FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Semestrální práce z KIV/DBM2

Datová analýza

Predikce výhry a doby trvání World of Warcraft Battlegroundu na základě historie statistik hráčů

Dominik Zappe – A23N0011P
(zapped99@students.zcu.cz)

Listopad 2024

Obsah

1 Zadání

Účel práce

Účelem semestrální práce je vyzkoušet si vyřešit malý datově-analytický problém, ve kterém použijete některý z postupů a technik řešených na cvičeních.

Cíle

- Zvolit nebo nalézt dataset související s definovaným řešitelným problémem.
- Data vhodně předzpracovat a zvážit případně přidání dalších zdrojů nebo tvorbu sekundárních atributů (*feature engineering*), které by mohly pomoci v analýze.
- Zvolit a aplikovat vhodné metody analýzy nebo dolování dat.
- Prezentovat výsledky s důrazem na metodologii, interpretaci výsledků a získané poznatky.

Struktura práce

Formulace problému

Na začátku práce je potřeba definovat, jaký problém vás zajímá a chcete se pokusit ho vyřešit. Můžete (A) vycházet z konkrétních dat a vytvořit hypotézu, kterou budete chtít ověřit, nebo (B) nejprve stanovit otázku ke zkoumání a vhodná data vyhledat až následně.

Například pro přístup (B) si lze položit hypotetickou otázku, zda délka názvu předmětu na VŠ má vliv na úspěšnost zkoušky. To by vedlo k potřebě získat data o výsledcích zkoušek a názvech předmětů. Následně by mohlo být podstatné nález porovnat s další univerzitou a zjistit, zda je výsledek obecný nebo specifický pro danou školu.

Výběr/Získání dat

Dle zvoleného přístupu buď (A) již máte primární dataset a můžete zkusit dohledat další, který by přidal relevantní informaci pro zpracování, nebo (B) budete muset veškerá data vyhledat na základě zvolené otázky.

Pokud byste potřebovali k volbě datasetu/otázky inspiraci, zkuste se podívat například na:

- **Kaggle.com** – kde jsou jednak zajímavé datasety s náměty na zpracování, případně tam bývají vypisované i soutěže pro řešení Data Science problémů.

- **API u vaší oblíbené služby/aplikace/hry** – poměrně často lze získat nějaká data o vaší aktivitě, případně anonymizovaná data o ostatních uživatelích. Je možné, že by tam bylo něco zajímavého k dalšímu zpracování.

Data by měla být veřejná nebo eticky získaná s příslušnou licencí vhodnou pro naše použití. Pokud máte vlastní citlivá data, zvažte, do jaké míry je bude potřeba anonymizovat, a v projektu používejte až anonymizovanou verzi.

Předzpracování dat

V této fázi budete řešit klasické transformace pro očištění dat od chybějících hodnot, diskretizaci numerických hodnot, normalizaci/standardizaci, redukci dimenzionality atd. Kromě toho zvažte, zda by nebylo vhodné vytvořit i nějaká sekundární data, například:

- **Data snapshot charakteru** – v podstatě aktuální stav v databázi, který se průběžně mění. Zvažte, zda nepoužít data z různých časových okamžiků a nevytvořit časové řady nebo odvodit například aktivitu uživatelů (pokud se mezi snapshoty změní atribut jako počet odehraných her, poslední přihlášení, oblíbené položky, lze usoudit, že uživatel byl někdy v mezičase aktivní).
- **Časové řady** – pokud pracujete s časovými řadami, budete potřebovat vytvořit odvozené hodnoty *lag* a *lead* intervalů. Například u meteorologických dat byste pro záznam s aktuální teplotou přidali atribut teploty z předchozího dne nebo z příslušného období minulého roku.

Analytický postup

Zde je potřeba individuálně zvolit statistickou techniku nebo model strojového učení, který může vyřešit vaši otázku nebo nějakým způsobem pomoci v analýze. Zvažte hlavně koncepty z první půlky semestru (Weka), ale není problém řešit otázku i jinými způsoby nebo alternativními knihovnami (například `scikit-learn` v Pythonu).

Nezapomeňte v dokumentaci zdůvodnit výběr metody a jaké očekávání máte od zvoleného postupu.

Prezentace výsledků

Chápejte prezentaci jako prostor, kde se pokusíte předat (laikům) insight, který jste z dat získali. Nejde jen o to ukázat výsledky, ale i vysvětlit, co znamenají a jak by mohly být využity. K tomu může sloužit pomocná ilustrace nebo dobře sestavený podpůrný slide.

Kromě toho mě bude zajímat technická část, tj. vysvětlení metodologie, kterou jste použili, jak jste se k výsledkům dostali, a jaké výzvy jste museli řešit.

Výstupy

- **Dokumentace** – popis dat, metodologie (včetně postupu předzpracování dat), dosažených výsledků a jejich interpretace. V dokumentaci uvítám i popis slepých uliček a problémů, které jste řešili.
- **Prezentace** – krátká prezentace výsledků na příslušném cvičení.

2 Data

Zvolenými daty pro analýzu je dataset `https://www.kaggle.com/datasets/sosperec/massive-world-of-warcraft-pvp-mists-of-pandaria?select=games.csv`. Jedná se o záznamy z her hráčů proti hráčům (*PvP*) z privátního serveru hry *World of Warcraft* – nejedná se o oficiální server. Data obsahují 586 149 zápasů, z čehož 195 838 se týká *Battlegroundů* (jedná se herní režim, kde více hráčů soupeří proti vícero hráčům – 10 proti 10, 15 proti 15 a 40 proti 40). Data obsahují časové značky začátku zápasů, trvání zápasů, konkrétní mapu a formát Battlegroundu (10vs10…). Dataset dále obsahuje druhý soubor – podrobnosti k jednotlivým hráčům. Záznamů hráčů je celkem 5 915 447, z kterých 3 740 347 se účastnilo nějakého Battlegroundu. O hráčích jsou dostupné následující informace – anonymizované ID (stejné ID napříč ruznými zápasy pro konkrétní hráče), rasa, třída, zda-li hráč daný Battleground vyhrál, kolik jiných hráčů zabil a kolikrát sám zemřel, kolik poškození za celou hru udělal, resp. jaké léčení způsobil a pro obě tyto hodnoty ještě také kolik sám dostal (poškození, léčení). Jedná se o poměrně rozsáhlý dataset, co se časového sběru týče – data začínají dubnem 2020 a končí březnem 2023.

Pro stažení a načtení dat byl vytvořen skript *download.py*, konkrétně funkce *dataset_download()*.

2.1 Formulace otázek a hypotéz

Nad daty bylo dotázáno několik jednoduchých otázek / hypotéz pro ověření pravdivosti.

1. Je třída *Paladin*, *Hunter*, *Warrior* a *Death Knight* lepší než ostatní třídy? Mají tyto třídy větší pravděpodobnost výhry?
2. Je rasa *Human* lepší než ostatní rasy? Má tato rasa větší pravděpodobnost výhry?
3. Zvyšuje více *healerů* (hráčů, kteří léčí) šanci na výhru?
4. Jsou hráči více aktivní o víkendu?

Na tyto otázky lze jednoduše odpovědět jednoduchou statistickou analýzou – průměry, standardní odchylky atp. Jinou možností by bylo „multivariate analýza“ – v práci byl však zvolen jednodušší postup.

Proto byl navrhnut následující problém pro zpracování pomocí strojového učení: „Predikuj výsledek a dobu trvání Battlegroundu ještě před jeho začátkem za pomoci historie hráčů a konkrétní mapy“. Jsou zde hned tři problémy:

1. Predikce výkonu individuálních hráčů – Regrese

2. Predikce toho, jak celkově Battleground dopadne – Klasifikace
3. Predikce doby trvání Battlegroundu – Regrese

Jelikož je dat velké množství byl zvolen přístup přes *neuronové sítě* (ty by měly fungovat lépe než např. SVM, které by fungovalo spíše pro menší datasety). Veškerá implementace strojového učení je ve skriptu *machine_learning.py* – bylo využito knihovny *torch*. Pro implementaci a následné srovnání byly zvoleny dva přístupy:

1. Pro každý z výše zmíněných problémů naimplementovat vlastní model – tedy 2 regresory a 1 klasifikátor. Tyto podmodely jsou později spojeny a zastřešeny jedním velkým modelem, který pouze využívá jednotlivé podmodely.
2. End-to-End přístup, kdy se do neuronové sítě přivedou všechny vstupy a chce se po ní, aby si problém sama vnitřně vyřešila (trend dnešní doby).

2.2 Předzpracování

Předzpracováním se zabývá skript *dataloader.py*.

