Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Simon Janežič

Pristopi strojnega učenja za analizo igre League of Legends

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kazalo

Povzetek

V diplomskem delu želimo z uporabo strojnega učenja napovedovati izide tekem igre League of Legends. Igra je večigralska in združuje elemente strateških in akcijskih iger. Zaradi popularnosti igre so večkrat na leto po vsem svetu organizirana tekmovanja, kjer se profesionalne ekipe borijo za denarne nagrade. Napovedovati poizkušamo tako profesionalne kot neprofesionalne tekme. Izziv je že pridobitev podatkov za oba tipa tekem. Podatke za neprofesionalne tekme pridobimo s pomočjo uradne spletne knjižnice, za profesionalne pa s filtriranjem vsebin spletnih strani, ki vsebujejo rezultate tekem. Raziskavo pričnemo s krajšo analizo, kjer primerjamo statistike igralcev različnih rangov ter najdemo zanimive ugotovitve, ki bi jih igralci nižjih rangov lahko uporabili za izboljšanje. Na podatkih nato uporabimo in testiramo algoritme za klasifikacijo. Pri neprofesionalnih tekmah dobimo podobne rezultate kot obstoječ članek na to temo, ki je bil objavljen v Journal of Machine Learning Research. Primerjamo tudi rezultate klasifikatorjev obeh tipov tekem ter ugotovimo, da so profesionalne tekme bolj napovedljive.

Ključne besede: strojno učenje, podatkovno rudarjenje, League of Legends, napoved izida igre.

Abstract

Our goal is to use machine learning for predicting winners of League of Legends matches. League of Legends is a multiplayer game that combines elements from strategic and action games. Every year, multiple professional League of Legends competitions are being held acros the globe. We try to predict both professional and non-professional matches. Getting data for both types of matches is already a challenge. For non-professional matches official application programming interface is used, while data for professional matches is gathered using web scraping. We begin our research by using the collected data for initial analysis, where we compare player statistics from different ranks. We find some interesting differences that lower-ranked players could use to improve their game without huge amount of effort. After that, we use standard machine learning approaches to predict match winners. Classificators for non-professional matches yield similar results to a recently published study. We also compare the results for both types of matches and find that it is easier to predict the outcomes of professional matches.

Keywords: machine learning, data mining, League of Legends, game outcome prediction.

Poglavje 1

Uvod

League of Legends je večigralska spletna igra razvita v podjetju Riot Games[1]. Združuje elemente strateških in akcijskih iger. Vsak igralec nadzoruje svojega heroja z unikatnimi sposobnostmi in se skupaj s soigralci bori proti nasprotni ekipi igralcev. Vsako ekipo sestavlja 5 igralcev. Cilj igre je uničiti glavno nasprotnikovo zgradbo, ki je z vseh strani zavarovana z obrambnimi zgradbami. Za dosego tega cilja je potrebna strategija, ekipno usklajevanje in mehanične sposobnosti posameznika (npr. refleksi, natančnost). Igra je izredno popularna, saj ima po zadnjih informacijah 67 milijonov mesečno aktivnih uporabnikov [5]. Primer izgleda igre podaja slika 1.1.

Zaradi popularnosti igre so večkrat na leto po vsem svetu organizirana tekmovanja, kjer se profesionalne ekipe borijo za denarne nagrade. V Ameriki in Evropi sta najbolj prestižni organizirani tekmovanji *NA League Championship Series* in *EU League Championship Series*, kjer sodeluje 10 profesionalnih ekip z vsakega kontinenta. Podobna tekmovanja obstajajo tudi na Kitajskem, v Južni Koreji ter jugovzhodni Aziji. Najboljše ekipe vsake posamične regije se enkrat na leto zberejo na svetovnem prvenstvu, katerega glavna nagrada je v letu 2015 znašala milijon dolarjev [5].

[1] `http://riotgames.com`

Slika 1.1: Uporabniški vmesnik igre.

S priljubljenostjo tekmovanj je postalo zanimivo tudi področje napovedovanja izzidov tekem, saj se pojavlja vse več stavnic, ki ponujajo stave na različna profesionalna tekmovanja. Tudi napovedovanje neprofesionalnih tekem je koristno, saj se lahko igralec na podlagi verjetnosti zmage odloči, da prihodnje tekme ne bo igral in si s tem prihrani od 20 do 50 minut, kar je običajna dolžina ene tekme.

V diplomski nalogi poizkušamo z metodami strojnega učenja napovedati izid profesionalnih in neprofesionalnih (običajnih) tekem na podlagi zgodovine igranja udeležencev. Pri profesionalnih tekmah so tako kot pri večini športov nekatere ekipe konsistento boljše kot druge. Za običajne tekme pa igra teži k temu, da ima vsaka stran enake možnosti za zmago. To doseže z ocenami igralcev, ki temeljijo na ocenjevalnem sistemu Elo, poznanem po njegovi uporabi pri šahu, bilijardu in drugih igrah [2]. Zaradi te razlike med dvema tipoma iger se pričakuje boljša napovedna sposobnost pri profesionalnih tekmah.

1.1 Opis igre League of Legends

V eni tekmi League of Legends naekrat sodeluje 10 igralcev. Razdeljeni so v modro in rdečo ekipo. Pred pričetkom tekme mora vsak igralec iz nabora več kot 100 herojev izbrati enega, ki ga bo nadzoroval. Dva igralca v isti tekmi ne moreta izbrati enakega heroja. Izbira herojev poteka v večih fazah. Modra ekipa ima manjšo prednost, saj začne z izbiro prvega heroja. Vsak igralec izbere heroja s katerim bo zasedel eno izmed spodaj opisanih pozicij/vlog na igralnem polju (slika 1.2) [5].

- Zgornja (top): običajno zelo vzdržljivi heroji, ki težko umrejo. V skupinskih bojih je njihova naloga zamotiti ali pa celo ubiti najbolj nevarne nasprotnikove heroje.
- Srednja (mid): heroji, ki lahko zelo hitro umrejo in tudi hitro ubijejo nasprotnikovega heroja. V skupinskih bojih poleg nosilne pozicije naredijo največ škode. Običajno heroji, ki lahko napadejo z daljše razdalje.
- Nosilna (carry): heroji, ki lahko zelo hitro umrejo in tudi hitro ubijejo nasprotnikovega heroja. V skupinskih bojih naredijo največ škode. Običajno heroji, ki napadajo s krajše razdalje.
- Podporna (support): heroji, katerih sposobnosti omogočajo podporo ekipi. Primera takih sposobnosti sta preprečitev soigralčeve smrti in omejitev gibanja nasprotnikovega heroja. Brez pomoči zelo težko ubijejo nasprotnikovega heroja.
- Gozdna (jungle): običajno dobra kombinacija vzdržljivosti in napadalne moči. Za razliko od ostalih pozicij se v v začetnem delu tekme premika po celem igralnem polju ter pomaga soigralcem, ki to potrebujejo.

Večino herojev ne spada strogo v eno vlogo ampak jih je možno igrati na več načinov. Primere nekaterih herojev in njihove izstopajoče sposobnosti

najdemo na tabeli 1.1.

Tabela 1.1: Primeri nekaterih herojev in njihove izstopajoče sposobnosti.

ime heroja	**izstopajoče sposobnosti**
Blitzcrank	Izstreli desno roko, zagrabi nasprotnika in ga povleče k sebi.
Ashe	Izstreli puščico, ki potuje čez celo igralno polje ter rani in začasno ustavi nasprotnika.
Twitch	Sposobnost aktiviranja nevidnosti. Po pretečenem času ali po prvem uporabljenem napadu postane viden, njegovi napadi pa so začasno veliko hitrejši.

Po izbiri herojev se tekma prične. Igralno polje z označenimi zgradbami in s pozicijami, ki jih v začetku tekme zasedejo igralci je prikazano na sliki 1.2. Cilj je uničiti glavno nasprotnikovo zgradbo, ki je z vseh strani zavarovana z obrambnimi zgradbami. Preden lahko ekipa te zgradbe uniči morajo svoje heroje primerno nadgraditi. Heroji se nadgrajujejo z aktivnostmi, ki vključujejo ubijanje nasprotnikovih herojev, uničevanje nasprotnikovih zgradb in ubijanje raznih pošasti, ki jih nadzoruje računalnik. Vse te aktivnosti igralcu prislužijo igralno valuto (denar) in točke izkušenj. Heroj preko pridobljenih točk izkušenj postaja v vseh pogledih vedno močnejši. Kako porabiti pridobljen denar pa je izbira igralca. Skozi tekmo lahko igralec kupi veliko različnih objektov, ki dodatno nadgradijo določene statistike heroja ali pa pomagajo drugim članom ekipe. Primer takih je objekt, ki izboljša obrambo vseh bližnjih soigralcev [5].

Ko heroj umre, igralec nekaj časa ne more narediti ničesar. Daljša kot je tekma, daljše so kazni za smrt. Hkrati pa sčasoma vsi heroji postajajo dovolj močni, da precej hitro lahko uničijo nasprotnikove zgradbe. Zaradi teh razlogov proti koncu tekme igralci vedno pogosteje zapustijo njihove pozicije in se borijo kot skupina, saj vsaka smrt lahko pomeni dovolj časa za konec tekme.

Slika 1.2: Igralno polje z označenimi pozicijami in zgradbami. Polje je razdeljeno na dve polovici, ki pripadata modri in rdeči ekipi.

