Can you be a winner winner chicken dinner?

Patrik Racsko

Slovenská technická univerzita
Fakulta informatiky a informačných technológií

1 Opis problému, dát

Masívne známa kompetitívna hra s názvom Playerunkown's BattleGrounds si podmanilo srdce hráčov svojou jedinečnosťou a prepracovanosťou. Hra sa začína so 100 hráčmi na masívnej mape, na začiatku sú hráči neozbrojení a vyskakujú z lietadla na ľubovoľné miesto, kde sa môžu ozbrojiť s náhodne generovanými predmetmi a bojovať o prvé miesto. Aby sa zabezpečilo, že sa každý hráč stretne, tak sa na mape náhodne vytvorí zóna, v ktorej môžu bojovať a v prípade, že sú mimo tej zóny, tak im ubúda život. Zóna sa cyklicky zmenšuje, až kým úplne nezmizne a v každej iterácií uberá viac a viac života hráčom, ktorí sú mimo nej. Hráči s najlepšou stratégiou vyhrajú.

Existuje veľa rôznych faktorov, ktoré ovplyvnia umiestnenie v rebríčku počas hry. Je dôležité vedieť ako sa hráči, skupina hráčov umiestní v rebríčku. Zo spomenutých dôvodov je jednoznačné, že je potrebné vytvoriť model, ktorý bude predikovať umiestnenie hráčov v rebríčku na základe ich štatistík z hier. Predikovať sa bude atribút winPlacePerc na škále od 1 (prvé miesto) do 0 (posledné miesto).

V rámci predikcie sa bude jednať o klasifikáciu do rôznych percentuálnych intervalov. Vhodným kandidátom na predikciu sú stromové algoritmy. Z toho dôvodu sa sa zameriame na analýzu rôznych typov stromových algoritmov.

1.1 Opis dát

Dátová množina je získaná z platformy Kaggle, kde bola vytvorená pre súťaž v predikcii. Veľkosť dátovej množiny je približne 640MB. Celkový počet atribútov je 28, z toho je počet nominálnych atribútov 3 a počet numerických atribútov je 25. Celkový počet inštancií je 4446966. Dátová množina obsahuje len jednu prázdnu inštanciu, ale na druhú sú niektoré chybné hodnoty, ktoré vznikli dedukciou a analýzou dát. Dátová množina obsahuje anonymizované štatistické dáta naformátované, takým spôsobom, že každá inštancia obsahuje informácie hráča po ukončení hry. V dátovej množine sa uvažuje 5 rôznych skupinových variácií hry Solo - Hráči sú sami, Duo - Hráči hrajú vo dvojici, Squad - Hráči sú v skupine štyroch, Custom - Jedná sa o špeciálne podujatia, ktoré trvali len určitý čas.

Každá relácia hry je označená jedinečným reťazovým identifikátorom (atribút matchId). Hráči v rovnakých skupinách sú označení rovnakým reťazovým identifikátorom (atribút groupId). Hráči po smrti alebo ukončení hry sú umiestnení v

priečke (atribút winPlacePerc), ktoré sa odvíja percentuále od toho koľko tímov je stále živých a koľko tímov je eliminovaných. Ostatné atribúty uchovávajú informácie o rôznych štatistikách hráčov počas hry, ako napr. Rôzne zozbierané munície, oživenia, vzdialenosti spravné autom, pešo, plávaním atď.

2 Stručný opis prác iných autorov

Hlavným cieľom autorov článku [2] bolo sa zamerať na filtrovanie nerelevantných atribútov v dátovej množine a pracovať s atribútmi, ktoré sú najviac signifikantné pre predikovanú hodnotu. Na filtrovanie použili Weka Explorer, ktorý je hlavne rozdelený do dvoch kategórií: Hodnotiteľ atribútov (angl. Attribute Evaluator) a Prehľadávací algoritmus (angl. Search Algorithm). Výstupom z Weka zistili, že atribút walkDistance predstavuje najväčší dopad na cieľovú premennú.

