TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

IAY40LT
Elar Pottmann 052360IASB

Tarkvaralise pokkerimängija loomine kasutades Weka klassifikaatoreid

bakalaureusetöö

Peeter Ellervee
Ph.D.
professor

Tallinn 2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesoleva lõputöö eesmärgiks oli koostada tarkvaraline pokkerimängija kasutades Weka tarkvarapaketis (välja töötatud Waikato Ülikoolis) sisalduvaid klassifikaatoreid.

Töös leidsid käsitlust probleemid, mis puudutasid eelnimetatud klassifikaatorite treenimiseks ja mudelite loomiseks vajatavate andmete leidmist, töötlust sobivale kujule, klassifikaatorite treenimist; aga samuti loodud pokkerimängija häälestamist ning testimist reaalset mängusituatsiooni modelleerivas testimiskeskkonnas. Ka lootis töö autor saada vastuse küsimusele, kas on võimalik kasutada ebatäielikest algandmetest tuleneva nn. „foldimisprobleemi“ lahendamiseks ühe klassi klassifikaatorit (*one class classifier*).

Vastavalt läbiviidud testidele võib õelda, et loodud pokkerimängija mudelid ületasid oma mänguoskustelt kõige lihtsamalt mängivaid vastaseid (nõustuv, tõstev, ja umbes mängiv mängija), aga jäid alla lihtsale ekspertteadmistega mängijale. Töö tulemused võtab kokku esitatav diagramm. Teiseks võib väita, et ühe klassi klassifikaatori kasutamine „foldimisprobleemi“ lahendamisel on võimalik, kuid lihtsam lahendus oleks kasutada siiski täielikku algaandmete kogumit.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 4 peatükki, 12 joonist, 2 tabelit.

Abstract

Creating poker player using Weka's classifiers

Aim of this thesis was to compose a poker player using classifiers, included in software package Weka – the datamining tool, developed by University of Waikato.

The current work studied the problems concerning finding the poker games database, usable for training used classifiers and creating models; and processing that data into a suitable form of treatment. Then the attributes for that classifiers and candidates for modelling were chosen. Also, author wanted to get answer to the question whether it is possible to use one class classifier to generate action „fold“ from poker games data set, which does not contain corresponding knowledge.

According to the tests, author can say that the created poker player models exceeded their easy coded opposing players (caller, raiser and randomly playing player) by skills of play, but were left behind the player, supplied with simple expert's knowledge. To show this, the current work presents a diagram summarizing the results of testing. Secondly, it can be argued that the use of one class classifier to solve the abovementioned „folding problem“ is of course possible, but is questionable due to the complexity. The simplest solution would be to use a full set of training data, with complete information about players actions and their pocket cards.

The thesis is in Estonian and contains 32 pages of text, 4 chapters, 12 figures, 2 tables.

Lühendite ja mõistete sõnastik

amountToCall	Atribuut: summa, mis tuleb maksta, jäämaks mängu.
betsToCall	Atribuut: mitu pakkumist on eelnevalt tehtud.
Big blind, small blind (blinds)	Pakkumisjärjekorras esimese 2 mängija poolt tehtud sundpakkumised. Kindlustab, et oleks, mille peale pakkuda.
checkRaise	Atribuut: olukord, kus mängija esiteks nõustub, seejärel tõstab pakkumist, kui pakkumisjärg temani jõuab.
coldCall	Atribuut: situatsioon, kus mängija nõustub tõstega esimesel ringil (*preflop*), kui ta ei ole veel panustanud, k.a. *blinds*.
flushPossible	Atribuut: 5 ühest mastist kaardi võimalikkus.
Fold, foldimine	Mängija loobub edasisest pakkumisest.
lastActionRaise	Atribuut: mängija viimane tegevus oli tõste.
Pot	Mängus panustatud raha kokku.
potOdds	Atribuut: potiSuurus / *amountToCall*.
Preflop, flop, turn, river	Vastavalt 1., 2., 3., ja 4. pakkumisvoor.
raisedPreflop	Atribuut: mängija tõstis pakkumist *preflop*-s.
reRaise	Atribuut: ületõste.
Showdown	Mängu lõpus mängijad avavad oma kaardid, et selgitatada välja tugevaim kaardikombinatsioon.
stack	Atribuut: mängija kasutada olev raha hulk.

steeling	Atribuut: püüe võita viletsate kaartidega viimastelt positsioonidelt, kui eelnevalt pole pakutud.
Taskukaardid	Mängijale jagatud kaardid, mida teistele ei näidata.
Atribuut	Omadus (nominaalne või numbriline), mis kirjeldab mingit situatsiooni (sisendvektor).
Klass	Ühesuguste omadustega instantside kogum.
Klassiatribuut	Omadus, mis määrab antud sisendvektori (instantsi) kuuluvuse mingisse klassi.
Klassifikaator	Algoritm, mis võimaldab koostada vastavalt treeningpaketile mudeli, ja seejärel määratleda vastava instantsi (sisendvektor) kuuluvuse mingisse klassi.
RandomForest	Klassifitseerimisalgoritm, kus mudeli saadakse mingi hulga puustruktuuride loomise kaudu.
Sisendvektor (instants)	Väärtustatud atribuutide kogum.
SVM	***Support Vector Machine*** Klassifitseerimisalgoritm, mis põhineb otsustustasandite kontseptsioonil. Otsustustasand määrab piirid klasside vahel.
Treeningpakett	Treeningvektorite kogum.
Treeningvektor	Sisendvektor koos klassiatribuudiga.
Ühe klassi klassifikaator, ÜKK	***One class classifier*** Klassifitseerimisalgoritm, mis mis jagab instantsid 2-ks kasutades ainult ühte klassi kuuluvaid treeningvektoreid.

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Käesolev bakalaureusetöö on loogiline jätk autori eelmisele projektile [1], kus kasutati närvivõrke pokkerimängija loomiseks. Tulenevalt selle töö käigus närvivõrkude treenimiseks kasutatud andmete ebatäielikkusest ei olnud võimalik luua lihtsa ekspertsüsteemiga mänguoskustelt võrreldavat mängijat.

Tõuke teemaga jätkamiseks andis autorile pähe tulnud idee kasutada [1]-st jäänud probleemide lahendamiseks nn. ühe klassi klassifikaatorit (*one class classifier*).

Tegemist on õppeotstarbelise hobiprojektiga.

Töö käsitleb Hold'em No Limit stiilis mängitud ($0.25/$0.50) 9 mängijaga mänge.

1.1 Taust ja probleem

Nagu eelnevalt on öeldud, eelmises töös ei õnnestunud ehitada pokkerimängijat, mis valdaks mängu kogu soovitud ulatuses. Probleem seisneb selles, et pokkerimängude andmed (mängud), mida on võimalik hankida (koosnedes pokkeritubade või salvestustarkvara salvestistest), ei sisalda 100%-list informatsiooni mängijate käes olevate kaartide kohta. Kaarte näidatakse kaasmängijatele harva ja peamiselt mängu lõpus, nn. *showdown*-is. See tähendab aga seda, et täielikult puudub teave selle kohta, mis taskukaartide korral mängija oma kaardid *foldib* (loobub edasisest pakkumisest). Kui aga ei ole andmeid, kuidas inimene sellises situatsioonis käitus, siis ei ole ka võimalik koostada täpset mängu kirjeldavat mudelit. Nii ka juhtus – treenitud närvivõrgud ei osanud *foldida*, kuna treeningandmetes (algandmetes) puudus vastavasisuline oskus.

Kuigi puuduv oskus anti mängijale lisafunktsiooniga, ei ole see autori arvates sobiv lahendus.