Za razumevanje diplomskega dela je potrebna tudi obrazložitev koncepta odkrivanja nasprotnikovih informacij in skrivanja svojih. Vsak igralec v tekmi vidi le omejeno količino igralnega polja: svojo okolico, okolico soigralcev, okolico ekipnih zgradb in okolico vseh postavljenih razsvetljevalcev ekipe. Razsvetljevalec je objekt, ki ga med tekmo lahko pridobimo na več načinov: nekaj jih je danih zastonj, nekaj pa jih lahko tekom igre kupimo. Ko ga postavimo na igralno polje nam za nekaj minut razkrije njegovo okolico. Načeloma je ta objekt za nasprotnikovo ekipo neviden, a obstajajo objekti, ki jih lahko igralci uporabijo, da nasprotnikove razsvetljevalce najdejo ter jih uničijo. S postavljanjem in uničevanjem teh objektov torej govorimo o odkrivanju nasprotnikovih informacij in skrivanju svojih [5]. Če razsvetljevalci ne bi bili del igre, bi bila ta veliko bolj odvisna od sreče, saj ob mnogih premikih po igralnem polju ne bi vedeli, če nas nekaj korakov naprej čaka nasprotnik.

1.2 Sorodna dela

V članku objavljenem v Journal of Machine Learning Research [4] so poizkusili s strojnim učenjem napovedati izide neprofesionalnih tekem League of Legends na podlagi zgodovine igralcev. Z metodo zlaganja klasifikatorjev, ki združuje odločitvena drevesa, naključne gozdove, naivnega bayesa, logistično regresijo in podporne vektorje so uspeli doseči klasifikacijsko točnost 65%. Na podlagi rezultatov so predlagali izboljšavo sistema, ki pred začetkom tekme išče enakovredne igralce. Njihove rezultate v našem delu uporabimo za primerjavo z našimi.

Raziskovalca iz Univerze Stanford[2] sta s strojnim učenjem poizkušala napovedati izide podobne igre Dota 2. Kot vhodne podatke sta uporabila le izbrane heroje obeh ekip. Z logistično regresijo jima je uspelo doseči 70% natančnost. Na podlagi rezultatov sta nato razvila priporočilni sistem, ki pred tekmo na podlagi izbranih herojev nasprotnikov in soigralcev predlaga izbiro

[2] `http://www.danieljamesperry.com/s/Dota2.pdf`

optimalnega heroja. Sistem deluje tako, da za vse možne heroje na podlagi prej naučenega modela izračuna verjetnost zmage. Raziskovalci z univerze v Waterlooju[3] so prav tako s podatki o izbranih herojih poizkušali napovedati izide za igro League of Legends. Dosegli so precej manjšo natančnost (55%).

V članku, ki je bil predstavljen na konferenci Foundations of Digital Games [10] so za napoved izida uporabili nekoliko drugačen pristop. Izide igre Dota 2 so poizkusili napovedati z analizo bojev, ki se dogajajo med igro. Boj je interakcija med heroji nasprotnih ekip. To interakcijo so modelirali z grafom. Vozlišča predstavljajo vpletene igralce, usmerjene povezave pa interakcije med njimi. Obstaja tudi posebno vozlišče smrti. Povezava s tem vozliščem pomeni smrt heroja. Tekmo so predstavili kot zaporedje grafov (bojev). S teh grafov so nato za vsa možna vozlišča (10 igralcev in vozlišče smrti) izračunali več metrik (npr. število povezav, ki kažejo v to vozlišče) in te uporabili za strojno učenje. Dosegli so 80% natančnost. Potrebno je omeniti, da je to natančnost pri napovedovanju izida, kjer so vsi boji dane tekme znani. Modela torej ne bi bilo mogoče uprabiti za napoved izida tekme, preden se ta začne. To tudi ni bil cilj članka. Cilj članka je bilo najti vzorce zmagovalnih ekip v bojih. Ker so za strojno učenje uporabili odločitveno drevo, so iz naučenih pravil te vzorce tudi našli.

[3] http://arghorashy.ca/assets/lol_match_prediction.pdf

1.3 Cilji diplomske naloge

V diplomski nalogi smo želeli:

1. Pridobiti podatke o profesionalnih in neprofesionalnih igrah.
2. Analizirati podatke z metodami podatkovnega rudarjenja ter prikazati zanimive vizualizacije.
3. Iz podatkov pridobiti značilke, ki so primerne za atributno strojno učenje.
4. Oceniti uporabnost različnih tehnik strojnega učenja ter primerjati naše rezultate z obstoječimi [4].

Poglavje 2

Uporabljene metode

Glavni cilj naše diplomske naloge je uporaba strojnega učenja za napovedovanje izida tekem. V tem pogljavu opišemo uporabljene algoritme strojnega učenja, metode za ocenjevanje njihove uporabnosti ter metode za ocenjavanje koristnosti atributov.

2.1 Algoritmi strojnega učenja

Problemi strojnega učenja so v splošnem razdeljeni na nadzorovano in nenadzorovano učenje. Pri nadzorovanem učenju razredni atribut znan pri nenadzorovanem pa ne. Razredni ali ciljni atribut je lastnost primerov učne množice, ki jo želimo iz ostalih atributov napovedati [6]. V naši učni množici je primer ena tekma. Za vsako tekmo imamo znanega zmagovalca, ki ga želimo napovedati. Zmagovalec tekme je v našem primeru torej razredni atribut, kar pomeni da bomo za napovedovanje uporabili metode nadzorovanega učenja.

Metode nadzorovanega učenja se nadalnje delijo na dve večji skupini: klasifikacija in regresija. Pri klasifikaciji imamo opravka z diskretnim razrednim atributom, pri regresiji pa z zveznim [6]. Zmagovalec tekme je lahko v naši domeni modra ali rdeča ekipa, kar pomeni, da imamo opravka z diskretnim

razrednim atributom. Torej uporabimo klasifikacijske metode.

Med znanimi klasifikacijskimi metodami smo se odločili za uporabo logistične regresije, metode podpornih vektorjev (SVM), naključnih gozdov in metode zlaganja s katero bomo združili napovedi vseh prej omenjenih algoritmov. Uporabili smo implementacijo omenjenih metod v knjižnici *scikit-learn* [8]. Knjižnjica vsebuje velik nabor metod za strojno učenje ter podatkovno rudarjenje. Uporablja popularna paketa *Numpy* in *Scipy*, ki omogočata vektorsko predstavitev, računanje in manipuliranje podatkov. Zaradi podpore vektorskih operacij je knjižnjica tudi učinkovita.

2.1.1 Logistična regresija

Kljub njenemu imenu je logistična regresija linearna metoda za klasifikacijo in ne regresijo. Pri tej metodi so verjetnosti, ki opisujejo možne izide učnega primera, modelirane s sigmoidno funkcijo [8]:

$$S(t) = \frac{1}{1 + e^{-t}}$$

Implementacijo logistične regresije v knjižnici *scikit-learn* najdemo v razredu `LogisticRegression`. Implementacija deluje tako za binarne kot večrazredne probleme. Vključuje tudi regularizacijo. Regularizacija je pomemben del večih algoritmov strojnega učenja saj preprečuje premočno prilagajanje modela na učne podatke. O premočnem prilagajanju je govora takrat, ko model strojnega učenja deluje bistevno slabše za nove, prej nevidene podatke. Moč regularizacije nadzorujemo z regularizacijskim parametrom. *Scikit-learn* ima za logistično regresijo implementirana 2 tipa regularizacije in s tem tudi 2 različna optimizacijska problema [8]:

1. *Lasso* ali *L1* regularizacija:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i(X_i^T w + c)\right) + 1\right)$$

2. *Ridge* ali *L2* regularizacija:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i(X_i^T w + c)\right) + 1\right)$$

Tu je w vektor uteži posameznih atributov, C regularizacijski parameter, y_i vrednost razreda učnega primera i in X_i vektor vrednosti posameznih atributov učnega primera i.

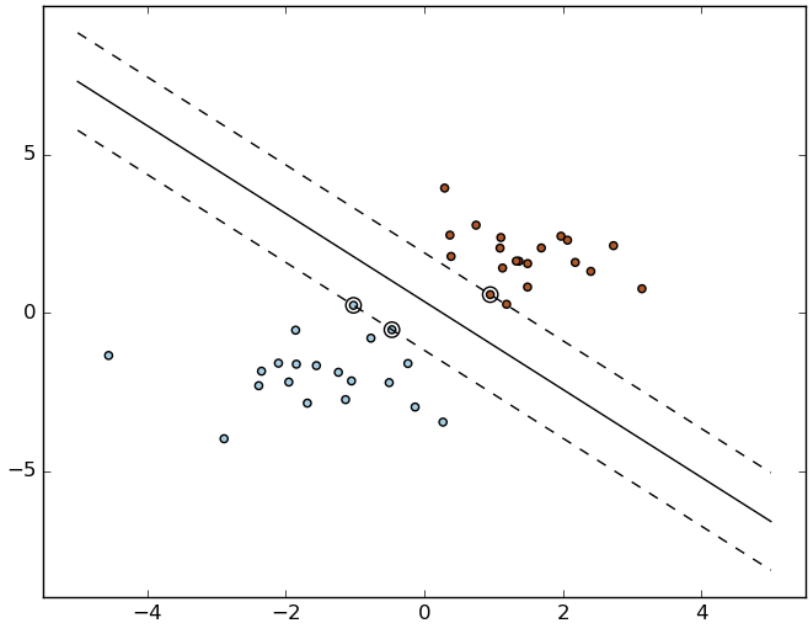
Regularizacija *Lasso* na nek način naredi tudi izbor atributov, saj lahko utež nekega atributa nastavi tudi na vrednost 0, medtem ko bo regularizacija *Ridge* takemu atributu sicer znižala vpliv, ne bo pa ga izničila [8]. V našem primeru smo poizkusili oba načina regularizacije.

2.1.2 SVM

Metode podpornih vektorjev spadajo med najbolj uspešne metode strojnega učenja. Lahko se uporabljajo za klasifikacijo in regresijo. V osnovni obliki SVM pri klasifikaciji deluje za dvo-razredne probleme [6]. Algoritem poizkuša v prostoru atributov postaviti optimalno hiperravnino (2.1). To je hiperravnina, ki je enako in najbolj oddaljena od najbljižjih primerov iz obeh razredov. Najbljižjim primerom optimalne hiperravnine pravimo podporni vektorji, razdalji hiperravnine od podpornih vektorjev pa rob. Torej je optimalna hiperravnina tista, ki ima maksimalni rob [6].