Viacerí autori [1, 2] sa v rámci funkčného inžinierstva (angl. Feature Engineering) zamerali na zredukovanie a spájanie atribútov v dátovej množine. Nové atribúty predstavovali znormované hodnoty, spojené atribúty so spoločnou vlastnosťou.

Výsledkami v práci sa autorom [2] podarilo dokázať, že redukciou vlastností (angl. Feature reduction) podľa výšky korelačných hodnôt je možné zlepšiť výkon, aj keď to neplatí pre každý scenár. V ďalšej práci [1] autori zistili, že čo sa výkonu týka, tak Lineárne regresné modely sú najlepšie, ale pre najpresnejší výsledok je vhodnejší model XGBoost.

3 Opis metód

3.1 Predspracovanie a výber atribútov

Kolineárny výber atribútov - Pearsonova korelácia Korelácia predstavuje štatistickú metódu, ktorá zisťuje lineárny vzťah medzi dvomi alebo viacerými premennými (v našom prípade atribútmi). Atribúty, medzi ktorými je lineárny vzťah majú vysokú hodnotu korelácie.[1] V rámci tejto metódy výberu najvhodnejších atribútov som sa zameral na najvyššiu korelačnú hodnotu. Vypočítal som hodnotu korelácie a následne ich zoradil podľa výšky korelačnej hodnoty. Ako prvý experiment som určil hraničnú hodnotu 0.6 a vybral podmnožinu atribútov, ktoré korelovali s predikovaným atribútom viac než spomínaná hraničná hodnota. Tento spôsob výberu atribútov predstavoval slepú uličku, nakoľko každý z modelov dosiahol horšie výsledky, než pri dátovej množine so všetkými atribútmi. Aj keď bol prvotne použitý pri natrénovaní modelu Náhodný les, kde bol použitý veľmi nízky počet odhadov (angl. estimators), čo môže byť aj príčinou nízkych hodnôt.

[1] Zdroj: `https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf`

Feature importance Výhodou použitia gradient boostingu je to, že po vytvorení podporovaných stromov je relatívne jednoduché získať skóre dôležitosti pre každý atribút. Vo všeobecnosti dôležitosť poskytuje skóre, ktoré naznačuje, ako užitočná alebo cenná bola každá vlastnosť pri konštrukcii podporovaných rozhodovacích stromov v modeli. Čím viac sa atribút používa na prijímanie kľúčových rozhodnutí s rozhodovacími stromami, tým vyššia je jeho relatívna dôležitosť. Táto dôležitosť sa počíta explicitne pre každý atribút v súbore údajov, čo umožňuje zoradiť a porovnávať atribúty. Dôležitosť sa vypočíta pre jediný rozhodovací strom podľa množstva, za ktoré každý bod rozdelenia atribútov zlepšuje mieru výkonnosti, váženú počtom pozorovaní, za ktoré je uzol zodpovedný. Meradlom výkonnosti môže byť čistota (Gini index) použitá na výber deliacich bodov alebo iná konkrétnejšia chybová funkcia[2]. Teda zoznam vytvorený po natrénovaní modelu slúžil nie len ako výber atribútov, ale aj ako dôkaz kauzality pri výbere atribútov koreláciou.

RFE (Recursive Feature Elimination) - Táto metóda pracuje rekurzívnym mazaním atribútov a následným budovaním modelu na nezmazaných atribútoch. Ako vyhodnocovaciu metriku používa presnosť (angl. accuracy), pomocou spomínanej metriky zoradí atribúty podľa veľkosti hodnoty. Najdôležitejšie atribúty ohodnotí číslom 1 [3]. V tejto práci som použil na trénovanie gradientový strom LGBM, nakoľko dosahuje pri určitých parametroch najlepšie výsledky a zároveň poskytuje najoptimálnejší výkon. Nakoľko algoritmy modelov sa líšia, tak najlepším riešením by bolo spustiť RFE aj s ostatnými modelmi, ale tu som natrafil na slepú uličku, nakoľko pri Náhodnom lese sa to nedokáže natrénovať pri vhodných parametroch. Riešením, tejto slepej uličky, je použitie výstupu RFE s natrénovaným modelom LGBM, v prípade XGBOOST modelu je predpoklad, že sa atribúty veľmi nelíšia, nakoľko algoritmy sú takmer rovnaké. Otázkou je náhodný les, ktorý môže byť ako považovaný ako ďalšia práca, prípadne pokračovanie tejto práce.