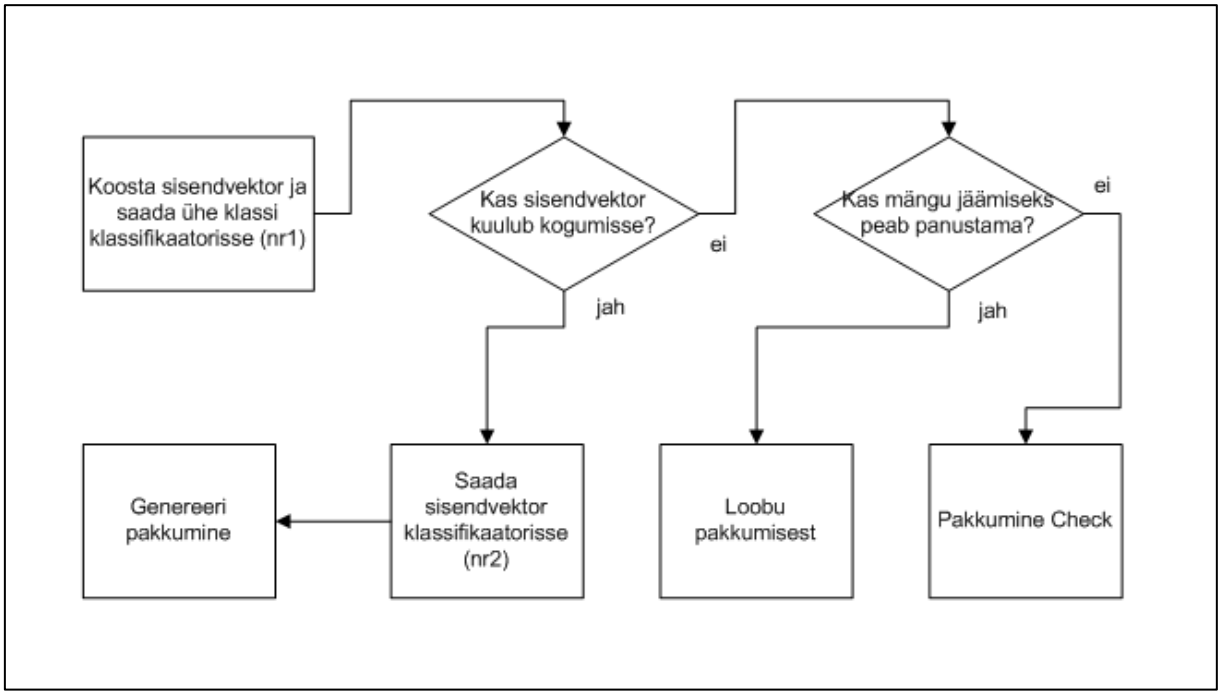
Kirjeldatud probleemi saab kahtlemata lahendada, ostes täieliku andmekogumi. Samuti võib proovida leida kasumlikult mängiv inimene ja paluda tal oma mängud vastavat tarkvara kasutades (nt. PockerTracker, HUD) salvestada. Kahjuks autoril puudub soov teha kulutusi, samuti puudub profimängijast tuttav.

Teine probleem: [1]-s oli kasutada küllalt väikesemahuline komplekt algandmeid.

1.2 Ülesande püstitus

Lähtudes eelpoolkirjeldatust, tõstatas autor ülesande:

- Leida sisendandmeteks andmekogum, mis on mahult suurem kui eelnevalt kasutatud.
- Uurida, kas on võimalik kasutada *fold* probleemi lahendamiseks nn. ühe klassi klassifikaatorit (*one class classifier*, edaspidi ka ÜKK). **Kontrollida hüpoteesi:** kui ÜKK on treenitud mingite treeningandmetega, mis antud juhul kirjeldavad pokkeri mängu ja koostatud on vastav mudel, siis juhul, kui mudelisse sisestada treeningandmetega kattuv sisendvektor, annab klassifikaator vastuseks positiivse vastuse, vastasel juhul aga negatiivse vastuse, mida sel juhul võib lugeda mängija lahkumiseks mängust ehk tegevuseks *fold*.
- Koostada pokkerimängija (Joonis 1.) kasutades lisaks ÜKK-le mingit teist valitud klassifikaatorit ja testida keskkonnas, mis vastab 9 mängijaga pokkerilauas toimuvale mängude käigule.



Joonis 1. Ülesanne.

1.3 Metoodika

Töös vajalikud klassifikaatorid on realiseeritud kasutades Weka 3.6.11. tarkvara. Kood on kirjutatud java-s kasutades Netbeans IDE 7.1(Build 201112071828). Andmete korrastamisel oli kasutusel MySQL Server 5.5. Testid on läbi viidud autori poolt koostatud simulaatoril. Arvuti: Intel(R) Core(TM) i5 CPU M 560 @ 2.67GHz, 3,42Gb RAM, operatsioonisüsteemiga Ms Windows XP Professional Version 2002.

Klassidiagrammid on vormistatud IBM Rational Rose Enterprise Edition 7.0.0.0.-ga. Kasutatud on ka MS Office Visio 2007.

Arenduse käigus kasutati Lean Startup arendusmetoodikat [2]. See tähendab, realiseerisin ideed kodeerides, seejärel katsetasin ja sain mingid tulemused, millest tekkisid analüüsides uued ideed; ja nii läbisin mitmeid tsükleid (iteratsioone), kuni jäin tulemusega rahule.

2. Pokkerimängija koostamine

Käesolev peatükk kirjeldab kahe pokkerimängija mudeli loomist. Selleks leitakse ja töödeldakse kõigepealt vajalikud algandmed, seejärel valitakse atribuudid ning töödeldud andmete põhjal modelleerimiskandidaadid, misjärel viiakse läbi klassifikaatorite valik ning koostatakse treeningpaketid valitud klassifikaatoritele.

2.1 Andmete otsimine ja valik

Lähtuvalt ülesandepüstitusest vajame sobivat algandmete komplekti; see peaks olema järgmiste omadustega:

- Mängukirjeldused on mingis kindlas formaadis – see tähendab, et neid on võimalik automaatselt analüüsida ja töödelda.
- Sisaldab mänge 9 mängijaga laudadest – vastavalt ülesandepüstitusele.
- Sisaldab mänge, kus näidatakse mängijate taskukaarte – loomaks võimaluse modelleerida mängijate käitumist.
- On järjestatud – see tähendab, et on võimalik modelleerida mängijate käitumist ajas.

Kõigepealt on vaja leida sobiv andmebaas. [3] pakub välja mõned võimalused, kuidas ja kust vajaminevaid pokkerimängude kirjelduste andmeid on võimalik tasuta alla laadida:

- FlopTurnRiver.com – autoripoolne otsing lehel ei andnud tulemusi.
- PokerFTP.com – autoripoolne otsing lehel ei andnud tulemusi.

Kuna ei õnnestunud eelmainitud andmeallikaid kasutada, siis viisin läbi otsingu Internetis, mille tulemusena õnnestus leida leht, mis pakkus tasuta valiku eeldatavalt sobivaid andmeid [4]. Sealt laadisin alla komplekti PokerStars-is mängitud Hold'em No Limit ($0.25/$0.50) mänge, mis sisaldas 1231 faili, igaühes 1000 mängu.

Analüüsides hangitud faile lähemalt, võib öelda, et need sisaldavad mänge 2-, 6- ja 9-mängijalistest pokkerilaudadest. Mängud on segatud, see tähendab ei ole võimalik järgida

mängude järgnevust, samuti sisaldub vigu (mängud, mis ei järgi mingit kindlat formaati – sobiv formaat: Joonis 2.). Siit tuleneb ülesanne järgnevaks andmete töötluseks.