Knjižnica *scikit-leran* vsebuje več implementacij podpornih vekotorjev. Uporabil sem implementacijo v razredu `SVC`. Implementacija je zelo vsestranska, saj poleg linearne ločitve učnih podatkov podpira tudi bolj kompleksne funkcije (npr. polinomsko). Takim funkcijam pravimo jedra. Knjižnjica nam omogoča tudi implementacijo lastnega jedra. V našem primeru sem poizkusil

vsa jedra, ki jih podpira razred *SVC*. Poleg izbire jedra implementacija vsebuje tudi regularizacijo in s tem regularizacijski parameter, ki določa vpliv le te. Glede na izbiro jedra ima metoda lahko tudi dodatne parametre [8].



Slika 2.1: Primer optimalne hiperravnine v prostoru atributov. Modre pike so učni primeri, ki pripadajo prvemu razredu, rdeče pike pa drugemu.

2.1.3 Naključni gozdovi

Osnovna ideja metode naključnih gozdov je, da zgenerira več modelov odločitvenih dreves. Za klasifikacijo posameznega primera se nato uporabi metoda glasovanja - vsako odločitveno drevo prispeva en glas razredu, v katerega klasificira primer. Klasifikator naključnih gozdov nato izbere razred z največ glasovi. Za razliko od klasične metode odločitvenih dreves se pri naključnih gozdovih pri izboru atributov v vsakem vozlišču posamičnega drevesa upošteva le naključno zbrana podmnožica vseh atributov. Z metodo naključnih gozdov zmanjšamo varianco odločitvenih dreves. Spada med najboljše algoritme strojnega učenja [6].

Knjižnica *scikit-learn* implementira naključne gozdove v razredu `RandomForestClassifier`. Implementacija namesto glasovanja rezultate posamičnih odločitvenih dreves združi s povprečenjem verjetnosti za vse razrede. Torej namesto, da posamično drevo glasuje za en sam razred, upoštevamo verjetnosti vseh razredov, ki jih napove to drevo [8]. Pri tem modelu lahko nastavljamo veliko parametrov. Najpomembnejša sta število zgeneriranih odločitvenih dreves in največja velikost naključne podmožice atributov, ki bodo obravnavani za vsako vozlišče posamičnega drevesa. Nastavimo lahko tudi več parametrov za preprečitev gradnje prekompleksnih dreves. Primer takega parametra je največja globina posamičnega zgeneriranega drevesa.

2.1.4 Zlaganje večih modelov

Zlaganje [9] spada med metode združevanja modelov strojnega učenja. Združimo jih tako, da zgradimo novo učno množico, kjer vsak model predstavlja atribut, njegove vrednosti pa so napovedi modela. Tu lahko uporabimo dejanske vrednosti napovedanih razredov ali pa njihove verjetnosti. Poizkusili bomo oba pristopa. Razredni atribut pridobimo s prvotne učne množice. Nato lahko nad novo pridobljeno učno množico uporabimo poljubno metodo strojnega učenja. Želimo, da končni združeni model v napovedovanju premaga

vsakega posamičnega, ki je del njega. V našem primeru bomo z logistično regresijo združili vse do sedaj opisane metode, torej logistično regresijo, SVM in naključne gozdove.

2.1.5 Večinski klasifikator

Večinski klasifikator vedno napove isti razred in sicer tistega, ki se v podatkovni množici najpogosteje pojavi. Uporablja se le za primerjavo z ostalimi klasifikatorji. Njegove ocene predstavljajo spodnjo mejo, ki jo morajo premagati vsi koristni klasifikatorji. V *scikit-learn* je implementiran kot del razreda `DummyClassifier`.

2.2 Metode za izbiro parametrov ter ocenjevanje modelov

Izbiro parametrov ter ocenjevanje modelov smo za neprofesionalne tekme izvedli po naslednjem postopku:

1. Izbor 1000 naključnih tekem (25% celotne množice) za končno testiranje (testna množica). Preostalim 3000 rečemo učna množica.

2. Izbira parametrov modelov z uporabo mrežnega iskanja na podlagi ocen 10-kratnega prečnega preverjanja nad učno množico.

3. Izvedba končnega testiranja pridobljenih modelov nad testno množico.

Za profesionalne tekme si takšnega postopka ne moremo v celoti privoščiti, saj je število primerov majhno (300). Zaradi tega naredimo dve prilagoditvi:

1. Izbira parametrov in testiranje potekata nad vsemi podatki. Za zmanjšanje možnosti pretiranega prilagajanja uporabimo mrežno iskanje na podlagi rezultatov metode izloči enega (angl. leave-one-out).

2. Če dobimo več modelov z zelo podobnimi rezultati, izberemo preprostejšega (npr. tistega z največjim vplivom regularizacije). S tem dodatno zmanjšamo možnost pretiranega prilagajanja.

2.2.1 Metrike za ocenjevanje učenja

Za ocenjevanje posameznega koraka prečnega preverjanja in končno testiranje smo uporabili dve metriki in sicer klasifikacijsko točnost (CA) in površino pod krivuljo AUC.

Klasifikacijska točnost (CA)

Klasifikacijska točnost je najbolj preposta in intuitivna metrika za ocenjavanje klasifikacije. Pove nam delež pravilno klasificiranih primerov:

$$CA = \frac{N_p}{N}$$

N je število vseh klasificiranih primerov, N_p pa število pravilno klasificiranih primerov [6]. Za podatke z neuravnoteženo razredno porazdelitvijo klasifikacijska točnost ni dobra mera. V našem primeru je razredna porazdelitev precej uravnovešena.

Površina pod krivuljo (AUC)

Mera AUC je poleg CA najbolj popularna metrika za ocenjevanje binarnih klasifikatorjev. Predstavlja površino pod ROC (Receiver operating characteristic) krivuljo [3]. Primerom učne množice glede na vrednost razrednega atributa pravimo pozitivni ali negativni. Površina pod ROC krivuljo nam pove verjetnost, da bo klasifikator naključnemu pozitivnemu primeru napovedal višjo verjetnost za pozitivni razred kot naključnemu negativnemu primeru. Za razliko od CA je mera neodvisna od razredne porazdelitve. Večinski klasifikator dobi AUC oceno 0.5, idealni klasifikator pa oceno 1. Za ocene naših modelov zato pričakujemo vrednosti med tema dvema ekstremoma [6].

2.2.2 Prečno preverjanje

Prečno preverjanje je metoda s katero za učenje modelov in njihovo testiranje uporabimo vse podatke. V našem primeru bomo uporabili k-kratno prečno preverjanje, kjer primere učne množice najprej razdelimo na k delov [6]. Nato za vsakega od k delov učne množice naredimo naslednje:

1. Izbrani del uporabimo kot testno množico
2. Nad ostalimi deli množice naučimo model strojnega učenja ter ga ocenimo na testni množici.

Ocene vseh k modelov za pridobitev končne ocene preprosto povprečimo.

Omenjenjena je bila tudi metoda izloči enega (angl. leave-one-out). Ta metoda je posebni primer k-kratnega prečnega preverjanja, kjer je $k = n$, pri čemer je n število vseh primerov v učni množici. Torej je ob vsakem koraku metode izloči enega velikost testne množice 1, velikost učne pa n-*1*. Ta pristop je koristen, kadar imamo manjše število primerov. Za velike podatkovne množice bi namreč gradnja n modelov trajala predolgo časa [6].

V *scikit-learn* je prečno preverjanje implementirano v razredih `KFold` in `LeaveOneOut`, kot bljižnica pa obstaja tudi priročna funkcija `cross_validation` [8].

2.2.3 Mrežno iskanje optimalnih parametrov

Parametre, ki niso direktno naučeni med izvajanjem algoritma strojnega učenja (npr. vpliv regularizacije) lahko določimo tako, da poizkusimo več kombinacij le teh ter vsako kombinacijo ocenimo glede na rezultate prečnega preverjanja. Temu preprostemu postopku pravimo mrežno iskanje [8].

V našem primeru bomo uporabili najbolj osnoven tip mrežnega iskanja, kjer vse kombinacije parametrov, ki naj jih algoritem preizkusi, navedemo sami. Knjižnica *scikit-learn* ima točno v ta namen implementiran razred `GridSearchCV` [8]. Poleg kombinacij parametrov lahko podamo tudi parametre prečnega preverjanja in funkcijo, ki se bo uporabila za ocenjevanje posamičnega koraka prečnega preverjanja. Za neprofesionalne tekme smo uporabili 10-kratno prečno preverjanje, za profesionalne tekme pa metodo izloči enega. Za ocenjevanje smo uporabili CA.

2.3 Ocenjavanje atributov

Poleg gradnje in ocenjavanja klasifikatorjev želimo najti pomembne atribute z velikim vplivom na izid tekme. Vpliv vsakega atributa je mogoče oceniti z merami za ocenjevanje atributov.

Uporabili smo eno izmed preprostejših mer ocenjavanja atributov: informacijski prispevek. Kot pove ime, mera temelji na količini informacije. Vpeljimo notacije:

n - število učnih primerov,
$n_{k.}$ - število učnih primerov iz razreda r_k,
$n_{.j}$ - število učnih primerov z j-to vrednostjo danega atributa A,
n_{kj} - število učnih primerov iz razreda r_k in z j-to vrednostjo danega atributa A.