Hlavním problémem zde byla nekonzistentnost dat. Některé řádky byly označeny jako „bg“, ačkoliv se nejednalo o Battleground ale o arénu (menší formát PvP – 2vs2, 3vs3). Bylo tedy zapotřebí data pořádně přefiltrovat – zvolený postup na základě názvu mapy (arény mají odlišné mapy).

Dále sloupec „player_id“ byl zapsán jako hodnota v plovoucí desetinné čárce, ačkoliv se nikde nevyskytovala žádná desetinná část – převod na celočíselný typ.

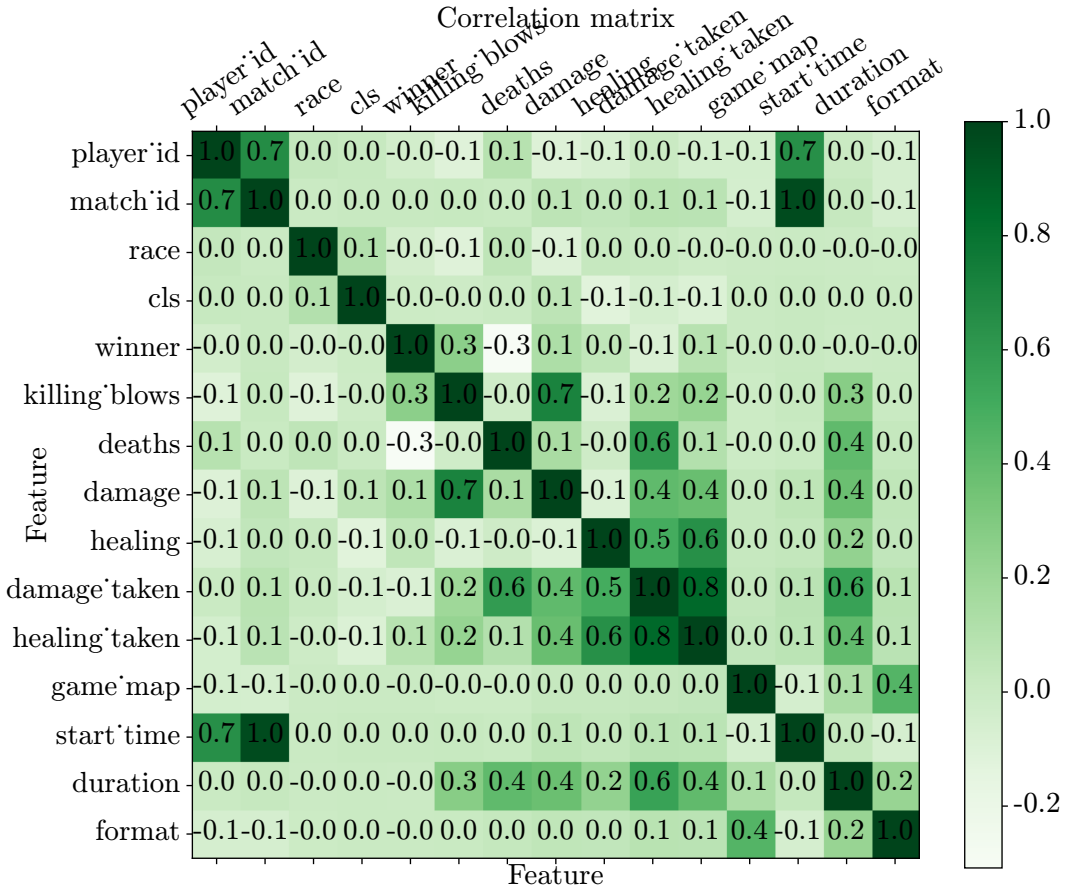
Sloupec „start_time“ a „duration“ byli ve formátu časů – nepřijatelné pro pozdější zpracování strojovým učením. Časová značka začátku Battlegroundu byla převedena do času od začátku časů (1. 1. 1970) v sekundách. Trvání Battlegroundu bylo převedeno z formátu mm:ss na sekundy.

Dalším problémem byly řetězce. Jedná se o sloupečky „race“, „cls“, „game_map“ a „format“ (tento sloupec byl syntenticky přidělán na základě sloupce „game_map, byl také užit pro zmíněnou filtraci“. Zvoleným řešením je *One hot encoding* – např. pro sloupec ras bylo vytvořeno mapování unikátních řetězců na index – následně byla hodnota ve sloupci ras nahrazena tímto indexem do One hot encodingu. Např. nachází-li se v datech jako první záznam o hráči hodnota rasy „human“, budou pak všichni „human“ nahrazeni hodnotou 0.

Případné další předzpracování specifické pro jednotlivé statistické metody / metody strojového učení budou zmíněny v odpovídající kapitole.

3 Statistická analýza

Jako základní statistická byla spočtena korelace jednotlivých sloupců mezi sebou.

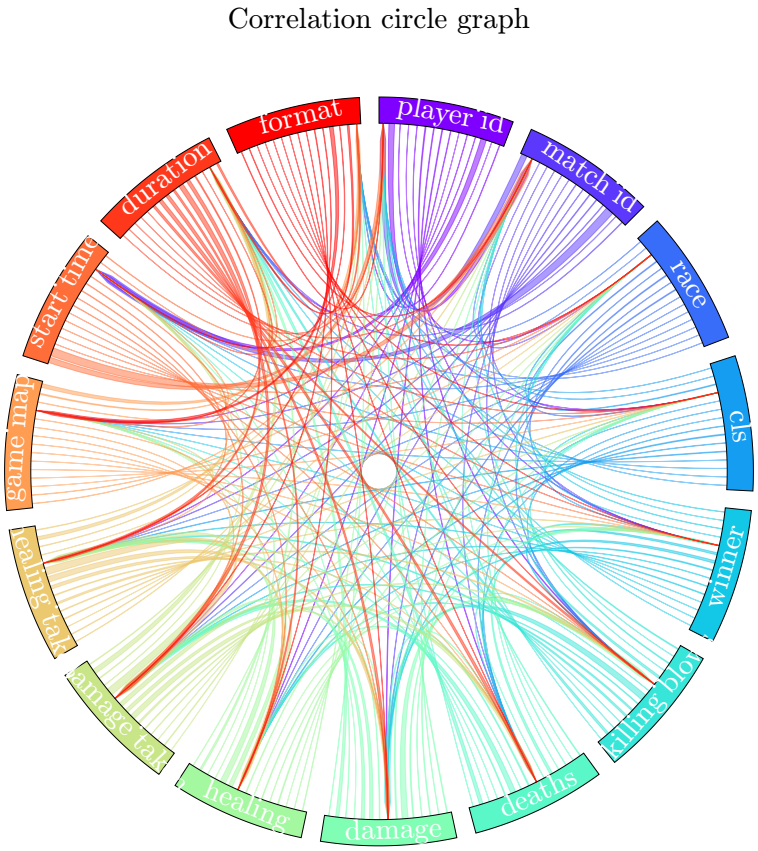


Obrázek 1: Matice korelací

Z Obrázku 1 je vidět, že např. „match_id“ je redundantní informace, když se vezme v potaz, že dva různé Battlegroundy nejspíše nezačnou přesně ve stejný moment („start_time“. Dále je vidět, že zda-li hráč vyhrál kladně koreluje se zabitími, které má; naopak záporná korelace je se smrtmi, které má. Dále je vidět, že poškození kladně koreluje se zabitími, dle očekávání; stejně tak smrti kladně korelují s utrpěným poškozením. Dále utrpěné poškození kladně koreluje se získaným léčením – náznak *tank* třídy. Také je vidět, že léčení kladně koreluje s utrpěným poškozením a získaným léčením – je-li hráč léčitel, je cílem pro ostatní

hráče a musí sám sebe léčit – opět smysluplné. V neposlední řadě je vidět, že jednotlivé herní statistiky hráčů také kladně korelují s délkou hry – čím déle Battleground trvá, tím více se poškození dá a utrpí, obdobně s léčením, smrtmi a zabitími.