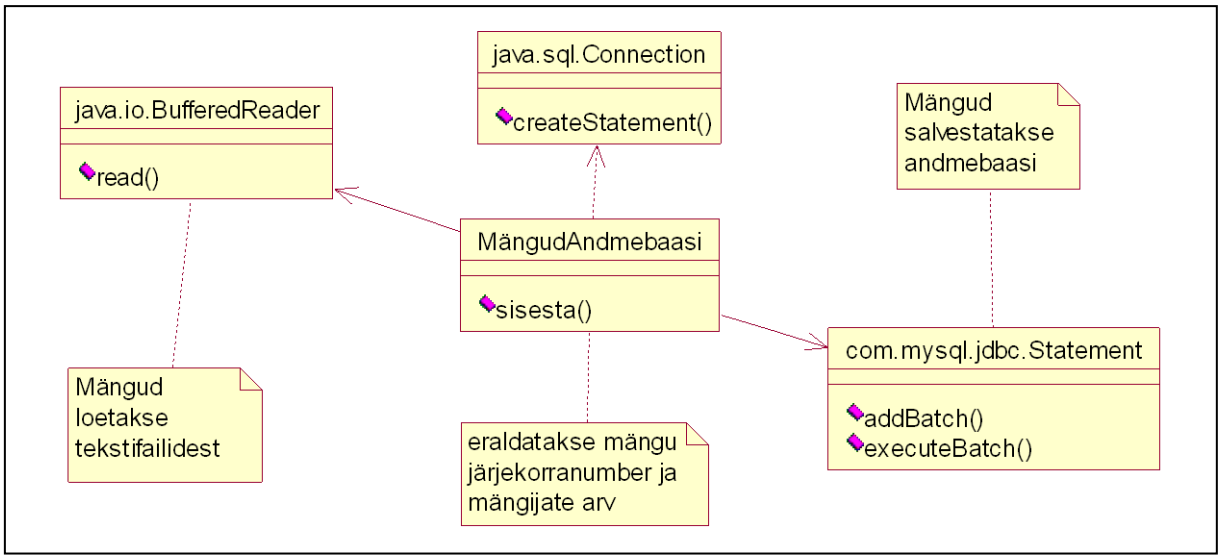
```
PokerStars Game #59966330166:  Hold'em No Limit ($0.25/$0.50) - 2009/07/01 12:45:23 ET
Table 'X6IRCRTh+20gjz8aubWnwA' 9-max Seat #6 is the button
Seat 1: Y2+HX4KI85b7Iog/Gg7n7A ($6 in chips)
Seat 2: M8ggOSsQmjuWf51T98o/RA ($50 in chips)
Seat 3: RPoA9nYcMrr9wzO3cyxb9Q ($57.10 in chips)
Seat 4: OLkOHswAWaY/G7UNAQJTQQ ($53.70 in chips)
Seat 5: n6VQBu6crjYjy6DxNPlNZA ($12.90 in chips)
Seat 6: OjX3DPxludIseYYMQOsNEw ($102.20 in chips)
Seat 7: c+cNFeDyGClqqBPQK9T2nQ ($36.40 in chips)
Seat 8: NeT/ivHFBm3Gomvc3n2Fyw ($43.40 in chips)
Seat 9: Qs8lOOwlWmK9cugIk75qbw ($10 in chips)
c+cNFeDyGClqqBPQK9T2nQ: posts small blind $0.25
NeT/ivHFBm3Gomvc3n2Fyw: posts big blind $0.50
*** HOLE CARDS ***
Qs8lOOwlWmK9cugIk75qbw: folds
Y2+HX4KI85b7Iog/Gg7n7A: raises $5.50 to $6 and is all-in
M8ggOSsQmjuWf51T98o/RA: folds
RPoA9nYcMrr9wzO3cyxb9Q: folds
OLkOHswAWaY/G7UNAQJTQQ: folds
n6VQBu6crjYjy6DxNPlNZA: folds
OjX3DPxludIseYYMQOsNEw: folds
c+cNFeDyGClqqBPQK9T2nQ: calls $5.75
NeT/ivHFBm3Gomvc3n2Fyw: folds
*** FLOP *** [5s 9c Qh]
*** TURN *** [5s 9c Qh] [9d]
*** RIVER *** [5s 9c Qh 9d] [6c]
*** SHOW DOWN ***
c+cNFeDyGClqqBPQK9T2nQ: shows [Th Ad] (a pair of Nines)
Y2+HX4KI85b7Iog/Gg7n7A: shows [4c 4h] (two pair, Nines and Fours)
Y2+HX4KI85b7Iog/Gg7n7A collected $11.90 from pot
*** SUMMARY ***
Total pot $12.50 | Rake $0.60
Board [5s 9c Qh 9d 6c]
Seat 1: Y2+HX4KI85b7Iog/Gg7n7A showed [4c 4h] and won ($11.90) with two pair, Nines and Fours
Seat 2: M8ggOSsQmjuWf51T98o/RA folded before Flop (didn't bet)
Seat 3: RPoA9nYcMrr9wzO3cyxb9Q folded before Flop (didn't bet)
Seat 4: OLkOHswAWaY/G7UNAQJTQQ folded before Flop (didn't bet)
Seat 5: n6VQBu6crjYjy6DxNPlNZA folded before Flop (didn't bet)
Seat 6: OjX3DPxludIseYYMQOsNEw (button) folded before Flop (didn't bet)
Seat 7: c+cNFeDyGClqqBPQK9T2nQ (small blind) showed [Th Ad] and lost with a pair of Nines
Seat 8: NeT/ivHFBm3Gomvc3n2Fyw (big blind) folded before Flop
Seat 9: Qs8lOOwlWmK9cugIk75qbw folded before Flop (didn't bet)
```

Joonis 2. Mängu sobiv formaat: 9 mängijat, kaartide näitamisega.

2.2 Andmete ettevalmistus

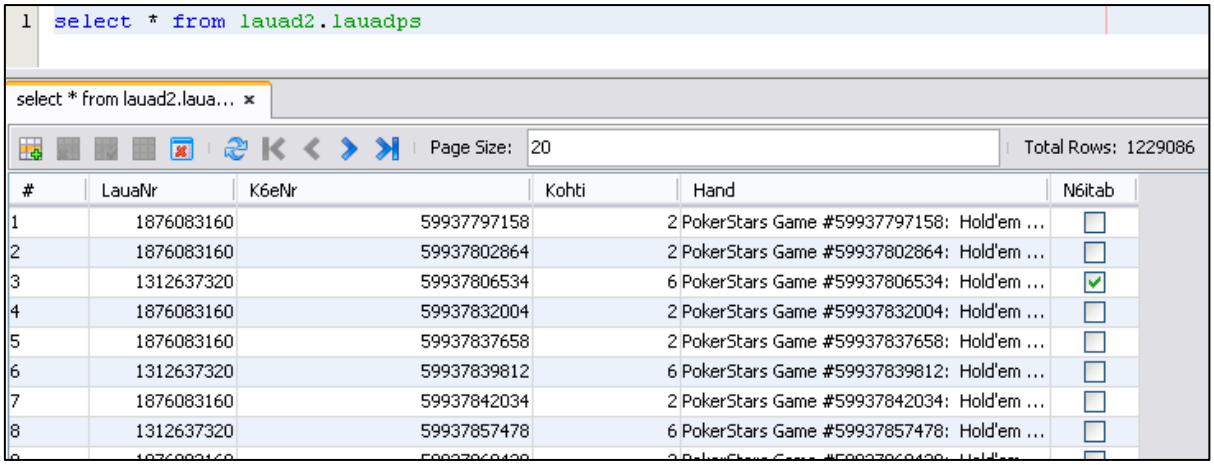
Eelmises punktis kirjeldatust lähtudes püstitan ülesande, mida kirjeldan alamülesannetena:

- Sisestada mängud andmebaasi (MySQL), võimaldamaks nende sorteerimist ja järjestamist. Selleks loetakse mängud failidest, eraldatakse sorteerimiseks vajalikud suurused mängude järjekorranumbrid ja mängijate arvud mängudes ja salvestatakse andmebaasi (Joonis 3., Joonis 4.).



Joonis 3. Salvestamine andmebaasi.

Mängud loetakse algsetest tekstifailidest töötlemiseks andmebaasi.

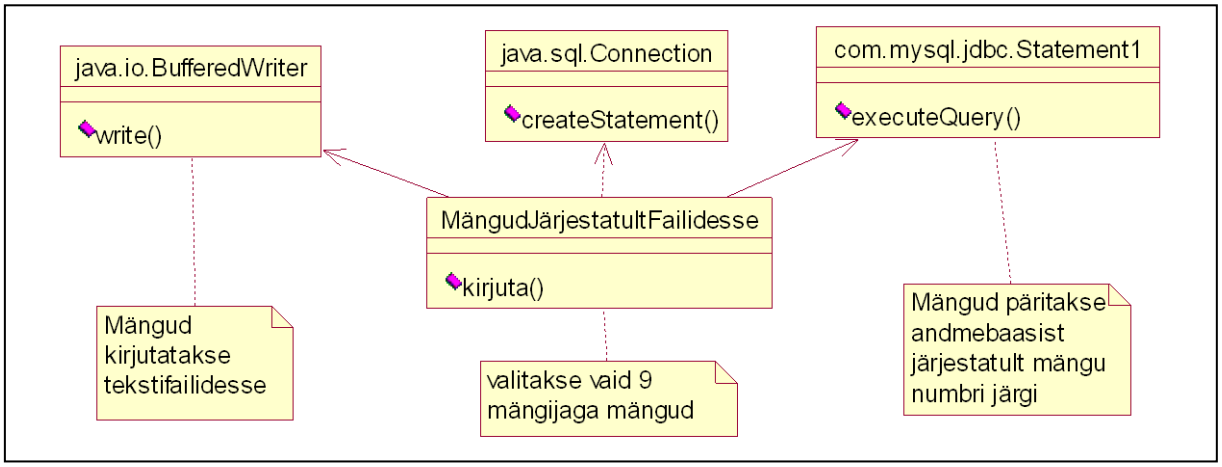


#	LauaNr	K6eNr	Kohti	Hand	N6itab
1	1876083160	59937797158	2	PokerStars Game #59937797158: Hold'em ...	☐
2	1876083160	59937802864	2	PokerStars Game #59937802864: Hold'em ...	☐
3	1312637320	59937806534	6	PokerStars Game #59937806534: Hold'em ...	☑
4	1876083160	59937832004	2	PokerStars Game #59937832004: Hold'em ...	☐
5	1876083160	59937837658	2	PokerStars Game #59937837658: Hold'em ...	☐
6	1312637320	59937839812	6	PokerStars Game #59937839812: Hold'em ...	☐
7	1876083160	59937842034	2	PokerStars Game #59937842034: Hold'em ...	☐
8	1312637320	59937857478	6	PokerStars Game #59937857478: Hold'em ...	☐

Joonis 4. Mängud andmebaasis.