Vpeljimo naslednje deleže:

$$p_{kj} = n_{kj}/n$$

$$p_{k.} = n_{k.}/n$$

$$p_{.j} = n_{.j}/n$$

Vpeljimo naslednje entropije:

$$H_R = -\sum_k p_{k.} \log p_{k.}$$

$$H_A = -\sum_k p_{.j} \log p_{.j}$$

$$H_{RA} = -\sum_k \sum_j p_{kj} \log p_{kj}$$

Informacijski prispevek atributa A je definiran kot prispevek informacije atributa k določitvi vrednosti razreda[6]:

$$Gain(A) = H_R + H_A - H_{RA}$$

Vrednost razredne entropije H_R je za vse atribute enaka. Bolj kot je razredna porazdelitev učne množice enakomerna, večja bo vrednost te entropije. H_A predstavlja entropijo vrednosti danega atributa. Torej je odvisna od porazdelitve vrednosti atributa A v učni množici. H_{RA} pa je entropija produkta dogodkov razred-vrednost atributa. Ta entropija nam pove, kakšna je ob ločitvi učne množice glede na možne vrednosti danega atributa A, razredna porazdelitev podmnožic.

Informacijski prispevek v osnovni obliki deluje le nad diskretnimi atributi [6]. Ker v našem primeru delamo z zveznimi atributi, je te pred izračunom informacijskega prispevka potrebno pretvoriti v diskretne. Eden od pristopov je, da vrednosti zveznega atributa razdelimo na dva dela, mejo pa postavimo tako, da bo informacijski prispevek atributa največji.

Mera ima dve slabosti. Prva je, da precenjuje večvrednostne atribute. V našem primeru se lahko temu problemu z zgoraj opisano diskretizacijo izognemo. Druga slabost informacijskega prispevka je njegova kratkovidnost. Nekateri atributi so pomembni šele, ko jih gledamo v kombinaciji z drugimi atributi. Te interkacije informacijski prispevek ne upošteva [6].

Uporabili smo implementacijo informacijskega prispevka v orodju *Orange*. *Orange* je odprtokodno orodje za vizualizacijo ter podatkovno analizo [1].

Poglavje 3

Podatki

V tem poglavju opišemo kje in kako smo pridobili podatke ter kako smo jih obdelali v obliko, ki je primerna za strojno učenje.

3.1 Neprofesionalne tekme

Za pridobivanje neprofesionalnih tekem smo uporabili paket *RiotWatcher*[1]. Paket nudi funkcije v programskem jeziku Python za pridobivanje podatkov z uradne spletne knjižnice *Riot Games API for League of Legends*[2]. Funkcije vrnejo normalne slovarje programskega jezika Python. Glavne funkcije, ki smo jih uporabili za pridobivanje podatkov, so opisane v tabeli 3.1. Primer uporabe paketa v interaktivni konzoli programskega jezika Python:

```
from riotwatcher import RiotWatcher
watcher = RiotWatcher(RIOT_GAMES_API_ID)
match_id = 2381146548
print(watcher.get_match(match_id))
{u'queueType': u'RANKED_SOLO_5x5', ...}
```

[1] http://github.com/pseudonym117/Riot-Watcher

[2] https://developer.riotgames.com

Tabela 3.1: Uporabljene funkcije RiotWatcher.

ime funkcije	**vhodni podatki**	**izhodni podatki**
`get_matchlist`	identifikacijska številka igralca	zgodovina tekem
`get_match`	identifikacijska številka tekme	podrobnosti tekme
`get_stats`	identifikacijska številka igralca	celoletne agregirane statistike igralca

Za začetek smo potrebovali nekaj identifikacijskih številk igralcev. Pridobili smo jih z javno dostopne lestvice najboljših igralcev. S funkcijo `get_matchlist` smo nato pridobili zgodovino zadnjih 15 tekem vsakega igralca. Ker s tem še nismo imeli dovolj podatkov za nadaljevanje smo uporabili metodo `get_match` za podrobnosti vsake posamične tekme. Za zgodovinske agregirane statistike vsakega posamičnega igralca, ki je sodeloval v tej tekmi, smo uporabili še funkcijo `get_stats`. Za vse udeležence ene tekme smo s tem pridobili tudi njihove identifikacijske številke. Z novo pridobljenimi identifikacijskimi številkami smo nato lahko celoten opisan postopek ponovili ter tako pridobili nove tekme. Seveda smo se morali prepričati, da nismo večkrat pridobili podatke o istih tekmah. Na tak način smo pridobili podatke za 4000 tekem. Opisan postopek predstavlja glavni del spisane Python skripte[3]. Ta vsebuje veliko obravnavanja možnih napak, saj se je program moral zaradi omejitve števila klicev spletne knjižnice izvajati več dni zapored.

[3] `http://github.com/simejanko/diplomska-lol/blob/master/get_data.py`

3.1.1 Gradnja učne množice

Za pridobitev urejene učne množice smo morali prvotne podatke še kar precej obdelati. V ta namen smo spisali Python skripto[4], ki opravi to nalogo.

Primeri v učni množici so neprofesionalne tekme. Za vsakega udeleženca dane tekme imamo agregirane celoletne statistike. Pomembnejši primeri takih statistik:

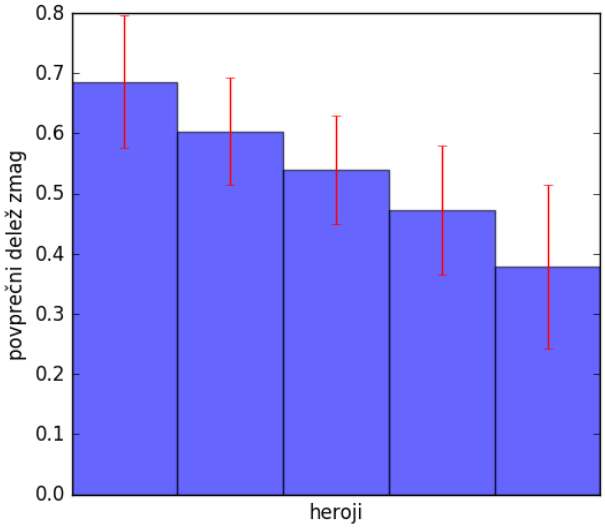
- število odigranih tekem: dober pokazatelj izkušenosti ter koristen za združevanje z drugimi atributi,
- število zmag,
- število ubojev: kolikokrat je igralec skozi celo leto uspešno ubil nasprotnikovega heroja,
- število smrti,
- število asistenc: kolikokrat je igralec skozi celo leto pomagal pri uboju nasportnikovega heroja.

Najprej se moramo prepričati, da statistike igralcev zajemajo le stanje, kakršno je bilo pred dano tekmo. Le tako namreč lahko kasneje resnično ovrednotimo modele strojnega učenja. Pri prvem koraku obdelave podatkov smo torej od celoletnih statistik igralcev odšteli statistike vseh tistih tekem, ki so se zgodile po trenutno opazovani tekmi. Pri vseh tekmah imamo tudi podatek o času začetka, zato ni bilo težko najti novejših tekem.

Poleg celoletnih statistik imamo za vsakega udeleženca tudi podatke o zadnjih desetih tekmah, odigranih pred opazovano tekmo. Drugi korak obdelave podatkov je torej izračun agregiranih statistik za teh deset tekem. S tem pridobimo še informacije o tem kako dobro je šlo igralcu v bližnji preteklosti.

[4] `http://github.com/simejanko/diplomska-lol/blob/master/atribute_extraction_relative_attrs.py`

Vsak igralec sam izbere heroja, ki ga bo igral. Na tabeli 1.1 opazimo, da so si heroji med seboj zelo različni. To pomeni, da igralcu določeni heroji bolj ležijo, kot drugi. To prikazuje slika 3.1. Ker izbire herojev poznamo preden se tekma začne, lahko to informacijo izkoristimo. Celoletne statistike igralca lahko pridobimo le za določenega heroja. Prav tako lahko pridobimo zadnjih 10 tekem (časovno pred opazovano tekmo), v katerih je igralec igral enakega heroja kot pri opazovani tekmi.



Slika 3.1: Povprečni deleži zmag najbolj igranih herojev in standardni odkloni. Za vsakega igralca smo dobili pet herojev, ki jih je igral največkrat. Nato smo izračunali deleže zmag tega igralca, ko je igral vsakega od petih herojev in jih razporedili od največjega do najmanjšega. Dobljenih pet vrednosti vseh igralcev smo nato povprečili in dobili zgornje vrednosti in standardne odklone.

Da povzamemo, po do sedaj opisanih korakih imamo agregirane statistike igralcev, izračunane v štirih različnih obdobjih ter pogojih:

- od začetka sezone do opazovane tekme (celoletne),
- 10 tekem pred opazovano tekmo,
- od začetka sezone do opazovane tekme za izbranega heroja (celoletne),
- 10 tekem pred opazovano tekmo z izbranim herojem.

V naslednjem koraku obdelave podatkov združimo atribute, kjer je to smiselno. Nekateri primeri združevanj so navedeni spodaj.

- Skoraj vse statistike je smiselno deliti s številom odigranih tekem (npr. bolj informativen kot število zmag je delež zmag).
- Namesto ubojev, smrti in asistenc uporabimo v domenah videoiger dobro poznano KDA razmerje:

$$KDA = (uboji + asistence)/smrti \tag{3.1}$$

- Uboje in asistence lahko združimo tudi kot sodelovanje igralca pri ubojih (angl. kill participation), ki je za eno tekmo izračunano po naslednji enačbi:

$$KP = (uboji + asistence)/uboji\ celotne\ ekipe \tag{3.2}$$

V poglavju 1.1 smo že omenili, da en igralec pripada rdeči ali modri ekipi. Namesto atributov vsakega posameznega igralca je torej smiselno, da izračunamo atribute vsake ekipe. Prej pridobljene atribute preprosto povprečimo med petimi igralci, ki pripadajo isti ekipi.