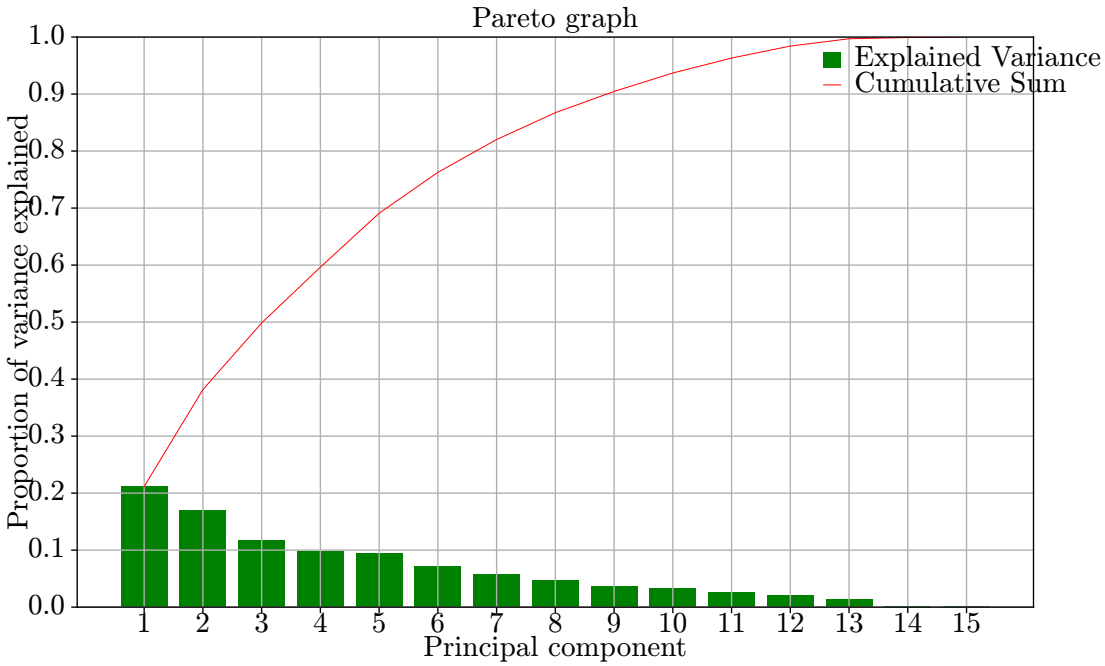
Následující Obrázek 2 je pouze experimentální zobrazení korelací a není z něj moc věcí vidět. Obrázek spíše vypadá jako mandala, než aby z něj šlo něco rozumného vyčíst.



Obrázek 2: Kolečko korelací

Další provedenou statistickou analýzou bylo *PCA* – Principal Component Analysis – za účelem redukce dimenzionality problému a dat.

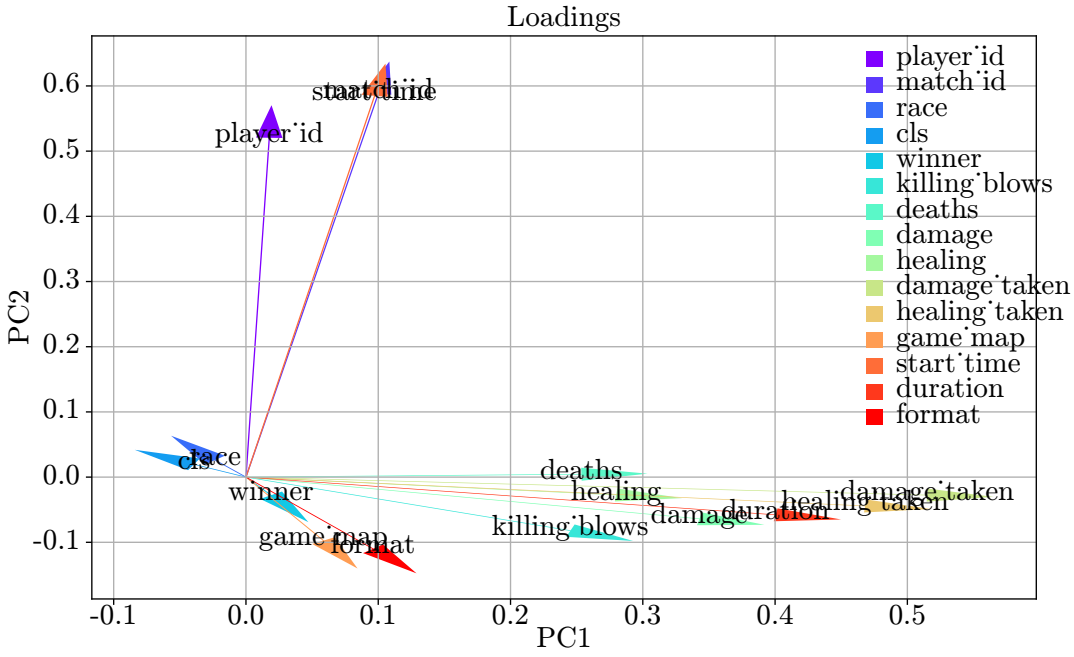
Z Obrázku 3 je vidět, kolik komponent je zapotřebí pro zachování daného procenta informací – např. pro zachování alespoň 90 % informací by stačilo použít 9 komponent.



Obrázek 3: PCA: Pareto graf

Na následujícím Obrázku 4 je vidět *zatížení hlavních komponent* (*Loadings*) v prostoru prvních dvou komponent. Ačkoliv to nejde takto napřímo reprezentovat, je ale možné si všimnout, jak spolu nějaké vektory (loadingy) mají podobný směr a velikost. Je vidět, že ID Battlegroundu a časová značka jeho začátku spolu kompletně splývají v prostoru prvních dvou komponent (nesou stejnou informaci). Obdobně je vidět, že rasa a třída spolu souvisí. Stejně tak herní statistiky hráčů spolu navzájem souvisejí.

Poznámka: bohužel jsem při tvorbě neuronových sítí zapomněl na PCA a už jsem nechtěl měnit návrh neuronových sítí a dimenzionality vstupů – nebylo tedy nakonec redukce dimenze užito, ač by to mohlo zlepšit výkon modelů. Takto jsou alespoň jednotlivé vstupy do neuronových sítí dobře reprezentovatelné. Navíc si neuronové sítě nejspíše vytvoří svojí vlastní redukci dimenzionality v rámci nějaké jedné skryté vrstvy.



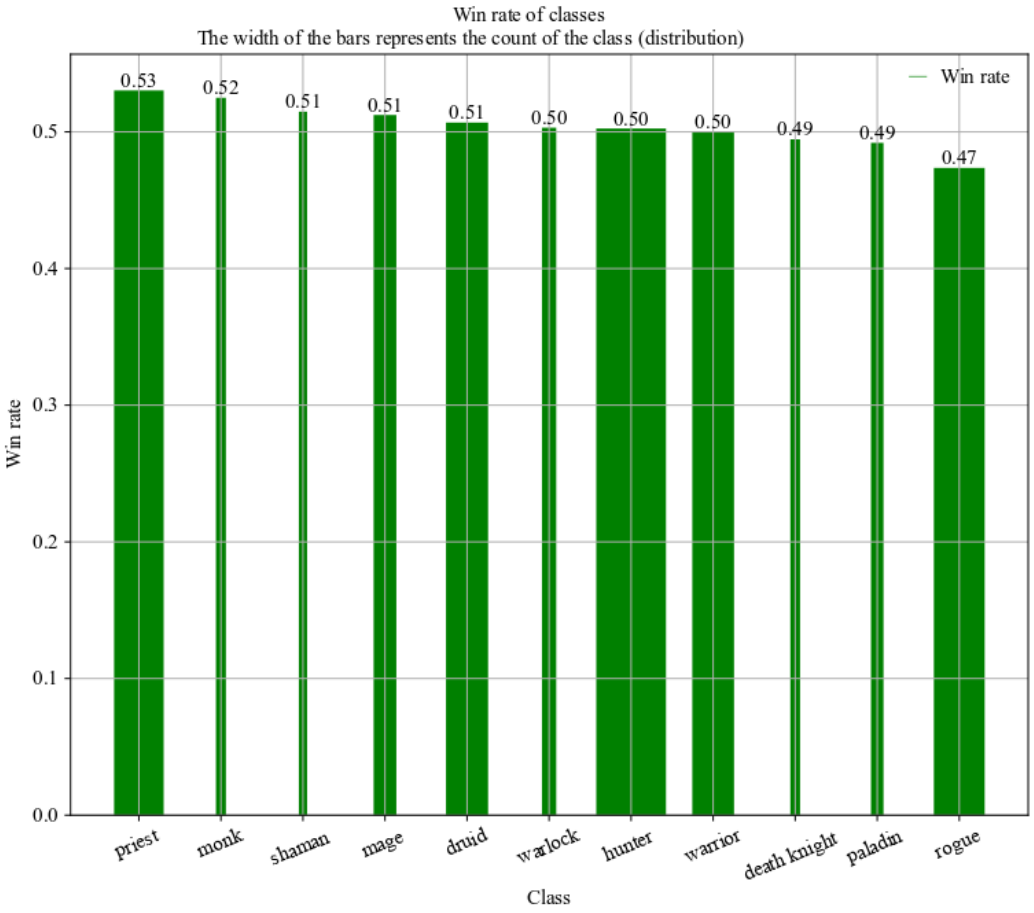
Obrázek 4: PCA: Zatížení hlavních komponent (Loadingy)

3.1 Otázka 1

Je třída *Paladin*, *Hunter*, *Warrior* a *Death Knight* lepší než ostatní třídy? Mají tyto třídy větší pravděpodobnost výhry?

