting-disruption-data-mining-software/

52. Shalizi, C. R. (2014). *Advanced Data Analysis from an Elementary Point of View.* Cambridge: Cambridge University Press.

53. Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing, 5*(4), 13 - 22.

54. Simeunović, V., & Milosavljević, M. (2010). *logistička regresija, kao osnova mašinskog učenja.* Bijeljina: Sinergija.

55. Summerfield, M. (2007). *Rapid GUI Programming with Python and Qt.* Upper Saddle River: Prentice Hall.

56. Statistični urad Republike Slovenije. (25. september 2014). Poslovanje podjetij v telekomunikacijskem sektorju, Slovenija, 2013. Najdeno 10. novembra 2014 na spletnem naslovu http://www.stat.si/novica_prikazi.aspx?id=6520

57. Tsiptsis, K., & Chorianopoulos, A. (2009). *Data Mining Techniques in CRM: Inside Customer Segmentation.* Chippenham: John Wiley and Sons.

58. Vuk, M., & Curk, T. (2006). ROC Curve, Lift Chart and Calibration Plot. *Metodološki zvezki, 3*(1), 89 - 108.

59. Wavelet, T., Trejo, J., Griffa, E., & Vallejo, J. (2012). *Telecom business transformation.* Stockholm: Ericsson.

60. *Web2py*. (b.l.). Najdeno 10. oktobra 2014 na spletnem mestu http://www.web2py.com/

61. Weber, S. (2004). *The Success of Open Source.* Cambridge: Harvard University Press.

62. Whetten, D., Smith, A., Pearce, C., & Alexander, J. (2005). *Cisco Callmanager Fundamentals* (2nd ed.)*.* Indianapolis: Cisco Press.

63. Yang, L.-S., & Chiu, C. (b.l.). *Subscriber Churn Prediction in Telecommunications*. Tamsui: St. John's University.