Podatke, pridobljene pri prejšnjem koraku, bi že lahko uporabili za učenje. Vendar v resnici nas bolj kot absolutne agregirane statistike za vsako posamično ekipo zanimajo relativne vrednosti. Te izračunamo tako, da sorodne atribute obeh ekip odštejemo. Vsak atribut torej predstavlja razliko neke statistike med modro in rdečo ekipo. Vse pridobljene atribute smo normalizirali tako, da so njihove vrednosti na intervalu $[-1, 1]$.

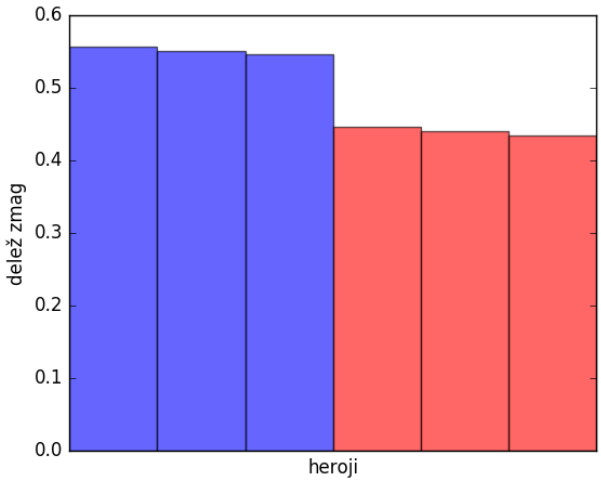
Končno število atributov je 50 (tabela A.2). Za strojno učenje manjkata le še razredni atribut ter zapis podatkov v primernem formatu. Prvega preprosto pridobimo iz podatkov opazovane igre in lahko zaseda vrednosti modra ali rdeča. Podatke pa smo se odločili zapisati v zapisu, kjer so vrednosti med seboj ločene s tabulatorjem.

3.1.2 Ekipna kompozicija

Pokazali smo že, da igralcem nekateri heroji bolj ležijo kot drugi. Nekateri heroji pa tudi v splošnem veljajo za močnejše. To opazimo tudi s slike 3.2. Poleg tega so tudi določene kombinacije herojev v ekipi močnejše kot druge. Razmerja moči herojev se lahko spreminjajo z vsako posodobitvijo igre, ki se običajno zgodi na vsaka dva tedna, zato izračun moči herojev iz celoletnih statistik vseh igralcev ne pride v upoštev. Ker je v igri na izbiro več kot 100 herojev, naših 4000 tekem ne bo dovolj za zanesljiv izračun teh statistik (nekateri heroji so redko igrani), poleg tega pa za pravične ocene napovedi v vsakem primeru nočemo uporabiti istih iger, ki jih bomo kasneje napovedovali.

Zaradi vseh teh razlogov smo pridobili ločeno množico tekem pri katerih so nas zanimali le izbrani heroji ter zmagovalci. Ostalih statistik za te igre nismo potrebovali, kar je bistevno znižalo število klicev spletne knjižnice in nam omogočilo pridobiti 150.000 tekem, ki so se zgodile pred prej pridobljenimi 4000 tekmami. Na podlagi teh tekem smo nato izračunali deleže zmag vseh herojev. Na ta način je pridobljena slika 3.2. Šli smo še korak dlje ter izračunali deleže zmag vseh pojavljenih kombinacij herojev (pari, trojke, ...).

Za vsako ekipo od prej pridobljih 4000 tekem smo nato na podlagi teh izračunov pridobili pet atributov: povprečni delež zmag izbranih herojev ekipe, povprečni delež zmag izbranih parov herojev ekipe, ..., povprečni delež zmag izbrane celotne ekipne kompozicije (pet herojev). Tudi te atribute smo spremenili v pet relativnih atributov, ki so bili dodani prvotnim. Skupno število atributov učne množice po tej transformaciji je bilo potem 55.



Slika 3.2: Trije najvišji (modra barva) ter trije najnižji (rdeča barva) heroji glede na delež zmag. Izračunano na ločenih podatkih 150.000 tekem za večjo zanesljivost.

3.2 Profesionalne tekme

Spletna knjižnica, ki smo jo uporabili za pridobivanje neprofesionalnih tekem, ne vključuje dostopa do profesionalnih tekem. Posledničnо smo se morali zateči k filtriranju vsebin spletnih strani, ki vsebujejo rezultate tekem. Odločili smo se za popularno spletno stran Gamepedia[5]. Na strani so v dokaj čisti tabelarični obliki objavljeni rezultati, statistike in časovne znamke posamičnih tekem iz različnih tekmovanj. Omejili smo se na najbolj prestižno korejsko ligo: League Champions Korea. Izmed vseh lig, ki se odvijajo v posamičnih regijah sveta, ravno ta velja za najboljšo. Pridobivanje tekem smo s pomočjo ogrodja *Scrapy*[6] implementirali kot programskega pajka[7]. *Scrapy* je odprtokodno Python ogrodje za pridobivanje podatkov s spletnih strani. Pajek na spletni strani poišče tabele, ki predstavljajo posamične tekme. Ko tako tabelo najde iz nje iz ustreznih okenc prebere različne statistike ter jih zapiše v datoteko v zapisu JSON. Na tak način smo pridobili podatke za 300 profesionalnih tekem.

[5] `http://gamepedia.com`

[6] `http://scrapy.org/`

[7] `http://github.com/simejanko/diplomska-lol/tree/master/scrap_esports_data/scrap_esports_data`

3.2.1 Gradnja učne množice

Za pridobitev urejene učne množice je bilo potrebno prvotne podatke dodatno obdelati. V ta namen smo spisali Python skripto[8], ki opravi to nalogo.

Primeri v učni množici so profesionalne tekme. Delno je procesiranje profesionalnih tekem zelo podobno procesiranju neprofesionalnih (običajnih) tekem:

- Za vsako tekmo moramo upoštevati le statistike ekip, ki so bili znane preden se je ta tekma odvila. Le tako lahko kasneje resnično ovrednotimo model.
- Poleg statistik za celotno znano zgodovino ekipe ločeno izračunamo tudi statistike za zadnjih nekaj (4) tekem. S tem pridobimo informacijo o nedavni pripravljenosti ekipe.
- Na podlagi preteklih tekem v katerih so igralci izbrali enake heroje kot za dano tekmo, izračunamo dodatne statistike.
- Združimo atribute, kjer je to primerno.
- Namesto ločenih atributov za vsako ekipo, uporabimo relativne atribute.
- Atribute normaliziramo tako, da so njihove vrednosti na intervalu $[-1, 1]$.

Opazimo, da atribute igralcev, ki pripadajo enaki ekipi, tu nismo povprečili. Pri profesionalnih tekmah imamo namreč tudi informacijo o poziciji igralcev. Razlike med vsemi pozicijami smo opisali v poglavju 1.1. Pri profesionalnih tekmah en igralec običajno vedno igra enako pozicijo. Torej bomo namesto povprečnih atributov igralcev gledali atribute vsake pozicije posebaj.

[8] `http://github.com/simejanko/diplomska-lol/blob/master/scrap_esports_data/atribute_extraction_esports.py`

Pri profesionalnih tekmah lahko izkoristimo še dejstvo, da sta se 2 ekipi v preteklosti lahko že srečali. Iz preteklih srečanj ekip izračunamo še en nabor statistik. V primeru, da je dana tekma prvo srečanje dveh ekip, uporabimo smiselne privzete vrednosti (npr. za delež zmag bi bila smislna vrednost 50%). Končno število atributov profesionalih tekem je 82. Vse atribute navaja tabela A.4.

Poglavje 4

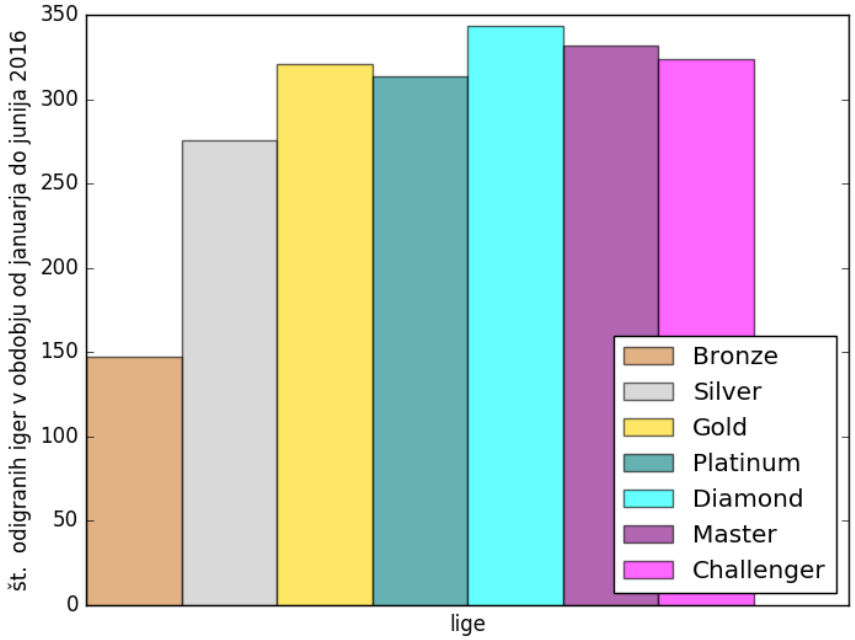
Analiza podatkov

Kljub temu, da je glavni cilj diplomske naloge napovedovanje izzidov iger, pridobiljeni podatki ponujajo priložnost za dodatno kratko analizo. Neprofesionalne tekme, ki smo jih pridobili so v bistvu oddigrane v zvrščevalnem načinu igre. To pomeni, da igra vsekega igralca glede na njegov uspeh zvrsti v eno izmed sedmih lig. Najslabša liga je imenovana *Bronze*, najboljša pa *Challenger*. Vredno je omeniti, da mora igralec, preden je lahko razvrščen, oddigrati kar veliko tekem (povprečno 500) v nezvrščevalnem (normalnem) načinu igre. To pomeni, da med razvrščenimi igralci ni začetnikov. Ker smo pridobili podatke za tekme vseh lig, z analizo v tem poglavju želimo primerjati povprečne statistike igralcev, ki pripadajo posamičnim ligam ter prikazati zanimivejše vizualizacije. Želimo odkriti vplivne statistike, ki bi jih igralci nižjih lig lahko izboljšali brez ogromnega napora. Ta analiza za samo napovedovanje iger ne predstavlja koristi, saj se v igrah, ki jih napovedujemo, med seboj spopadajo igralci enakih lig.