Pro odpověď na tuto otázku byl dataset pouze seskupen podle tříd a byl napočítán průměr, standardní odchylka a počet výskytů (standardní odchylka je v případě výhrovosti k ničemu, spíše jen slouží jako „sanity check“). V ideálním světě by všechny třídy měly mít „win rate“ (tedy napočtený průměr) roven 0.5, se standardní odchylkou 0.5 (nehledě na počet výskytů). Realita je vyobrazena na Obrázku 5. Počet výskytů byl v obrázku využit tak, že čím větší výskytovost, tím tlustší daný sloupec je. Tedy nejvíce záznamů je o třídě „Hunter“.

Z Obrázku 5 je navíc vidět, že moje domněnka (vyřčená hypotéza) je kompletně mylná. Mnou vyřčené 4 třídy se nacházejí na opačném konci grafu, než jsem očekával. Naopak nejvíce „rozbitou“ třídou je *Priest* a *Monk*. Nejhůře si vedou hráči třídy *Rogue*.

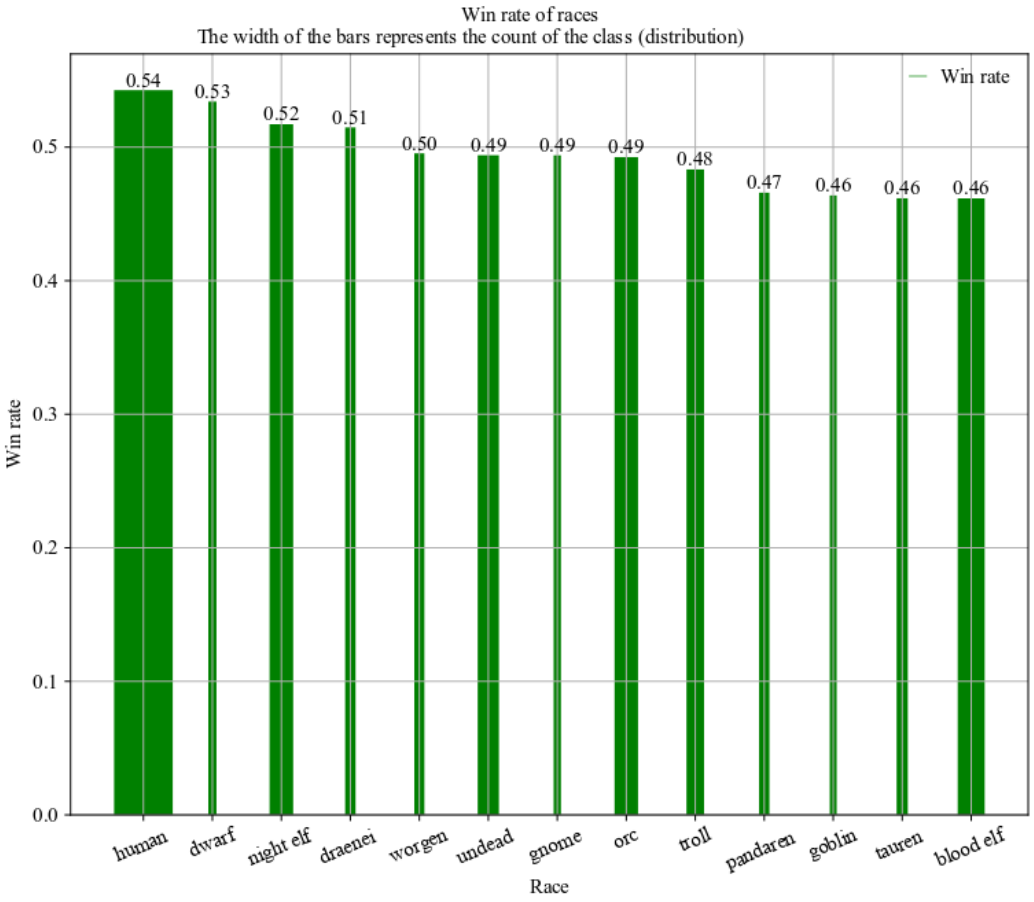


Obrázek 5: Win rate jednotlivých tříd

3.2 Otázka 2

Je rasa *Human* lepší než ostatní rasy? Má tato rasa větší pravděpodobnost výhry?

Pro odpověď na tuto otázku byl opět dataset pouze seskupen, nyní však podle rasy. Opět byl spočten průměr, standardní odchylka a počet výskytů. V ideálním světě se dají očekávat stejné hodnoty, jako pro třídy – realita je vidět na Obrázku 6. Opět platí, že nejtlustší sloupec je nejhranějším.



Obrázek 6: Win rate jednotlivých ras

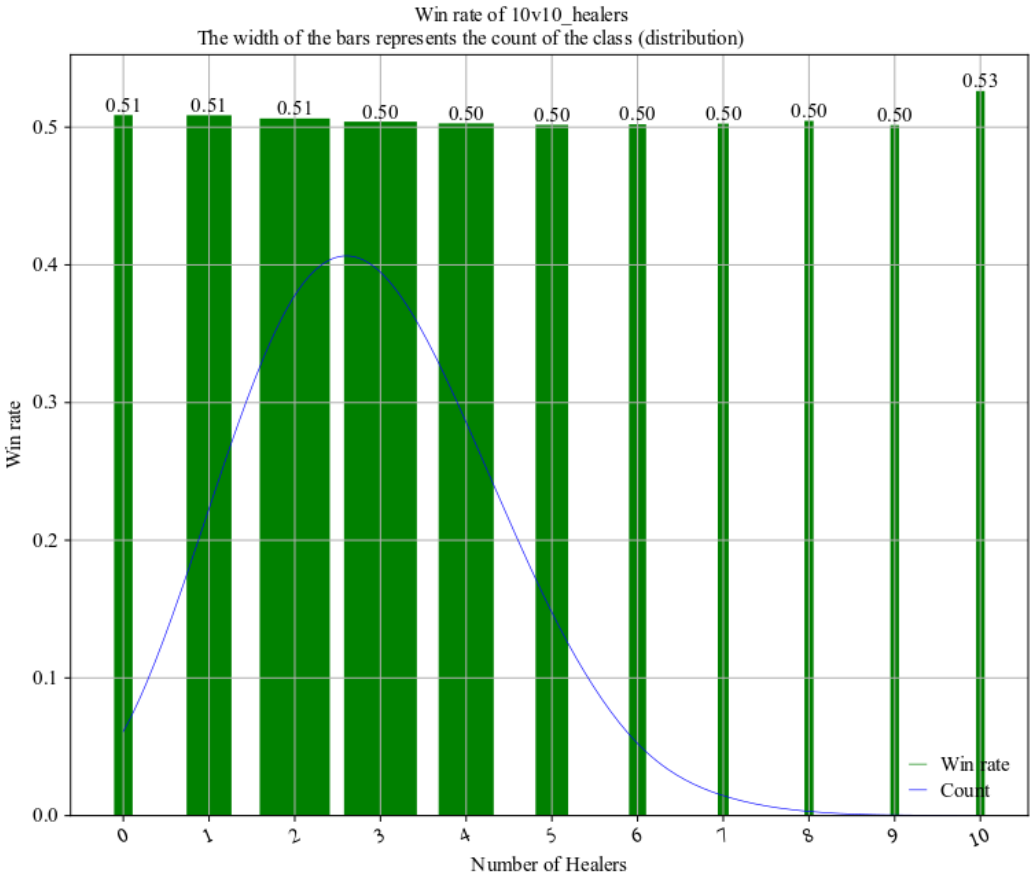
Z Obrázku 6 je bohužel vidět, že nejvíce hráčů PvP opravdu hraje Člověka, který má navrch. Zde se graf zásadně liší, od Obrázku 5. Jsou zde vidět mnohem větší rozdíly a mnou vyřčená hypotéza, že Člověk je nejlepší rasou pro PvP, je bohužel potvrzena. Zde jsou vidět možnosti, jak hru lépe vybalancovat (alespoň co se PvP týče). Zajímavým faktem je, že 4 přední příčky win ratu ras zabírají rasy Aliance. Naopak 6 nejhorších příček obsadili rasy Hordy (+ Pandaren – ten může být u obou frakcí). Pasivní schopnosti Hordských ras jsou spíše cílené na PvE, zatímco Alianské rasy mají pasivní schopnosti cílené do PvP – rozdíl je jednoznačně viditelný.