Nagu näha, kokku on 1229086 mängu, millest siiski enamus on 2- ja 6-mängijalised.

- Kontrollida mängude salvestuste sobivust formaadile (Joonis 2.), võtta ainult 9 mängijaga mängud ja sorteerida need vastavalt laua järjekorranumbrile, et saada võimalus hankida teavet, mis seotud mängude järgnevusega. Järgnevus on hiljem vajalik mängude simulatsioonil. Salvestada sorteeritud ja järjestatud käed tekstiformaadis failidesse (teise võimalusena oleks võinud mängud ka jätta andmebaasi ja hiljem neid andmebaasist pärida; Joonis 5.).



Joonis 5. Valikuline mängude salvestamine failidesse.

2.3 Atribuutide valik

Selleks, et oleks võimalik treenida klassifikaatorit, vajame sisend- ja väljundatribuute. Valik peab peegeldama mängusituatsiooni ja mängija käitumist selles situatsioonis nii laiapõhjaliselt kui võimalik. Lähtuvalt autori teadmistest ja arusaamadest valitud atribuudid antud töö ülesande lahendamiseks on esitatud alljärgnevates loeteludes. Atribuutide nimetused on valitud informatiivsuse järgi ja on tõlgitult kohati väga kohmakad – vajadusel, näiteks sobiva tõlke puudumisel, on lisatud sulgudes vastava nimetuse inglisekeelne vaste.

Sisendatribuudid:

- **võiduTõenäosus** – Saadakse: võetakse antud mängija taskukaardid ja lauakaardid(kui on), omistatakse teistele mängijatele juhuslikud kaardid, lisatakse puuduvad

lauakaardid ja selgitatakse välja võitja. Korratakse mingi arv (praegu 2000) kordi. Siit arvutatakse võidu tõenäosus. See atribuut väljendab tegelikult mängija taskukaarte ja võib olla asendatud kaarte kirjeldavate atribuutidega.

- **potiSuurus** – Hetkel *pot*-is olev raha hulk.
- **panustatudRing** – Antud pakkumisringi jooksul panustatud summa.
- **viimanePakkumine** – Antud pakkumisringi käigus tehtud viimase pakkumise suurus.
- **summaNõustuda** (*amountToCall*) – Summa suurus, kui palju on vaja maksta, et jääda mängu.
- **mängijaidAlguses** – Mängijate hulk, kes alustasid mängu. Hiljem see atribuut filtreeritakse välja, et võimaldada koostatud mudelitel mängida ka mänge, kus alguses on vähem mängijaid kui 7-9.
- **mängijaid** – Antud hetkel mängus osalevate mängijate hulk.
- **positsioon** – Mängija positsioon pakkumisjärjekorras.
- **pakkumistNõustuda** (*betsToCall*) – Pakkumiste arv, mis on tehtud enne antud mängijani jõudnud pakkumisjärjekorda.
- **AKQlauas** – Kas lauakaartide hulgas on ässa, kuningat või emandat?
- **mastVõimalik** (*flushPossible*) – Kas on võimalik, et keegi mängijatest kogub viis ühest mastist kaarti?
- **lauaVõimalused** (*potOdds*) – Arvutatakse: potiSuurus/summaNõustuda.
- **panustanudRing** – Antud pakkumisringil mängija poolt panustatud summa.
- **panustanudKokku** – Mängija poolt kogu mängu jooksul panustatud summa.
- **rahaKokku** – Mängijal kasutada olev summa (*stack*).
- **esimenePakkuja** – Mängija asub esimesel positsioonil.
- **viimanePakkuja** – Mängija asub viimasel positsioonil.

- **varastamineVõimalik** (*steeling*) – Kas on nn. „varastamise“ olukord? See on mängusituatsioon, kus mängija asudes viimastel positsioonidel ja omades viletsaid taskukaarte teeb siiski pakkumise, kui kõik teised eelnevalt on loobunud, lootes võita nn. pimedalt (*small blind, big blind*) pakutud summa.

- **üleTõste** (*reRaise*) – Kas on olukord, kus on võimalik eelmise tõste ülepakkumine?

- **külmNõustumine** (*coldCall*) – Situatsioon, kus mängija nõustub tõstega esimesel ringil, kui ta ei ole veel panustanud, kaasaarvatud pimedad pakkumised.

- **nõustumineJaTõste** (*checkRaise*) – Kas on võimalik olukord, kus mängija enne nõustub; seejärel aga tõstab, juhul kui pakkumiskord uuesti temani jõuab?

- **suurVõit** – Kas mängitakse suure summa peale? Suure summa piirsuuruseks on võetud 20 suurt pimedat panust (*big blind*).

- **tõstisEsimeneRing** (*raisedPreflop*) – Kas mängija tõstis ka *preflop*-is, esimeses pakkumisringis?

- **viimaneTõste** (*lastActionRaise*) – Kas mängija viimane tegevus oli tõste?

Väljundatribuudid:

- **onKogumis** – See atribuut lisatakse igale treeningvektorile, võimaldamaks ÜKK-l klassifitseerimisülesannet lahendada.

- **pakkumineCheck** – Mängija nõustub 0 panusega.

- **pakkumineCall** – Mängija nõustub pakkumisega.

- **pakkumineRaise1** – Mängija teeb pakkumise (või tõstab pakkumist) 75% *poti* suurust.

- **pakkumineRaise2** – Mängija teeb pakkumise (või tõstab pakkumist) 75-125% *poti* suurust.

- **pakkumineRaise3** – Mängija teeb pakkumise (või tõstab pakkumist) 125-225% *poti* suurust.
- **pakkumineRaise4** – Mängija teeb pakkumise (või tõstab pakkumist) 225-425% *poti* suurust.
- **pakkumineRaise5** – Mängija teeb pakkumise (või tõstab pakkumist) 425-825% *poti* suurust.
- **pakkumineRaise6** – Mängija teeb pakkumise (või tõstab pakkumist) 825% *poti* suurust või enam.

2.4 Modelleerimiskandidaatide valik

Klassifitseerimisülesande lahendamine kujutab endast sisuliselt mudeli loomist ning seejärel kasutamist – see tähendab vastavalt koostatud mudeli sisendite väärtustele väljundväärtuste saamist. Antud töö kontekstis tähendab see seda, et võetakse üks mängija või mängijate kogum ning koostatakse andmekaevandamise algoritme kasutades (Weka) selle mängija või mängijate mudel, mida seejärel mängusituatsioonis ka kasutatakse.

Kuna soov on, et koostatud mudel testides ka võimalikult edukas oleks, siis tuleks valida võimalike modelleerimiskandidaatide seast parim (või parimad).

Esiteks vajame parima väljaselgitamiseks valikukriteeriume. Olgu need mängitud mängude arv, balanss, balanss mängu kohta, näidatud kaartide arv, näidatud kaartide arv võiduga.

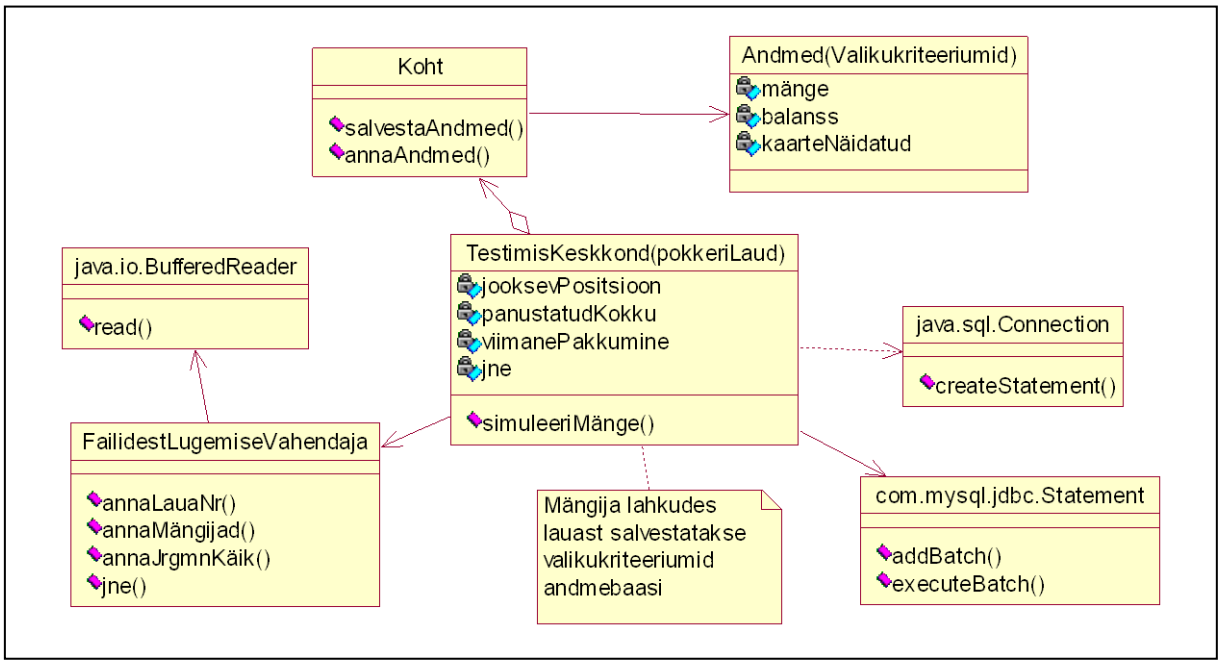
Teiseks vajame keskkonda, mis võimaldaks andmeid sorteerida.