Napisali smo ločen program[1] za pridobivanje atributov, ki zapiše v vsako vrstico statistike in ligo enega igralca. Število zapisanih igralcev je približno 40.000. Za filtriranje in računanje povprečij smo uporabili knjižnico *NumPy*,

[1] `http://github.com/simejanko/diplomska-lol/blob/master/good_player_analysis/atribute_extraction.py`

za vizualizacijo pa knjižnico *matplotlib*.

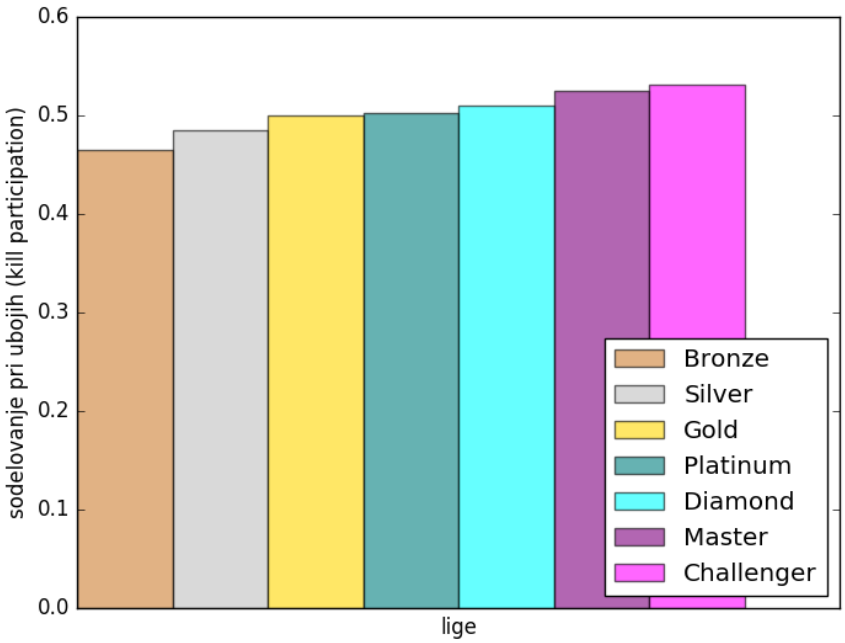


Slika 4.1: Povprečno število odigranih tekem v obdobju od januarja do junija leta 2016. Vsak stolpec predstavlja povprečno vrednost za eno ligo.

Rezultati s slike 4.1 so večinoma zelo pričakovani. Boljši igralci igrajo pogostoje. Zanimiv je padec vrednosti v zadnjih dve ligah. Tu najdemo le peščico igralcev. V ligi *Master* je 0.06% igralcev, v *Challenger* pa 0.01% igralcev . Tu je govora o porazdelitvi vseh igralcev[2] in ne le teh, ki smo jih zbrali. V naših podatkih smo med 40.000 igralci uspeli zbrati približno 1000 igralcev iz najvišjih dveh lig, kar tudi predstavlja vse igralce v teh dveh ligah na strežniku s katerega smo dobili podatke (zahodno-evropejeski strežnik). Problem, ki ga predstavlja tako nizko število igralcev, je čas čakanja na novo tekmo. Igra poizkuša za eno tekmo najti igralce v enakih/podobnih ligah. Za

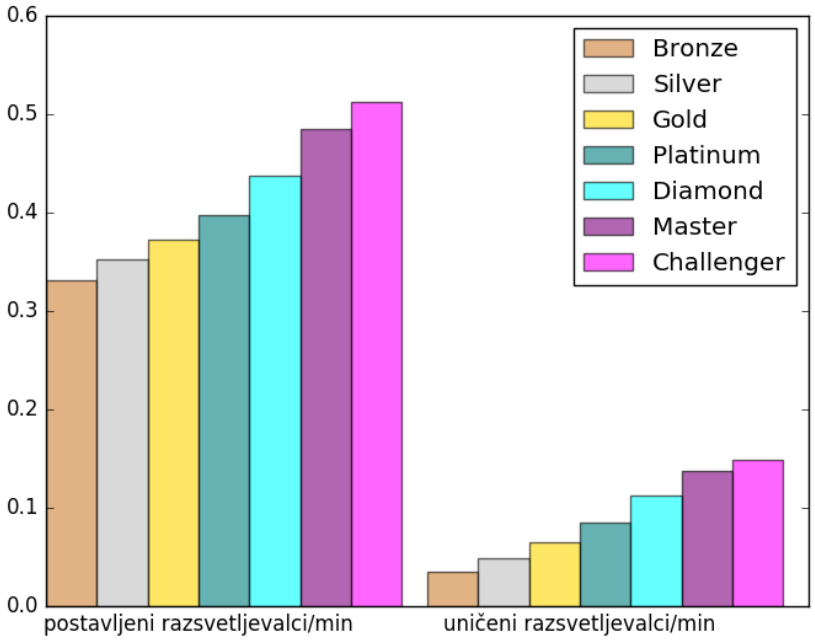
[2] `http://na.op.gg/statistics/tier`

igralce najvišjih dveh lig to pomeni od 15 do 30 minut čakanja na novo tekmo, včasih pa je čakalni čas tudi do 60 minut. Zaradi tega razloga se nekateri igralci odločajo za igranje z drugimi uporabniškimi računi, kjer lahko igrajo v nekoliko nižjih ligah. Temu bi torej lahko prepisali padec vrednosti zadnjih dveh lig na grafu 4.1.



Slika 4.2: Povprečno sodelovanje pri ubojih, ki smo ga izračunali po enačbi 3.2.

Na sliki 4.2 opazimo, da igralci višjih lig sodelujejo pri večjem deležu ubojev svoje ekipe. To pomeni, da se igralci višjih lig več časa premikajo v skupinah namesto, da igra vsak po svoje.



Slika 4.3: Povprečno število postavljenih in uničenih razsvetljevalcev na minuto.

Najbolj zanimiv rezultat pokaže slika 4.3, saj so razlike med ligami velike, hkrati pa predstavlja dve statistiki, za izboljšavo katerih igralci nižjih lig ne bi potratili veliko časa. Statistiki predstavljata količino odkrivanja informacij nasprotnika ter skrivanja svojih informacij. Ta koncept in objekt razsvetljevalca smo opisali v poglavju 1.1.

V nižjih ligah igralci nočejo investirati pridobljene igralne valute za dodatne razsvetljevalce ali pa za uničenje nasprotnikovih, čeprav so relativno zelo poceni. Zgodi se tudi, da pozabijo izkoristiti razsvetljevalce, ki so jim dani zastonj. V vsakem primeru povišanje teh dveh statistik za igralce nižjih lig ne bi bila zahtevna naloga.

Poglavje 5

Rezultati

Nad podatki opisanimi v poglavju 3 smo izvedli metode opisane v poglavju 2. V tem poglavju predstavimo in analiziramo rezultate teh metod.

5.1 Neprofesionalne tekme

Rezultate neprofesionalnih tekem smo primerjali z rezultati soležnih metod, ki so bile uporabljene v obstoječem članku opisanem v poglavju 1.2. Rezultate prikazuje tabela 5.1.

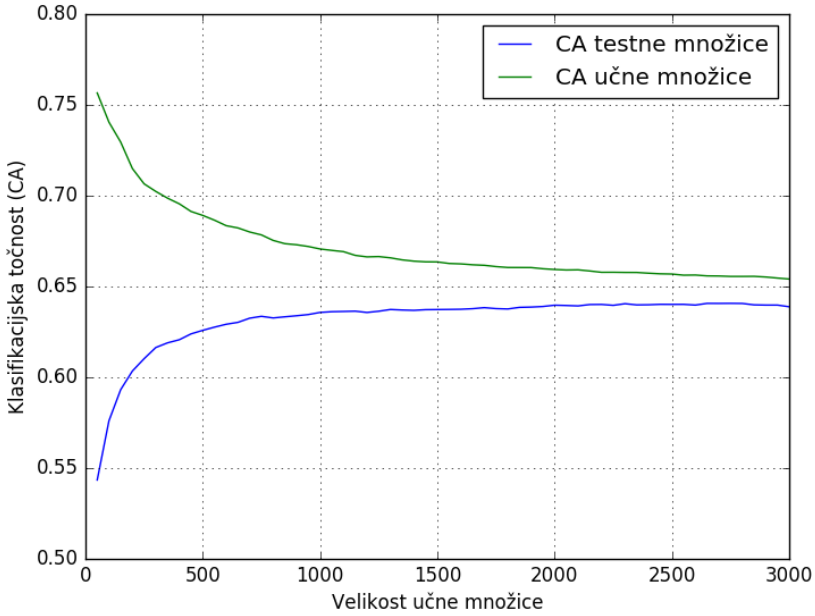
Tabela 5.1: Rezultati strojnega učenja za napoved izida neprofesionalnih tekem. Metrike so bile izračunane na testni množici. ΔCA predstavlja primerjavo z rezultati soležnih metod z obstoječega članka [4].

metoda strojnega učenja	**AUC**	**CA**	**ΔCA**
Večinski klasifikator	0.5	50,89%	/
Logistična regresija	0.7022	64.13%	+2.94%
SVM	0.7019	64.33%	+2.14%
Naključni gozdovi	0.6906	64.92%	+2.54%
Zlaganje	0.7031	65.22 %	+0.41%

Vsi testirani modeli (z očitno izjemo večinskega klasifikatorja) so po ocenah zelo blizu. Najbolje je ocenjena metoda zlaganja, od posamičnih metod pa so po klasifikacijski točnosti najbolje ocenjeni naključni gozdovi, po AUC pa logistična regresija. Dobili smo malenkost boljše rezultate od obstoječega članka, kjer je prav tako najbolje ocenjena metoda zlaganja. Razlika je, da so z zlaganjem združili pet klasifikatorjev [4].