3.3 Otázka 3

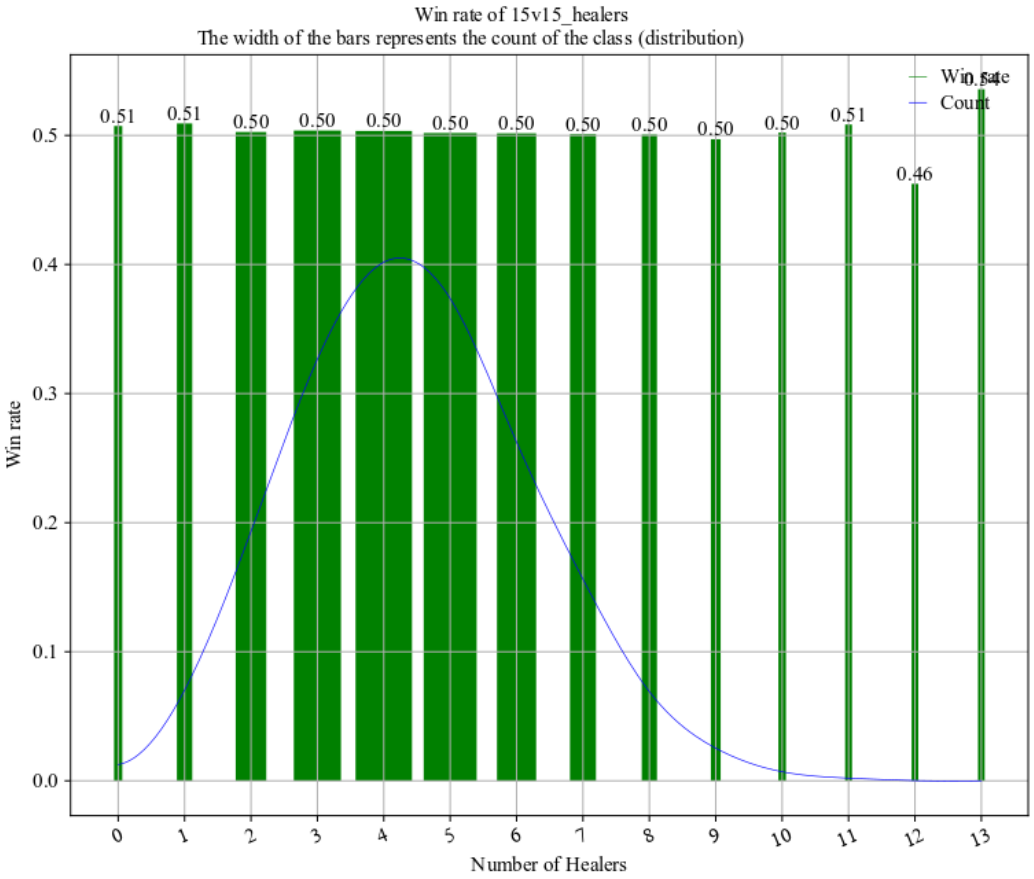
Je výhodné mít v týmu více *healerů* (hráčů, kteří léčí)?

Zde se jedná o poněkud složitější otázku – to hlavně z důvodu, že v původních datech nejsou hráči jednoznačně označeni, jestli hráli jako *DPS* (*Damage dealer*), *Healer* (*Léčitel*) nebo *Tank*. Jen některé třídy mohou být léčiteli, byli tedy z dat vyfiltrování hráči, kteří ani nemohou léčitelem být. Následně byl zvolen filtr stylem: „Pokud daný hráč dal alespoň dvakrát více léčení, než poškození, bude prohlášen za léčitele“. Jakožto „sanity check“ zde slouží počet léčitelů vůči ostatním a procentuelně je tímto dosaženo, že 19.6 % hráčů je označeno jako léčitel, což odpovídá hernímu systému, kde na 5 hráčů se hra snaží spojit 3 DPS, 1 Healer a 1 Tank. Hry byly následně seskupeny podle formátu, neboť jiný počet léčitelů dává smysl pro tým 10 lidí, jiný počet pro tým 40 lidí. Na následujících třech Obrázcích 7, 8 a 9 jsou vidět výsledky. Opět platí, že nejtlustší sloupec je nejzastoupenějším případem. V těchto grafech je navíc osa X seřazená vzestupně (počet healerů v Battlegroundu), je proto modrou čarou zaznamenán také počet takových Battlegroundů, kde se tento fenomén vyskytl (prakticky stejná informace jako tloušťka sloupce). Díky modré čáře je však ověřitelná i správnost seskupování a výpočtu, neboť očekávání je takové, že se počet léčitelů na Battleground chová normálně.

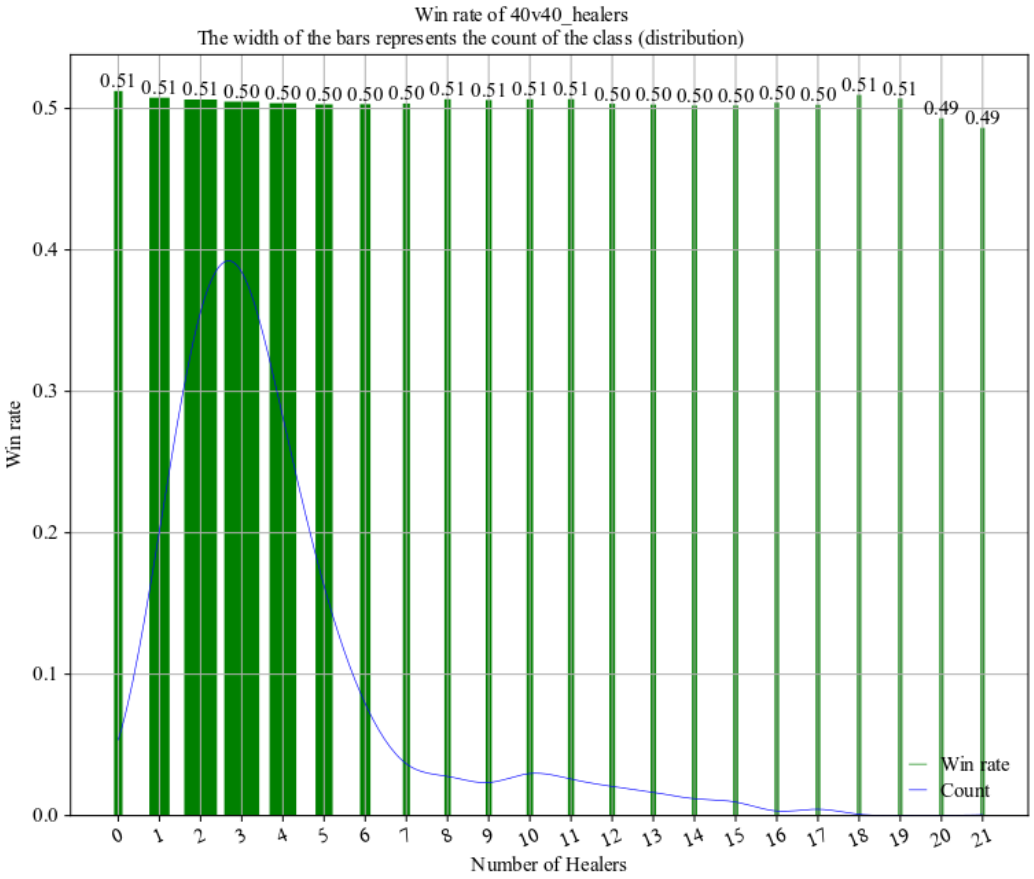
Z grafů je obecně vidět, že není nutně pravda, že čím více léčitelů v zápase je, tím větší je výhernost týmů s léčiteli. Spíše to vypadá, že na počtu léčitelů přiliž nezáleží.



Obrázek 7: Win rate podle počtu léčitelů v Battlegroundu (10 vs 10)



Obrázek 8: Win rate podle počtu léčitelů v Battlegroundu (15 vs 15)



Obrázek 9: Win rate podle počtu léčitelů v Battlegroundu (40 vs 40)

3.4 Otázka 4

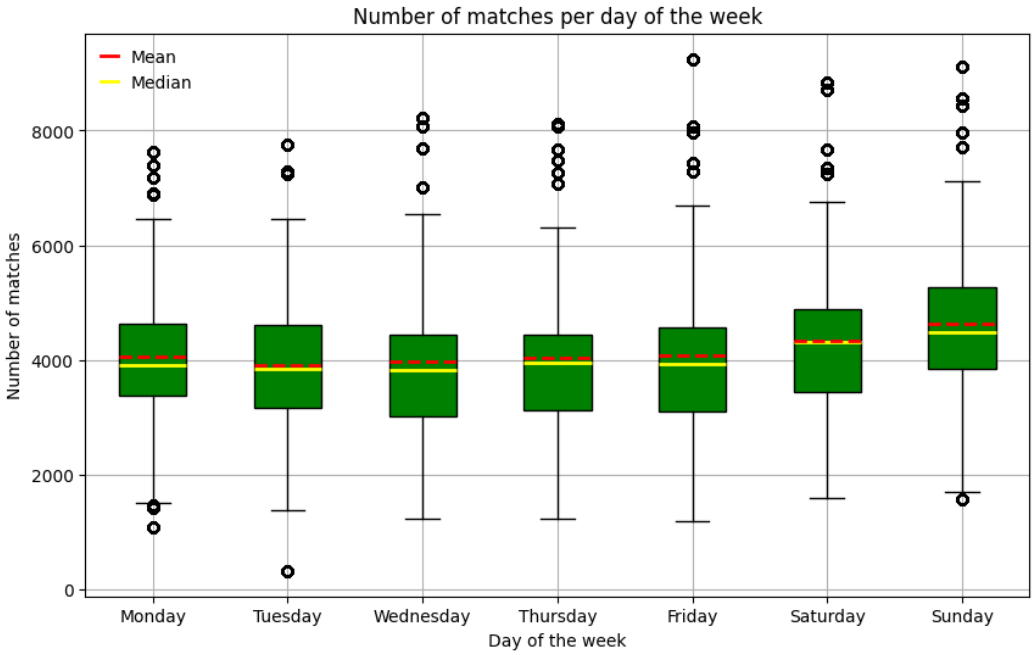
Jsou hráči více aktivní o víkendu?