Arvestades eelöelduga koostasin kasutades juba olemasolevat testkeskkonda (simulaatorit) algoritmi, mille tööpõhimõte on järgmine (Joonis 6.):

- Simuleeritakse reaalset pokkerimängu, valides järjest mänge vastavatest tekstifailidest.
- Kui mängija on lauas, siis kogutakse tema kohta jooksvalt teavet – mõõdetakse valikukriteeriumeid.

- Kui mängija lahkub lauast, siis salvestatakse eelmainitud andmed MySQL andmebaasi, lisades eelnevalt salvestatule.

- Mängijate sorteerimine ja valik (Joonis 7.).



Joonis 6. Mängijate salvestamine valikuks.

```
1  select * from lauad2.salvestusps where m6nge>1000 and kaarteN6idatud>40 order by balanssM6ngus desc
```

select * from lauad2.salv... ×

Page Size: 20 Total Rows: 162 Page: 1 of 9 Matching Rows:

#	M6ngija	M6nge	Balanss	BalanssM6ngus	EsimeneIlmumine	ViimaneIlmumine	KaarteN6idatud	KaarteV6iduga
1	JplbbBahF/n1ERKuIXOyPw	1309	569.3	0.434912	55	631	42	25
2	EEQmXtQMzFJdmdd+AG8CWQ	1190	476.2	0.400168	81	661	80	47
3	q55FB8RNtQ+bM/Wvev9AUg	2430	950.5	0.391152	7	716	69	53
4	1u3lFKeNNI/L0VIcPWWHNA	1743	650.9	0.373437	16	748	60	37
5	IGte2AahDaTI8y1w7iNnVw	1713	639.0	0.37303	29	685	41	29
6	drKg+7JekfxR/0poTEK9Tg	1797	568.0	0.316082	16	727	42	24
7	kI4uHWfE9AkrtduyI7Drxw	2371	698.2	0.294475	41	721	55	40
8	ipxXdKAKjlxiVXZUKNpcDQ	2603	750.5	0.288321	9	750	70	52
9	HZkVla4kK2SI42CVrPZ/ig	7191	1905.0	0.264915	3	742	260	209
10	XW6LBvNMYzqJgNPzyAZh/g	2202	571.0	0.25931	19	740	47	38
11	a4IoVeRA8jeQR01dNic1PQ	2768	698.5	0.252348	9	749	65	54
12	uqwgmwBaqoBuPbS0xO2rwQ	3352	831.5	0.248061	0	744	100	56
13	inedarZeZYwn+j1jM/LBKg	4246	1051.9	0.247739	29	740	192	129
14	FFo5IbK7jvpjAWJplW2JJQ	1665	401.7	0.241261	28	748	45	36
15	qanMEqjABjWxC/JgQ/0l7w	6279	1493.3	0.237824	16	740	155	108
16	67of3RIuxyPF0TfDmAPJ6Q	3450	819.3	0.237478	2	737	68	48
17	8ehL7qlZzjgUh2H35iiWLw	3402	794.2	0.233451	14	748	68	48
18	h8mVE9pWGZkDzS8Dmkd3aA	2134	494.8	0.231865	28	742	46	32
19	jWDRpL0TJljg+mBCE9ZMqA	1568	362.1	0.230931	15	740	51	32
20	i9a2Kn3TYzz3AaJ/WOmPFg	1847	423.4	0.229237	3	724	45	35

Joonis 7. Mängijate valik andmebaasist.

Kui vaatame hoolikalt joonist (Joonis 7.), siis näeme, et andmebaasis sisalduvad modelleerimiskandidaadid on juba järjestatud vastavalt meid huvitavate kriteeriumite järgi koostatud päringule. Järjestuses on kõige paremad (kõige kasumlikumad) mängijad eespool. Kindlustamaks teadmist andmete õigsusest on võetud mängijad, kes on mänginud vähemalt 1000 mängu – kokku 162 inimest.

Otsustasin valida 2 mudeli loomiseks järgmised mängijad:

- **Mudel1:** mängija "HZkVla4kK2SI42CVrPZ/ig" – teistest suurema näidatud kaartide arvu (260) ja suhteliselt hea mänguoskuse (võitnud 0.265 rahaühikut mängus) tõttu.
- **Mudel2:** mängijate "JplbbBahF/n1ERKuIXOyPw" (0.435), "EEQmXtQMzFJdmdd+AG8CWQ" (0.400), "q55FB8RNtQ+bM/Wvev9AUg" (0.391), "1u3lFKeNNI/LOVIcPWWHNA" (0.373), "IGte2AahDaTI8y1w7iNnVw" (0.373), "drKg+7JekfxR/0poTEK9Tg" (0.316), "kI4uHWfE9AkrtduyI7Drxw" (0.294), "ipxXdKAKjlxiVXZUKNpcDQ" (0.288), "XW6LBvNMYzqJgNPzyAZh/g" (0.259), "a4IoVeRA8jeQR01dNic1PQ"(0.252), "uqwgmwBaqoBuPbS0xO2rwQ" (0.248), "inedarZeZYwn+j1jM/LBKg" (0.248), "FFo5IbK7jvpjAWJplW2JJQ" (0.241), "qanMEqjABjWxC/JgQ/0l7w" (0.238), "67of3RIuxyPF0TfDmAPJ6Q" (0.237),"8ehL7qlZzjgUh2H35iiWLw" (0.233), "h8mVE9pWGZkDzS8Dmkd3aA" (0.232), "jWDRpL0TJljg+mBCE9ZMqA" (0.231), "i9a2Kn3TYzz3AaJ/WOmPFg" (0.229) kogum – teistest parema mänguoskuse tõttu.

2.5 Treeningpakettide koostamine

Kui atribuudid ja modelleeritavad mängijad on leitud, siis järgmiseks sammuks peab olema treeningandmete koostamine klassifikaatoritele. Selleks on vaja atribuudid vastavalt mängusituatsioonile väärtustada.

Treeningpakett koosneb andmeridadest, millest iga rida ongi treeningkomplekt või sisendandmete kogum (ka sisendvektor).

Et treeningandmeid oleks võimalik Weka-s otse ja API kaudu kasutada, tuleb need salvestada Weka tarkvarale sobivale kujule. Valida on Weka-le omase arff- ja csv-failiformaatide vahel. Lihtsuse tõttu kodeerimisel valisin viimase.

Kuna pokkeri pakkumisvoorud, eriti *preflop*, oma olemuselt on erinevad, siis otsustasin koostada mudeli [3] eeskujul iga vooru jaoks eraldi – järelikult vajame ka eraldi treeningpakette – kokku 8 csv-formaadis faili (kummalegi loodavale mudelile iga pakkumisvooru jaoks).