Za logistično regresijo smo izrisali tudi učno krivuljo na sliki 5.1. Učna krivulja je koristna in intiutivna metoda za analizo rezultatov strojnega učenja. Z nje lahko do neke mere razberemo, česa se lotiti ali česa se ne lotiti, če želimo izboljšati natančnost modela. Pridobimo jo tako, da eno metodo strojnega učenja uporabimo in ocenimo pri različnih velikostih učne množice. Začnemo z majhnim številom učnih primerov, končamo pa ko smo uporabili celotno učno množico. Pri vsakem koraku ocenimo natančnost modela nad uporabljeno učno množico in ločeno testno množico, katero moramo določiti pred opisanim postopkom.

Ko se bližamo uporabi celotne učne množice na grafu 5.1 opazimo, da je CA na testni in učni množici skoraj enak. To pomeni, da naš model ni pretirano prilagojen učnim podatkom in da pridobivanje večje množice podatkov ne bi pripomoglo k natančnosti našega modela. Za zvišanje natančnosti bi morali torej razmišljati o spremembi obstoječih atributov ali o dodajanju novih.



Slika 5.1: Učna krivulja logistične regresije. Vrednosti na abcisi predstavljajo število primerov uporabljenih pri strojnem učenju, na ordinati pa natančnost naučenega modela. Zelena črta predstavlja natančnost izračunano na učni množici, modra črta pa na testni množici. Uporabili smo testno množico, ki smo jo pridobili pri postopku opisanem v poglavju 2.2.

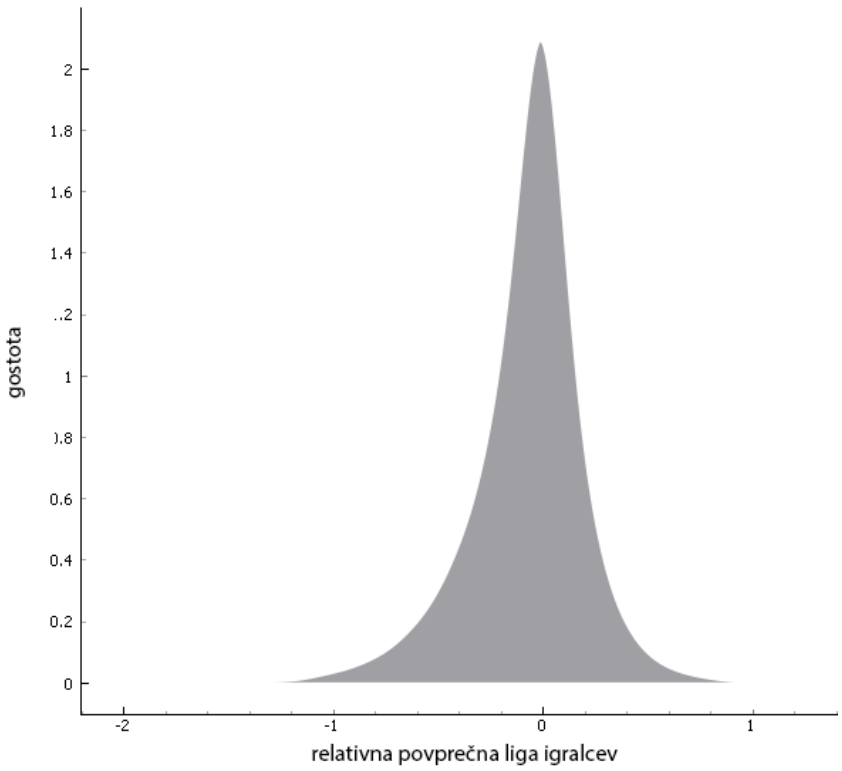
Tabela 5.2: Atributi neprofesionalnih tekem z navišjim informacijskim prispevkom. Imena atributov so podana za eno ekipo. Končni atributi so po postopku opisanem v poglavju 3 odšteti med ekipama (relativna vrednost).

atribut	**inf. prispevek**
Povprečna liga igralcev	0.044
Delež zmag parov herojev (atr. kompozicije - 3.1.2)	0.011
Delež zmag posamičnih herojev (atr. kompozicije - 3.1.2)	0.011
Povprečni KDA (formula 3.1) z izbranim herojem	0.009
Povprečno št. uničenih zgradb na igro	0.009

V tabeli 5.2 je opazno napomembnejši atribut povprečna liga igralcev. Čeprav igra poizkuša za eno tekmo najti igralce, ki so v enakih ligah, to ni vedno izvedljivo, saj igralci na novo tekmo nočejo čakati predolgo časa. Zato se pojavijo razlike v povprečni ligi obeh ekip. Porazdelitev te razlike prikazuje slika 5.2. Opazimo, da obstajajo celo tekme, kjer je razlika povprečne lige med ekipama 1. Tu gre najbrž za šum, saj pri določenih tekmah nismo morali pridobiti lige za vse pristojne igralce, zato ekipno povprečje ni vedno predstavljeno z vsemi petimi člani. Primer so tekme z igralci, ki še niso uvrščeni v nobeno ligo. Igralec ligi namreč pripada šele po prvih desetih odigranih tekmah v zvrščevalnem načinu igre (angl. ranked mode).

Pomembna sta tudi prva dva od petih atributov ekipne kompozicije opisanih v poglavju 3.1.2. Tudi za deleže zmag trojk herojev je informacijski prispevek relativno visok (0.006), pri daljših kombinacijah ekipne kompozicije pa precej pade.

Da je razmerje KDA med najbolje ocenjenimi atributi nas ne presenti, saj je dober pokazatelj mehaničnih sposobnosti igralca. Povprečno število uničenih zgradb pa je po drugi strani eden izmed pokazateljev ekipnega dela ter strategije.



Slika 5.2: Porazdelitev razlike povprečnih lig igralcev med ekipama.

5.2 Profesionalne tekme

Rezultate profesionalnih tekem smo primerjali z rezultati soležnih metod, ki so bile uporabljene za napoved neprofesionalnih tekem, na tabeli 5.1. Rezultate prikazuje tabela 5.3.

Tabela 5.3: Rezultati strojnega učenja za napoved izida profesionalnih tekem. Metrike so bile izračunane na podlagi metode izloči enega. ΔAUC in ΔCA prestavljata primerjavo z rezultati soležnih metod na tabeli 5.1.

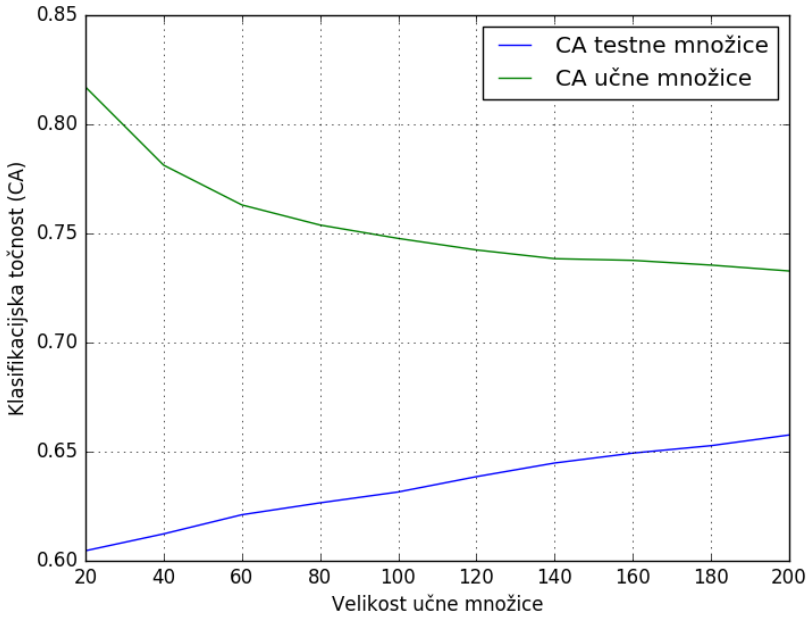
metoda strojnega učenja	**AUC**	**ΔAUC**	**CA**	**ΔCA**
Večinski klasifikator	0.5	0	55,03%	+4.14%
Logistična regresija	0.7115	+0.0093	69.80%	+5.67%
SVM	0.7302	+0.0283	70.46%	+6.13%
Naključni gozdovi	0.7255	+0.0319	70.81%	+5.89%
Zlaganje	0.7343	+0.0312	71.14%	+5.92%

Vsi testirani modeli (z očitno izjemo večinskega klasifikatorja) so po ocenah zelo blizu. Najbolje je ocenjena metoda zlaganja, od posamičnih metod pa so po CA najbolje ocenjeni naključni gozdovi, po AUC pa SVM.

Nekoliko boljši modeli so bili za profesionalne tekme pričakovani, saj so tako kot pri drugih športih določene profesionalne ekipe konsistentno boljše od drugih, medtem ko pri neprofesionalnih tekmah igra teži k temu, da med seboj igrajo čimbolj enakovredni igralci.