Osobně bych zde čekal, že ano, navíc však může mít vliv PvE obsah hry na aktivnost hráčů v PvP. Ve hře jsou velké PvE instance (*Raidy*), které jdou splnit pouze jednou za týden, pak se musí počkat na *reset* instancí, aby se mohly splnit znovu. Tento reset nastává v úterý – očekával bych tedy malý pokles hráčů v úterý, ale záleží, kdy hráči tento obsah hry plní a jestli vůbec.

Pro vytvoření grafů zde bylo využito jednoduché seskupování podle časové značky začátku

Battlegroundu (byla zapotřebí jednoduchá transformace na den). Jako typ grafu zde byl zvolen boxplot – viz Obrázek 10.

Z Obrázku 10 je zjevné, že hráči opravdu více hrají o víkendu, což se dalo očekávat. Navíc je vidět malý poklesu v úterý a středu, což by mohlo souviset se zmiňovanými *Raidy* (červená čárkovaná čára – průměr – se hnul pod hodnotu 4 000).



Obrázek 10: Počet zápasů v daný den v týdnu

4 Strojové učení

Úplně původně jsem v práci chtěl udělat klasifikátor, který by na vstupu přijal postavu hráče a řekl by mu, jestli udělal při tvorbě postavu dobrou volbu, či špatnou, vzhledem k tomu, jestli chce hrát PvP Battlegroundy. Jelikož však v datech tento sloupeček není, musel bych sám navrhnout metriku a o-labelovat data sám. Když už bych však měl funkci pro metriku a labelování, nemá pak smysl mít klasifikátor a mohl bych prostě nové vstupy vložit do dané funkce pro metriku.

Obdobně jsem původně chtěl navrhnout regresor, který by na vstupu přijal herní statistiky hráče z Battlegroundu a ohodnotil by hráče např. na stupnici od 0 do 100, jak dobře či špatně hrál daný zápas. Jelikož ale opět v datech žádné takové skóre není, nastává zde výše popsaný problém, který vede k nenutnosti implementovat jakékoliv strojové učení.

Tyto dva problémy byly slepou cestou a vzhledem k dostupným datům bylo nakonec navrženo následující (rekapitulace):

1. Predikce výkonu individuálních hráčů – Regrese
2. Predikce toho, jak celkově Battleground dopadne – Klasifikace
3. Predikce doby trvání Battlegroundu – Regrese

Výsledkem by měl být nástroj, který na vstupu přijme ID hráčů a název mapy Battlegroundu a model si vnitřně předpoví, jak budou jednotliví hráči hrát a dále předpoví, jak celkově hra dopadne a za jak dlouho zápas přibližně skončí.

4.1 Přístup 1: Model podmodelů

Prvním přístupem bylo rozložit si problém na výše zmíněné tři body – tři modely. Pro každý model byla vypracována zvlášť funkce pro předzpracování dat – zde bylo nejvíce práce. Samotné trénování pak bylo snadné, pro všechna data je v kódu společná trénovací smyčka.

Regresory používají ztrátovou funkci *MSE* (Mean Squared Error) a klasifikátory používají *BCE* (Binary Cross Entropy) (neboť se vždy jedná o binární klasifikaci – vyhrál buď jeden tým, nebo ten druhý).

První model přijímá na vstupu 10 her z historie hráče a predikuje, jak by mohla vypadat hra jedenáctá. Posledních 10 her hráče je bráno opravdu jako posledních 10 her ve smyslu datumů. Pokud hráč například poslední dobou nebyl aktivní, vezmou se klidně rok staré údaje. Naopak pokud hráč nemá 10 her, využije se průměr z jeho alespoň nějakých her pro „padding“. Pokud je hráč kompletně nový, použije se průměr přes všechny hráče jako

„padding“. Zde by bylo možná vhodnější užít percentil (např. 0.25), protože nový hráč nejspíše nebude alespoň tak dobrý, jako průměrný hráč. Trénovací dataset byl složen z hráčů, kteří mají alespoň 11 her a 10 her bylo vzato jako trénovací data a jedenáctá hra byla považovaná za kýžený cílový výstup sítě. Do sítě jsou na vstupu vložena úplně všechna dostupná data – včetně redundancí a neužitečných hodnot – síť je dostatečně velká, aby redundance zjistila a případně odhalila skryté závislosti s daty, které nám mohou přijít neužitečné. Na vstup tedy přijde určitě užitečné – „rasa“, „třída“, „zda hráč vyhrál daný zápas“, „počet zabití“, „počet smrtí“, „dané poškození“, „dané léčení“, „utrpěné poškození“, „získané léčení“ – dále méně užitečné (ale možná pro síť užitečné) – „player˙id“, „match˙id“, „mapa battlegroundu“, „formát battlegroundu“, „čas zahájení zápasu“, „trvání zápasu“. Diagram dimenzionalit navrženého modelu je k vidění na Obrázku 11

Druhý model přijímá na vstupu predikce z prvního modelu pro celé týmy a přidává k tomu informaci o mapě. Požadovaný výstup je zde Ano / Ne – zda-li vyhrál první či druhý tým. Jelikož při tvorbě trénovacího datasetu bylo zapotřebí hráče shlukovat do týmů a na privátním serveru, z něhož jsou data sesbíraná, byl zapnutý tzv. *cross-faction* – tj. za tým Hordy klidně může hrát rasa Aliance a naopak – není jednoduché hráče shlukovat. Je zapotřebí užít sloupce „winner“. Tím by však vždy Tým 1 byl např. hráči, co vyhráli, a Tým 2 vždy ti, co prohráli. Proto při tvorbě datasetu bylo hozeno mincí (random < 0.5) a výherní tým byl buď Tým 1, nebo Tým 2 – aby se síť jen hloupě nenaučila říkat, že Tým 1 vždy vyhrává. Diagram dimenzionalit navrženého modelu je k vidění na Obrázku 12.

Třetí model má stejné vstupy jako model druhý, ale výstup se liší. Nyní není výstupem třída – Ano / Ne – ale predikovaná délka zápasu (regrese). Jelikož se třetí model pouze liší regresí od druhého modelu (klasifikace), je jeho diagram shodný s diagramem druhého modelu, viz Obrázek 12.

U všech modelů byl problém s obrovskými čísly – všechna data bylo nutné normalizovat, pro budoucí zpětnou reprezentaci pro uživatele jsou ukládány tyto normalizační konstanty pro zpětnou denormalizaci.

Výsledný model pak pouze kombinuje tři zmíněné „podmodely“ a kombinuje jejich výstupy.

4.2 Přístup 2: End-to-end model

Oproti implementaci po částech byl tento model mnohem jednodušší na implementaci – je zde méně režie s různými předzpracováními. Prakticky se dělá podobné předzpracování, jako pro podmodel 2 a 3. Síť na vstupu přijme 10 historických dat pro každého hráče v Battlegroundu (záleží na pořadí, nejprve jsou hráči Týmu 1, pak hráči Týmu 2) a připojí se k tomu informace o konkrétní mapě, na které se bude hrát (orientačně vstupní počet neuronů závisí na formátu Battlegroudnu, ale řídí se vzorečkem $\#hracu \cdot \#features_{hrac} \cdot \#her_{historie} \cdot \#tymu + 1$ (+1 kvůli mapě) – ve finále má tedy síť 3 001 vstupních neuronů pro

formát 10v10, 4 501 pro formát 15v15 a 12 001 pro formát 40v40). Očekávaným výstupem je klasifikace výhry Týmu 1 – Ano / Ne – a zároveň regrese odhadované délky trvání zápasu. Diagram dimenzionalit navrženého modelu je k vidění na Obrázku 13.