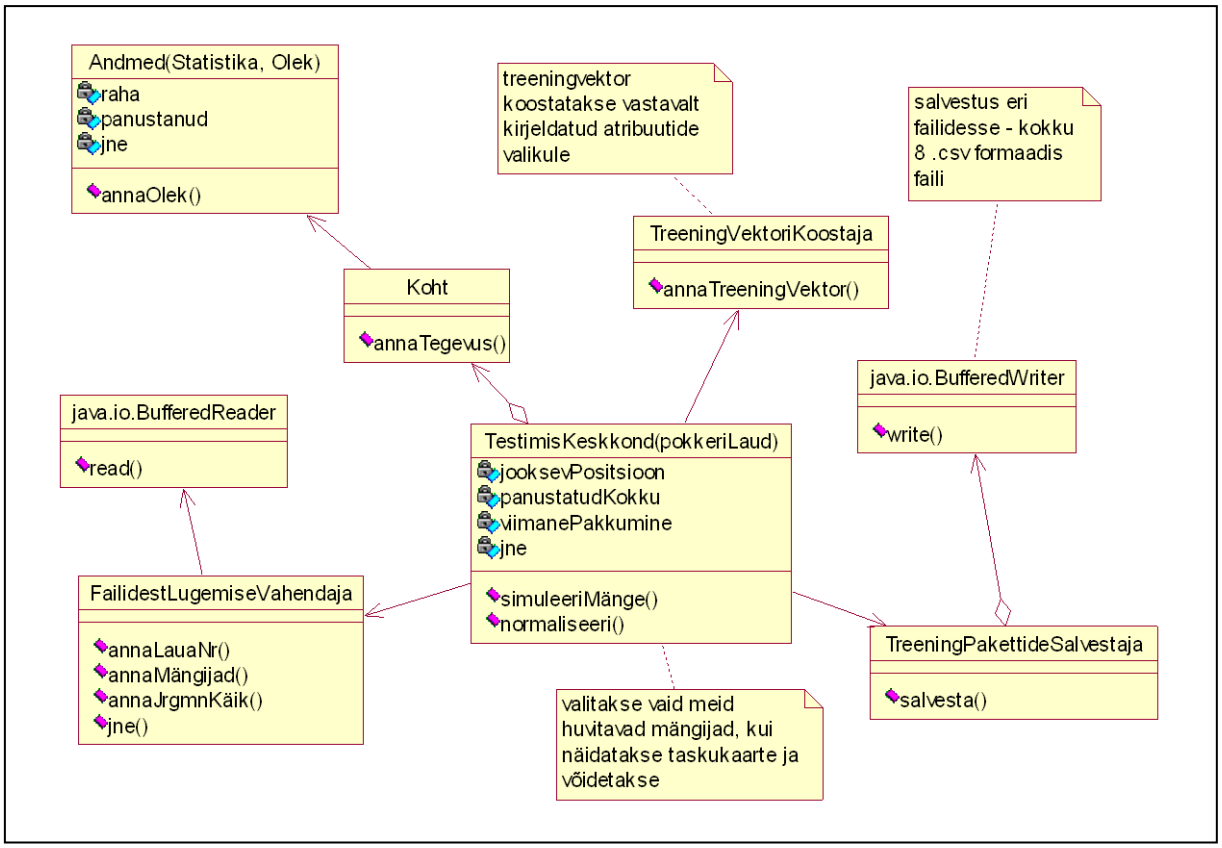
Siinkohal ei saanud kuidagi mööda minna treeningpaketi atribuutide normaliseerimisest. Kuigi edaspidikirjeldatav RandomForest klassifikaator töötab ka normaliseerimata andmetega, siis SVM kasutatavad andmed peaksid olema normaliseeritud just selleks, et vähemalt enne mudeli häälestamist kõrvaldada olukord, kus mõned atribuudid võiksid teistega võrreldes domineerida [5]. Nagu hiljem töö käigus selgub, on võimalik ja teinekord ka ülesande lahendamiseks vajalik, peale normaliseerimist mõne atribuudi väärtust jälle suurendada, tõstmaks tema mõju võrreldes teistega.

Kasutades Weka normaliseerimisfunktsiooni selgus, et normaliseerides andmeid kujule [0,1] (tänu harva esinevatele mänguolukordadele) tekkis palju olukordi, kus atribuutide enamik väärtusi sattus vahemiku [0,1] ühte äärde, mis põhjustab ebatäpsusi klassifitseerimisel. See tingis vajaduse realiseerida normaliseerimiseks oma algoritm, mis seisneb korrutamises mingi valitud teguriga (normaliseerimistegur).

Valisin normaliseerimistegurid visuaalselt kasutades Weka visualiseerimisfunktsiooni – nii, et vahemikku [0,1] satuks 90-95% väärtusi. Ülejäänud 5-10% (eelmainitud ekstreemsemad olukorrad) olid ka normaliseerimisteguriga korrutades 1-st suuremad – need võrdsustasin 1-ga.

Koostatud algoritm (Joonis 8.):

- Jälgib pidevalt, kas meid huvitav mängija on mängimas.
- Kui ta juhtub oma taskukaardid mängu lõpus avalikustama ning sealjuures võidab, koostatakse iga antud mängija otsustuse kohta selles mängus treeningvektor.
- Saadud treeningvektori atribuutide väärtused normaliseeritakse ja seejärel salvestatakse vastavasse treeningandmete faili.



Joonis 8. Treeningpakettide koostamine ja salvestus.

Salvestatud treeningandmete suurus alljärgnevas tabelis (Tabel 1.):

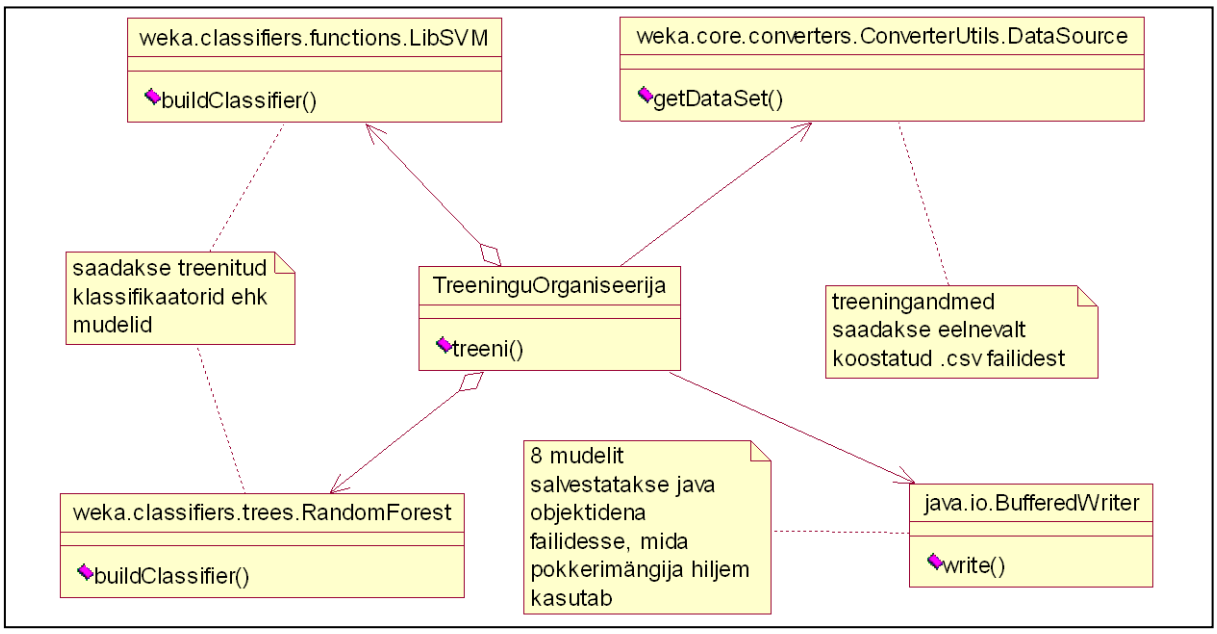
Tabel 1. Kasutatud treeningpakettide suurus.

Treeningpakettide suurus = kasutatud treeningvektorite arv				
	Preflop	**Flop**	**Turn**	**River**
Mudel1	244	161	112	67
Mudel2	1021	836	612	405

Treeningpakettide suurust vajame järgnevas analüüsis.

2.6 Klassifikaatorite valik ja treenimine

Ülesande lahendamisel kasutatavateks klassifikaatoriteks on valitud esiteks LibSVM pakett [6] (*Support Vector Machine*, edaspidi ka SVM), mida on võimalik integreerida Weka-sse ning mida saab käivitada ühe klassi reziimis mängija tegevuse *fold* genereerimiseks vastavalt ülesandepüstituses toodud hüpoteesile ja teiseks Weka tarkvara koosseisu kuuluv RandomForest klassifikaator, mida kasutati mängijate tegevuste saamiseks. Viimase asemel oleks võinud olla valitud ka mõni muu Weka klassifikaator, kuid see valiti autori kogemuse põhjal selle algoritmi suhteliselt kiire töö ja ka eelnevatel katsetustel saadud täpsete tulemuste pärast.



Joonis 9. Treenimine.

Klassifikaatorite treenimine sooritati järgmise algoritmi kohaselt (Joonis 9.):

- Hangi csv-formaadis failist treeningpakett.
- Treeni klassifikaator (loo mudel) kasutades klassifikaatori meetodit buildClassifier().
- Salvesta saadud mudel faili.
- Korda eelnevaid tegevusi igale klassifikaatorile.