Opazimo, da so profesionalne tekme nekoliko manj razredno uravnovešene. Kar 55% tekem zmaga ekipa na modri strani. To nesorazmerje je na profesionalni sceni igre že nekaj časa dobro poznano [7]. Ena izmed bistevnih razlik med modro in rdečo ekipo je izbira herojev. Prednost modre ekipe pri izbiri herojev smo že omenili v pogavlju 1.1. Manjše razlike se pojavijo tudi na igralnem polju, a podrobna analiza tega ni cilj diplomske naloge. Iz primerjave z neprofesionalnimi tekmami je jasno, da te manjše prednosti večinoma znajo izkoristiti le profesionalne ekipe.

Tudi za profesionalne tekme smo izrisali učno krivuljo na sliki 5.3. Za izris te smo bili primorani uporabiti 100 primerov kot testno množico. Opazimo, da bi potencialno lahko z pridobitvijo več primerov še za malenkost zvišali natančnost modela.



Slika 5.3: Učna krivulja logistične regresije za profesionalne tekme. Os x predstavlja število primerov uporabljenih pri strojnem učenju, y os pa natančnost naučenega modela. Zelena črta predstavlja natančnost izračunano na učni množici, modra črta pa na testni množici.

Tabela 5.4: Atributi profesionalnih tekem z navišjim informacijskim prispevkom. Imena atributov so podana za eno ekipo. Končni atributi so po postopku opisanem v poglavju 3 odšteti med ekipama (relativna vrednost).

atribut	**inf. prispevek**
KDA igralca na podporni poziciji (support)	0.135
KDA igralca na zgornji poziciji (top)	0.115
Delež zmag ekipe	0.115
KDA igralca na nosilni poziciji (carry)	0.108
KDA igralca na srednji poziciji (middle)	0.108

Tudi pri profesionalnih igrah je KDA (enačba 3.1) pomemben atribut. Lahko vidimo celo razporeditev pomembnosti pozicij. Ni presenečenje, da je delež zmag tu bolj informativen, kot pri neprofesionalnih igrah. Pri neprofesionalnih tekmah nam igra namreč glede na naše zmage daje vedno močnejše/slabše nasprotnike, zato delež zmag ne pomeni veliko.

Poglavje 6

Sklepne ugotovitve

Glavni cilj diplomskega dela je bilo oceniti uporabnost različnih tehnik strojnega učenja za napovedovanje izida League of Legends tekem. Za dosego tega cilja smo pridobili podatke o tekmah ter izračunali značilke, ki dobro opišejo igralce in so primerne za strojno učenje. Nato smo nad pridobljenimi podatki pognali več algoritmov strojnega učenja: logistično regresijo, metode podpornih vektorjev in naključne gozdove. Vse tri metode so dosegle primerljive napovedne točnosti, najbolje pa se je odrezala metoda zlaganja klasifikatorjev, ki združuje napovedi vseh treh uporabljenih algoritmov.

Rezultate smo primerjali z rezultati iz nedavno objavljene študije [4] v kateri poizkušajo napovedovati izide neprofesionalnih tekem. V primerjavi s to študijo naša učna množica vsebuje veliko več atributov. Dodatne atribute v precejšni meri predstavljajo statistike izračunane na nedavni zgodovini igralcev. Kljub dodatnim atributom smo izide tekem napovedovali le z malenkost boljšo natančnostjo, kjer je naš najboljši rezultat CA = 65.22% rahlo presegel rezultat CA = 64.81% iz študije. Članek smo razširili še z napovedovanjem profesionalnih tekem. Tehnike strojnega učenja so nad tem tipom tekem dosegle višjo natančnost, naprimer CA = 71.14%. Omenimo naj, da študija [4] ne analizira te vrste podatkov.

Diplomsko nalogo bi lahko še dodatno izboljšali oziroma razširili. Za profesionalne tekme smo s pomočjo učne krivulje ugotovili, da bi z večjo učno množico potencialno lahko še nekoliko izboljšali napovedno natančnost. Pri neprofesionalnih tekmah bi bilo zanimivo upoštevati kvaliteto internetne povezave igralcev. Ta namreč vpliva na odzivni čas igralca. Poleg tega pa lahko vsaka odsotnost igralca zaradi težav z internetno povezavo vpliva na izid tekme. Zanimivo bi bilo tudi napovedovanje izida tekem po petih, desetih in dvajsetih pretečenih minutah ter primerjava rezultatov.

Literatura

[1] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[2] Arpad Elo. *The Rating of Chess Players, Past and Present*. Arco Publishing, 1978.

[3] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[4] Colin Feo, Jeremy Ma, and Allen Sirolly. Predicting winning teams in League of Legends. *Journal of Machine Learning Research*, 32, 2014.

[5] Riot Games. League of Legends new player guide. `http://gameinfo.eune.leagueoflegends.com/en/game-info/get-started/new-player-guide/`. [Online; dostopano 1.6.2016].

[6] Igor Kononenko and Marko Robnik Šikonja. *Inteligentni sistemi*. Fakulteta za računalništvo in informatiko, 2010.

[7] Ferguson Mitchell. Blue wins more than red in 'League of Legends': The stats. `http://www.dailydot.com/esports/league-of-legends-red-blue-statistics-win-rate/`. [Online; dostopano 13.6.2016].

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[10] Pu Yang, Brent Harrison, and David L Roberts. Identifying patterns in combat that are predictive of success in moba games. *Proceedings of Foundations of Digital Games*, 2014.

Dodatek A

Opis atributov

V tem dodatku navedemo in opišemo atribute, ki smo jih uporabili pri napovedovanju izidov neprofesionalnih in profesionalnih tekem.

A.1 Neprofesionalne tekme

Večina statistik pri neprofesionalnih tekmah se izračuna pod štirimi različnimi pogoji. To pomeni štiri ločene atribute. Da ne bomo vseh statistik večkrat naštevali, smo navedbo atributov razdelili na tabelo pogojev atributov (A.1) in tabelo atributov (A.2).

Tabela A.1: Opis pogojev atributov za neprofesionalne tekme. Večino statistik na tabeli A.2 se izračuna pod vsakim zgornjim pogojem posebaj, kar pomeni štiri različne atribute.

pogoj	**opis**
nedavno	zadnjih 10 tekem igralca
celoletno	celotna sezona igralca
nedavno z izbranim herojem	zadnjih 10 tekem, kjer je igralec izbral enakega heroja
celoletno z izbranim herojem	tekme celotne sezone, kjer je igralec izbral enakega heroja

Tabela A.2: Atributi neprofesionalnih tekem. Vsi atributi so zvezni. Z izjemo atributov ekipne kompozicije so opisane statistike za enega igralca. Končni atributi so povprečeni med igralci ter odšteti od atributov nasprotne ekipe. Večina atributov obstaja v štirih različicah opisanih v tabeli A.1. Za atribute kompozicije to ne velja.

opis atributa
razmerje seštevka ubojev in asistenc ter smrti (KDA)
razmerje sodelovanja pri ubojih ekipe
delež zmaganih tekem
število odigranih tekem
povprečno število ubitih nasprotnikovih pošasti na tekmo
povprečno število ubitih nevtralnih pošasti na tekmo
povprečno število uničenih zgradb na tekmo
povprečna količina pridobljenega denarja na tekmo
razmerje med storjenimi ter prejetimi točkami škode
liga igralca
končna liga igralca v prejšnjem letu/sezoni
povprečno število postavljenih razsvetljevalcev na minuto igranja
povprečno število uničenih razsvetljevalcev na minuto igranja
povprečno število pridobljenih točk izkušenj na minuto igranja
število tekem v katerih je bil igralec na enaki poziciji (nezanesljivo)
povprečni globalni delež zmag izbranih herojev (ekipna kompozicija)
povprečni globalni delež zmag izbranih parov herojev (ekipna kompozicija)
povprečni globalni delež zmag izbranih trojk herojev (ekipna kompozicija)
povprečni globalni delež zmag izbranih četverk herojev (ekipna kompozicija)
globalni delež zmag celotne izbrane kompozicije (ekipna kompozicija)

A.2 Profesionalne tekme

Polovica statistik pri profesionalnih tekmah se izračuna pod tremi različnimi pogoji. To pomeni tri ločene atribute. Da ne bomo vseh statistik večkrat naštevali, smo navedbo atributov razdelili na tabelo pogojev atributov (A.3) in tabelo atributov (A.4). Druga polovica atributov, ki jim pravimo pozicijski atributi, pa se poleg teh treh pogojev izračuna še za vsako od petih možnih pozicij igralcev, ki so opisane v poglavju 1.1. Pozicijski atributi torej obstajajo v petnajstih različicah.

Tabela A.3: Opis pogojev atributov za profesionalne tekme. Večino statistik na tabeli A.4 se izračuna pod vsakim zgornjim pogojem posebaj, kar pomeni tri različne atribute.

pogoj	**opis**
nedavno	zadnje 4 tekme ekipe
celotno	celotna znana zgodovina ekipe
pretekla srečanja	pretekle tekme, kjer sta se spopadli isti ekipi

Tabela A.4: Atributi profesionalnih tekem. Vsi atributi so zvezni. Opisane so statistike za eno ekipo. Končni atributi so odšteti od atributov nasprotne ekipe. Nepozicijski atributi obstajajo v treh različicah opisanih v tabeli A.3. Pozicijski atributi obstajajo v petnajstih različicah - za vsako od petih pozicij, ki so opisane v poglavju 1.1, v treh pogojih opisanih v tabeli A.3.

opis atributa	**pozicijski atribut**
razmerje seštevka ubojev in asistenc ter smrti (KDA)	da
razmerje sodelovanja pri ubojih ekipe	da
delež zmaganih tekem	ne
delež zmaganih tekem na modri strani	ne
delež zmaganih tekem na rdeči strani	ne
delež zmaganih tekem izbranega heroja	da
globalen (vse ekipe) delež zmaganih tekem izbranega heroja	da
povprečno trajanje zmaganih iger	ne
povprečno trajanje izgubljenih iger	ne
povprečno število ubitih pošasti na minuto igranja	da