Zde byl největší problém s definicí ztrátové funkce, zkoušel jsem nejprve užít MSE, které však není vhodné pro klasifikaci – výsledky nebyly uspokojivé. Poté jsem zkoušel BCE, která však není zase vhodná pro regresi – výsledky opět neuspokojivé. Takto bylo vyzkoušeno vícero ztrátových funkcí, žádná však nefungovala. Byla proto implementovaná vlastní ztrátová funkce, která pouze kombinuje MSE a BCE – viz funkce *combined_loss()* v *machine_learning.py*. Výsledná hodnota ztrátové funkce je průměr MSE a BCE, které jsou zvlášt užity na jednotlivé výstupy sítě – MSE pro délku zápasu, BCE pro predikci výhry. Tato skutečně je v diagramu na Obrázku 13 podchycená výstupní vrstvou jako (1+1), nikoliv (2).

Druhým největším problémem je délka preprocesování dat a délka trénování většího modelu. Času je však dost, takže se nejedná o nějak zásadní problém. Navíc se všechny modely a normalizační konstanty cache-ují, na dlouhé trvání trpí pouze první spuštění.

4.3 Evaluace

Každému z modelů (i podmodelů) byl navrhnut *baseline* model – *Radnom Model* – tedy model, co si náhodně tipuje. Pro generování náhodných čísel byla vždy zvolena vhodná distribuce – pro predikci výkonu hráče bylo zvoleno normální rozdělení, pro predikci výsledku hry uniformní a pro délku trvání zápasu opět normální. Navíc je ještě možné porovnávat mezi sebou model podmodelů a end-to-end model.

Všechny modely byly trénovány na 10 epoch a po trénování byla změřena jejich úspěšnost na testovacím datasetu (pro regrese pouze hodnota ztrátové funkce, pro klasifikace navíc *přesnost* (*Accuracy*)). Byl také proveden experiment s delším trénováním, bohužel se však modely přeučovali na trénovacích datech a např. modely trénované po dobu 100 epoch měly na testovacích datech horší výsledky, než modely trénované po dobu 10 epoch.

Na následující Tabulce 1 jsou vidět dosažené hodnoty ztrátových funkcí (případně také přesností) na testovacích datech pro jednotlivé modely a jejich odpovídající Random baseline modely. Jelikož modely 2, 3 a end-to-end model jsou prakticky tři různé modely (pro každý formát zvlášť trénován vlastní model – 10vs10, 15vs15 a 40vs40), hodnoty v tabulce jsou průměrem přes tyto tři verze. Z Tabulky 1 je vidět, že všechny modely si dařily uspokojivě. V poslední části Tabulky 1 je možné si povšimnout srovnání jednotlivých přístupů – srovnání modelu podmodelů s end-to-end modelem.

Model	**Metrika**	**Hodnota**
Model 1	MSE loss	0.404
Random baseline	MSE loss	2.014
Model 2	BCE loss	0.710
Model 2	Přesnost	0.730
Random baseline	BCE loss	0.732
Random baseline	Přesnost	0.500
Model 3	MSE loss	0.872
Random baseline	MSE loss	1.986
Model podmodelů	Kombinovaná loss	0.693
Model podmodelů	Přesnost	0.760
End-to-End Model	Kombinovaná loss	0.702
End-to-End Model	Přesnost	0.783
Random baseline	Kombinovaná loss	1.371
Random baseline	Přesnost	0.489

Tabulka 1: Srovnání Modelů s jejich Random baseliny na testovacích datech

5 Závěr

Semestrální práce se zabývala datovou analýzou nad rozsáhlými daty z World of Warcraft Battlegroundů.

Ze statistické analýzy vyplývá, že nejvíce se v Battlegroundech vyplatí hrát *Human Priest* a nejméně vhodný je *Blood elf Rogue*. Samozřejmě nejvíce závisí na hráčovo dovednostech, a i *Blood elf Rogue* může být na vrcholku tabulek.

Dále ze statistické analýzy vyplývá, že na počtu léčitelů v týmu tolik nezáleží – vypadá to, že je jedno, jestli nějaký tým má o pár léčitelů navíc či naopak.

V neposlední řadě byla díky statistické analýze potvrzena hypotéza, že hráči obecně hrají více o víkendech.

Pomocí metod strojového učení (neuronové sítě) byl navrhnut nástroj, který na vstupu přijme ID hráčů, název mapy a jako výstup udává, zda-li vyhraje Tým 1 (Váš tým), či spíše Tým 2, a jak přibližně dlouhý Battleground bude. Výsledky nástroje jsou uspokojivé, ale bohužel modely nejsou nasaditelné do reálného užití, neboť ID hráčů jsou v datech anonymizována. Bylo by zapotřebí sehnat data ne-anonymizovaná, následné získání jmen / ID hráčů a název mapy by nebyl problém v reálném čase sehnat ještě před začátkem zápasu – tato data jsou dostupná všem účastníkům daného Battlegroundu.

Jako možné vylepšení do budoucna se nabízí užití redukce dimenze data (PCA), na což je práce připravená – bylo by zapotřebí pouze pozměnit dimenze v rámci neuronových sítí.

A Uživatelská příručka

Aplikace je napsaná v jazyce Python, pro její spuštění je prerekvizitou mít nainstalovaný Python verze 3+ (konkrétně byla testována nejméně verze Python 3.9+).

Dále je nutné doinstalovat si veškeré závislosti za pomocí standardního manažera balíčků *pip*

```
C:\SP>pip install -r requirements.txt
```

```
user@pc:/SP$ pip3 install -r requirements.txt
```

Při spuštění pozor na adresář, odkud se aplikace spouští! Počítá se se spuštěním z root adresáře, tedy ne z /src/. Relativní cesty v programu by pak mohly selhat!

Spuštění je pak již jednoduché:

```
C:\SP>python src/main.py
```

```
user@pc:/SP$ python3 src/main.py
```

Aplikace neočekává žádné vstupní parametry z příkazové řádky

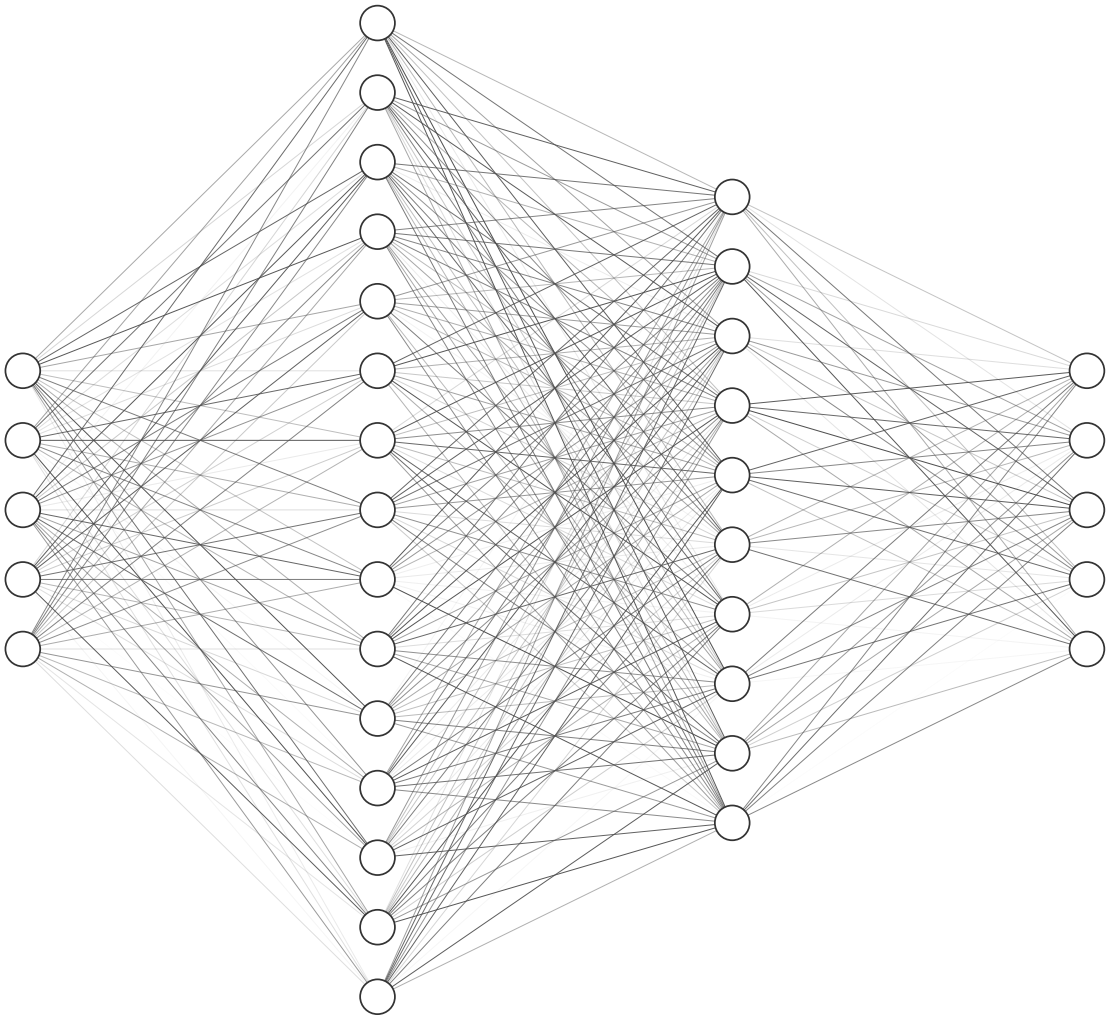
Pro příklad vlastního užití modelů slouží funkce *example_usage()* v *machine_learning.py*.