3. Testimine ja tulemused

Käesolev peatükk kirjeldab pokkerimängijate mudelite testimist. Kõigepealt pannakse kokku süsteem testkeskkonnast ja testitavatest ning abimängijatest. Seejärel viiakse testid läbi. Testide genereeritud tulemused esitatakse tulpdiagrammina.

3.1 Testimine

Testimiseks kasutasin testkeskkonda, kuhu integreerisin loodud mudeleid kasutavad tarkvaralised mängijad, samuti erinevate mänguoskustega abimängijad. Allpool esitan vastava loetelu koos oskuste kirjeldusega ja osavõtuarvuga mängudest:

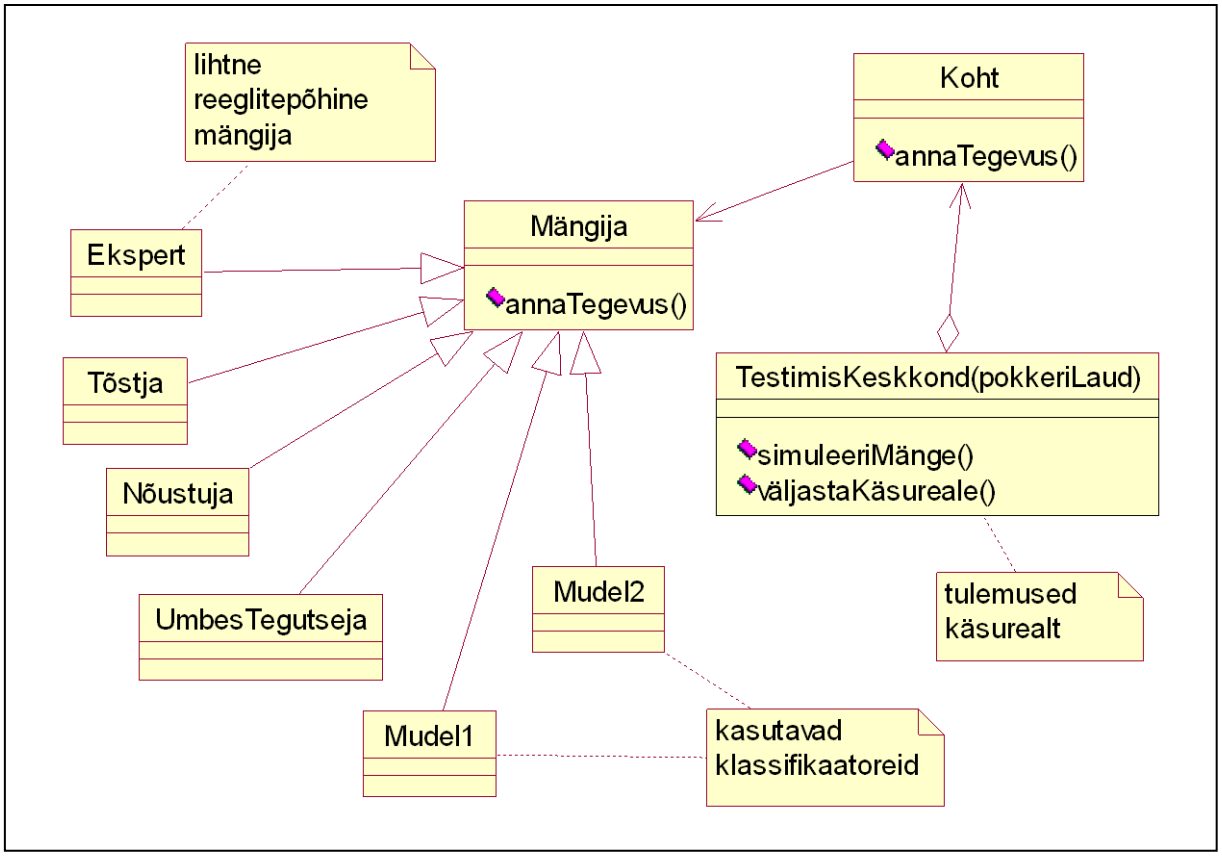
- Nõustuja – oskab ainult eelneva pakkumisega nõustuda – 2 eksemplari.
- Tõstja – oskab ainult eelnevat pakkumist tõsta või siis nõustuda, kui ei jätku raha tõstmiseks – 2 eksemplari.
- Umbes mängija – oskab kõiki tegevusi, kuid kasutab neid spontaanselt – 2 eksemplari.
- Reeglipärane ekspert – kasutab otsuste vastuvõtmisel järgnevat algoritmi [7] eeskujul esitatuna pseudokoodis (Joonis 10.) – 1 eksemplar:

```
If tn > oponentide arv * 1.5 then tõstaPakkumist
Else if ( pakkumisVoor != preflop
    && tn > 0.4 ) then nõustuPakkumisega
Else if tn > oponentide arv then nõustuPakkumisega
Else if  amountToCall == 0 then nõustuPakkumisega
Else if tn > potOdds then nõustuPakkumisega
Else loobuPakkumisest
```

Joonis 10. Reeglipärane ekspert.

- Mudel1 – testitav mängija, mis kasutab punktis 2.4 valitud modelleerimiskandidaadi käitumise järgi loodud mudelit – 1 eksemplar.
- Mudel2 – testitav mängija, mis kasutab punktis 2.4 valitud modelleeritavate kogumi järgi loodud mudelit – 1 eksemplar.

Testimiseks loodi rakendus (Joonis 11.).



Joonis 11. Testimine testkeskkonnas.

Testimine toimus järgmise plaani järgi:

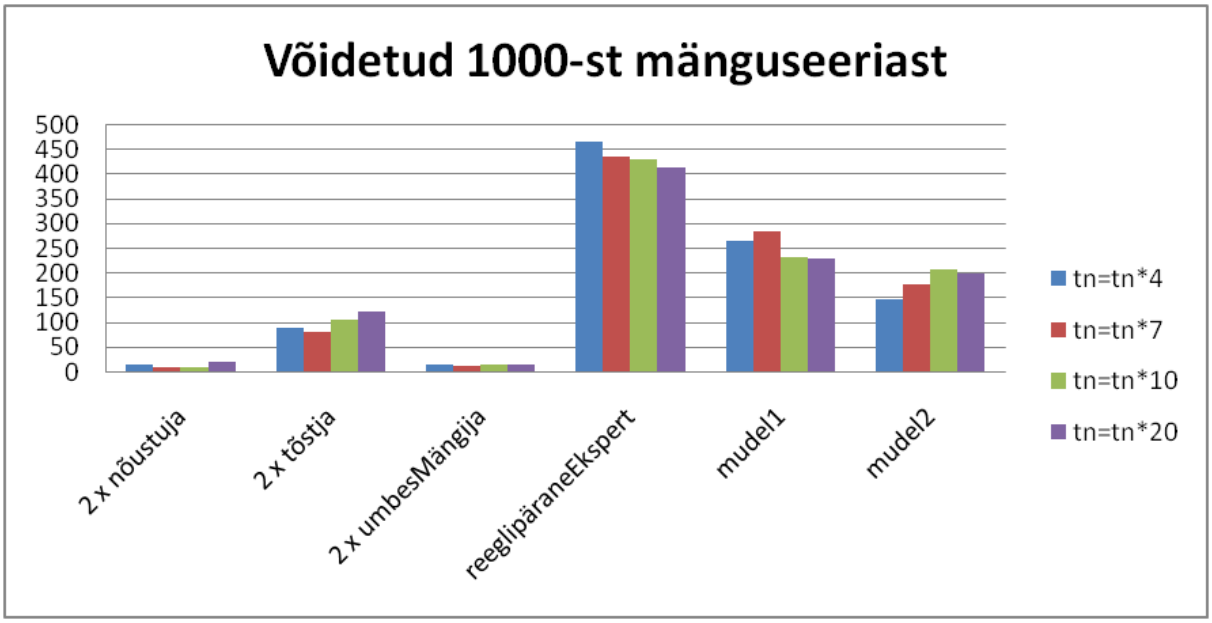
- Paiguta mängijad pokkerilauda valides igale mängijale suvalise koha.
- Anna igale mängijale võrdne arv rahaühikuid.
- Lase osavõtjatel mängida pokkerit kuni üks mängija on kogu raha endale võitnud.
- Korda eelnevat 1000 korda.

3.2 Tulemused

Esimene katseseeria ebaõnnestus. Tõrge tekkis juhul, kui 2 viimaseks mängijaks olid jäänud Mudel1 ja Mudel2. Tekkis olukord, kus mõlemad mudelid lahkusid pidevalt mängust loobudes edasisest pakkumisest.