Následuje ukázka možného spuštění pomocí *skript.bat* (Windows) a *skript.sh* (Unix).

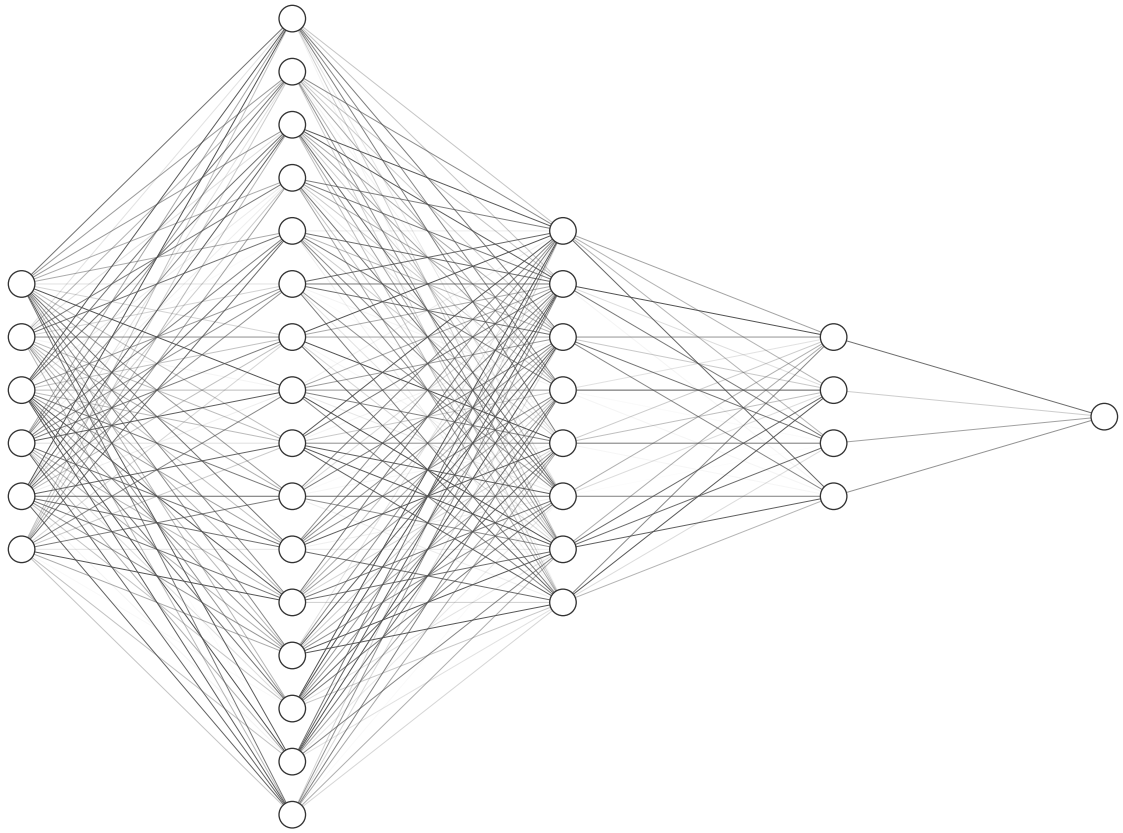
```
:: Create a virtual environment
python -m venv venv
:: Activate the virtual environment
call venv/Scripts/activate
:: Install the required packages
pip install -r requirements.txt
:: Run the app
python src/main.py
```

```
#!/bin/bash
# Create a virtual environment
python3 -m venv venv
# Activate the virtual environment
source venv/bin/activate
# Install the required packages
pip3 install -r requirements.txt
# Run the app
python3 src/main.py
```

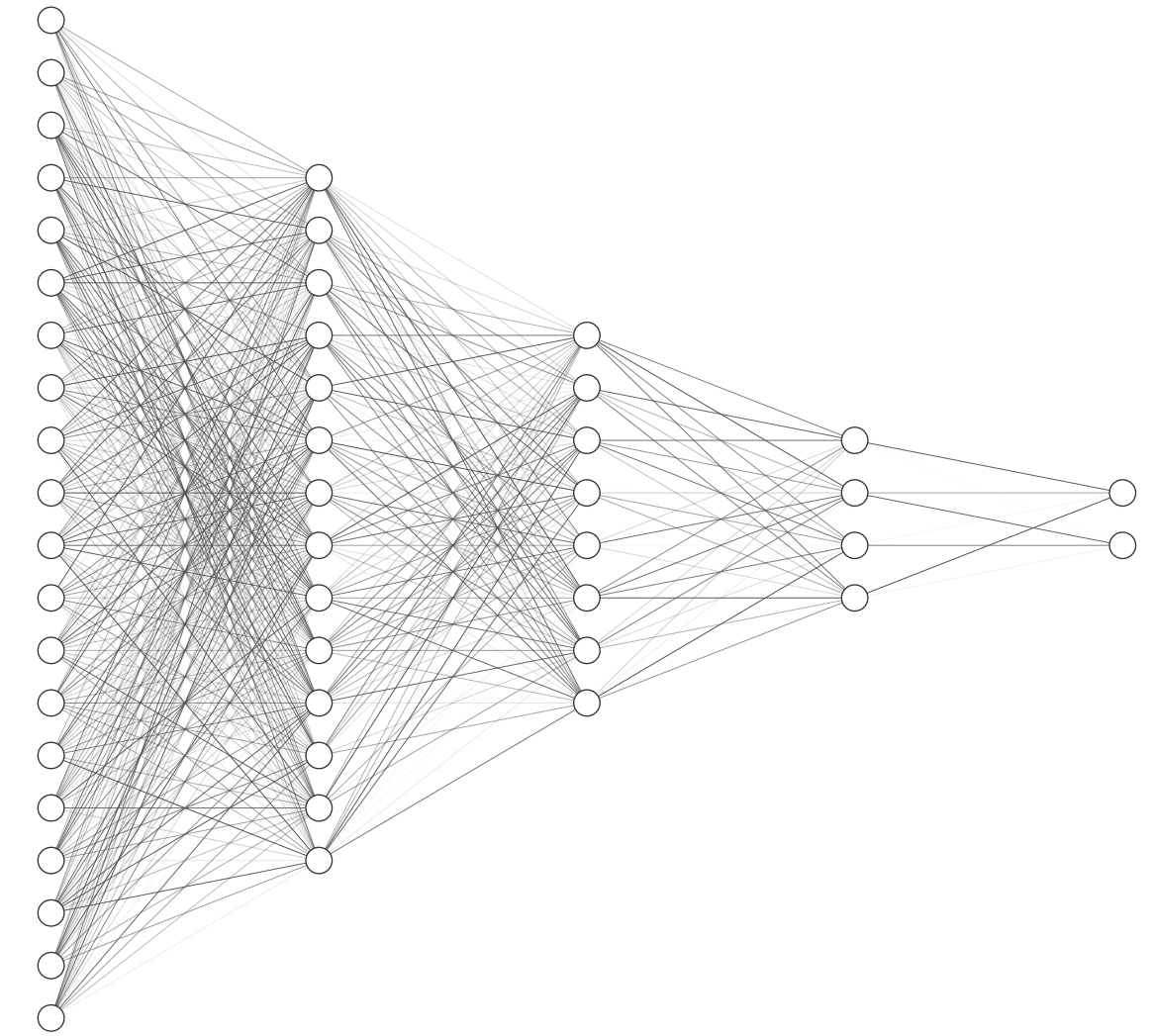
B Diagramy neuronových sítí



Obrázek 11: Diagram neuronové sítě „Submodel1“



Obrázek 12: Diagram neuronové sítě „Submodel2“ a „Submodel3“



Obrázek 13: Diagram neuronové sítě „EndToEndModel“