Analüüsides olukorda tulin järeldusele, et mõni atribuut (mängijate arv?) või atribuutide koostoime avaldab nii suurt mõju, et antud situatsiooni kirjeldav sisendvektor sunnib SVM-i otsustama loobumise kasuks. Jätsin atribuutide ja nendevaheliste seoste uurimise tulevikuülesandeks ning otsustasin lahendada probleemi lähtudes töö käigus saadud kogemustest atribuudi võiduTõenäosus mõju suurendamise teel. Selleks korrutasin eelminitud atribuudi väärtusi mingi arvuga alates 2-st, treenisin klassifikaatorid uuesti ning kontrollisin, kas tõrge veel eksisteeris. Kordasin eelnevat kuni probleem kadus (kordaja 4 korral).

Töö lõppresultaadid kujunesid välja katsetest, mis viidi läbi vastavalt eelpoolkirjeldatud testimisplaanile. Tulemused näitavad mängijate mänguoskusi väljendatuna võitude arvus 1000-st lõpunimängitud mängude seeriast. Viidi läbi 4 erinevat eelpoolmainitud katseseeriat, milles igaühes suurendati atribuudi võiduTõenäosus mõju 4, 7, 10 ja 20 korda. Esitatud on testimisrakenduse väljatrükid (Tabel 2.) ja kogu tööd kokkuvõttev tulpdiagramm, mis koostati antud väljatrükkide põhjal (Joonis 12.).



Joonis 12. Tulemused.

Nagu diagrammilt võib välja lugeda, mänguoskuste poolest parimaks osutus reeglitest juhinduv (Joonis 10.) ekspertteadmistega mängija. Teisele kohale asetus mudel, mis oli modelleeritud ühe inimmängija käitumise järgi ning kolmandaks osutus mudel, mis koostati arvestades paljude mängijate kogemusi.

Atribuudi võiduTõenäosus (tn) mõju reguleerimine avaldas mänguoskustele järgnevat toimet:

- Reeglipõhise mängija oskused kahanesid atribuudi tn mõju tõustes.
- Mudel1 tulemused valdavalt kahanesid samal ajal, kuigi parimaks katseseeriaks oli tn=tn*7.
- Mudel2 käitus vastupidiselt Mudel1-le, näidates paremaid resultaate tn mõju kasvades, kuigi parimaks katseseeriaks osutus tn=tn*10.
- Kui võtta Mudel1 halvim ja Mudel2 parim tulemus, siis need on peaaegu võrdsed, kuigi üldine trend näitab siiski Mudel1 edu Mudel2 ees.

On selge, et mudelid vajavad täiustamist. Järeldused tehtust ja ettepanekud edasiseks uurimistööks on esitatud käesoleva uurimuse järgmises, kokkuvõtvas peatükis.

Tabel 2. Tulemused.

Tulemused	
----------tulemused---------- tn kordaja: 4 l6punim6ngitudRingeKokku: 1000 m6ngeKokku: 132266 ------------võidud----------- n6ustuja: 15 t6stja: 89 umbesM6ngija: 16 reeglip6raneEkspert: 467 mudel1: 264 mudel2: 147 BUILD SUCCESSFUL (total time: 19 minutes 45	----------tulemused---------- tn kordaja: 7 l6punim6ngitudRingeKokku: 1000 m6ngeKokku: 69991 ------------võidud----------- n6ustuja: 6 t6stja: 119 umbesM6ngija: 9 reeglip6raneEkspert: 414 mudel1: 276 mudel2: 176 BUILD SUCCESSFUL (total time: 11 minutes 55
----------tulemused---------- tn kordaja: 10 l6punim6ngitudRingeKokku: 1000 m6ngeKokku: 100614 ------------võidud----------- n6ustuja: 10 t6stja: 106 umbesM6ngija: 15 reeglip6raneEkspert: 429 mudel1: 232 mudel2: 208 BUILD SUCCESSFUL (total time: 14 minutes 55	----------tulemused---------- tn kordaja: 20 l6punim6ngitudRingeKokku: 1000 m6ngeKokku: 140940 ------------võidud----------- n6ustuja: 21 t6stja: 121 umbesM6ngija: 16 reeglip6raneEkspert: 413 mudel1: 229 mudel2: 200 BUILD SUCCESSFUL (total time: 20 minutes 48

4. Kokkuvõte

Käesoleva töö põhieesmärgiks oli tarkvaralise pokkerimängija koostamine kasutades Weka klassifitseerimisalgoritme, sobiva algandmete kogumi leidmine ja andmete ettevalmistus klassifitseerimisülesande täitmiseks. Teiseks eesmärgiks oli kontrollida autori poolt ülesandepüstituses esitatud hüpoteesi ÜKK kasutamisest pokkerimängija tegevuse *fold* genereerimisel.

Olulisemad järeldused tehtud tööst:

- Ühe klassi klassifikaator on kasutatav vastavalt püstitatud hüpoteesile, kuid klassifikaatori häälestamise keerukuse tõttu jääb küsitavaks selle kasutamise otstarbekus.

- Tulenevalt läbiviidud testide tulemustest võib tõdeda, et ka idee klassifikaatorite kasutamisest pokkerimängija loomisel jääb just autori silmis kahtluse alla, sest lihtne ekspertteadmistega pokkerimängija edestas loodud mudeleid. See ei tähenda aga, et see idee ei ole kindlasti nende vahenditega realiseeritav – ilmselt napib ka töö autoril vastavasisulisi kogemusi.

- Nagu selgus testides vastupidiselt autori poolt eeldatule: mudel, mille treeningpakett sisaldas paljude mängijate kogemust suures mahus (Tabel 1.), osutus valitud testimistingimustel nõrgemaks võrreldes mudeliga, mis koostati vaid ühe mängija mänguandmete väiksemas mahus põhjal. Siin võib teha oletava järelduse, et kuigi tegu oli valikuga võitvatest mängijatest, esindasid modelleeritud kandidaadid erinevaid mängustiile ja harjumusi. Võib-olla oleks tulnud nende mängustiili eelnevalt uurida ja koondada erinevad mängustiilid erinevatesse mudelitesse.

Ettepanekud edasiseks uurimistööks ja edasiarenduseks:

- Vastavalt märkusele ÜKK kasutamise keerukuse kohta lahendust otsides tuleks leida või siis osta täielik mängude andmebaas, mis sisaldab mängijate 100%-liselt taskukaarte – sellise andmeallikaga on võimalik koostada pokkerimängija mudel kasutades vaid ühte klassifitseerimisalgoritmi, milleks on antud juhul RandomForest.

- Antud ülesande piires kasutati mängijate taskukaartide informatsiooni kandjana arvutatud atribuuti võiduTõenäosus (tn), mida annaks eeldatavasti edukalt asendada iga kaardi jaoks eraldi uue atribuudi loomisega (52 atribuuti) kujul 1 või 0.

- Juhul, kui tekib soov arendada antud töö käigus koostatud mängijate pakkumisalgoritme, tuleks hoolikamalt uurida testimispeatükis kirjeldatud tõrkesituatsiooni. Tõrke põhjustanud atribuut või atribuutide grupp tuleks kõrvaldada või selle mõju vähendada.

- Tulenevalt valitud testimisstrateegiast ei ole teada, kuidas käituvad koostatud mudelid olukorras, kus mängust osavõtjate arv 9 mängijaga pokkerilauas reaalajas kõigub vähem kui 9 ja 2 vahel. Selle teadmise hankmiseks tuleks korraldada testid, kus mõne mängija lahkumisel asendatakse see mingi aja möödudes uuega.

- Autor ei ole väga põhjalikult uurinud kasutatud atribuutide mõju loodud mudelite oskustele. Kasulik oleks reguleerida atribuutide valikut konsulteerides mõne pokkerispetsialistiga. Kõne alla võiks tulla ka automaatse atribuutide valimise ja töötlemise rakenduse loomine, kasutades näiteks geneetilisi algoritme.

Kasutatud kirjandus

[1] Pottmann, Elar, *Arvuti mängib pokkerit*. Tallinn, 2014.

[2] TheLeanStartup. (2015, 20.05.) [Online]. http://theleanstartup.com/

[3] Teófilo, Luís F. G., *Building a Poker Playing Agent based on Game Logs using Supervised Learning*. Porto, 2010.

[4] Outflopped. (2015, 20.05.) [Online]. http://web.archive.org/web/20110308091456/http://www.outflopped.com/questions/286/obfuscated-datamined-hand-histories

[5] Hsu, C., Chang, C., Lin, C., *A Practical Guide to Support Vector Classification*. Taiwan, 2010.

[6] LibSVM. (2015, 20.05.) [Online]. http://weka.sourceforge.net/packageMetaData/LibSVM/index.html

[7] Booker, Lawrence R., *A No Limit Texas Hold'em Poker Playing Agent*. London, 2004.