3.2 DM metódy

Random Forest Náhodný les (angl. Random forest) patri medzi algoritmy učenia s učiteľom. Počas výpočtu vytvorí les s niekoľkými stromami, ktorých úspešnosť predikcie priamo úmerne rastie s počtom stromov v lese. Môže byť využitý na klasifikačné a regresné účely. Algoritmus začína výberom "k"faktorov z celkových "m"faktorov. S náhodne vybranými faktormi, nájdeme koreňový uzol, pomocou prístupu najlepšieho rozdelenia. Následne, vypočítame dcérske uzly pomocou prístupu najlepšieho rozdelenia aby sme získali cieľový list. Tento proces opakujeme a výsledkom je kombinácia viacerých rozhodovacích stromov, tvoriacich náhodný les [2].

Veľkou výhodou algoritmu Random forest je, že sa model nepretrénuje, nakoľko je tam dosť stromov, na spresnenie predikcie. Na natrénovanie spome-

[2] https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python
[3] https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b

nutého modelu je potrebné predpripraviť dátovú množinu a zmeniť kategorické atribúty na numerické. Je potrebné otestovať viac prístupov, nakoľko autori v článku [2] sa zamerali na natrénovanie modelu len s upravenou dátovou množinou, neporovnali ako vhodné je natrénovanie modelu s nezmenenými atribútmi resp. s atribútmi, ktoré boli zmenené iným spôsobom.

LGBM Light Gradient Boosting Machine je rámec, ktorý využíva algoritmy založené na základe stromových učení. Odlišuje sa od ostatných algoritmoch založených na stromovom učení tým spôsobom, že algoritmus vytvára strom pridávaním listov, nie pridávaním ďalších úrovní. Light GBM zvolí list s maximálnou zmenou straty k rastu. Pokiaľ rastie ten istý list, tak algoritmus môže viac redukovať stratu než algoritmy, ktoré pridávajú úrovne.

Light GBM je citlivý na pretrénovanie, obzvlášť pri malej dátovej množine. Pracuje s viac než 100 parametrami, z toho dôvodu je vyladenie komplikované ako pri ostatných algoritmoch. Autori v práci [2] sa zamerali na vyladenie 8 parametrov a dosiahli najlepší možný výsledok v porovnaní s ostatnými modelmi. Cieľom v tejto práci bude snaha o pokročilé vyladenie ďalších parametrov na zmenšenie chybovosti.

XGBOOST XBOOST optimalizuje regularizovaný cieľ pre lepšiu generalizáciu a vytvára aditívne riešenie [1]. Pri používaní zosilnenia gradientu pre regresné metódy, každý regresný strom mapuje vstupné dáta na svoje listy, ktoré obsahujú kontinuálne hodnoty. XGBOOST minimalizuje regulovanú objektívnu funkciu, ktorá kombinuje konvexnú stratu a pokutu pre komplexnosť modelu. Trénovanie prebieha iteratívne, pridávaním nových stromov, ktoré predikujú tantiémá alebo chyby prioritných stromov, ktoré sú neskôr kombinované s predošlými stromami s cieľom získať finálnu predikciu [3].

V práci autori [1], využili 5-fold krížovú validáciu na ladenie parametra 'eta' a zvolili výsledok s najlepším výkonom, rovnako pokračovali aj v prípade 'max_depth'. Zo všetkých modelov, tento bol zatiaľ najúspešnejší. Cieľom v tejto práci bude snaha o ďalšie vyladenie parametrov na zmenšenie chybovosti.

3.3 Vyhodnocovanie

Nakoľko sa jedná o predikovanie kontinuálnej premennej, tak je vhodným rozhodnutím použiť metriku na MAE (Mean Absolute Error), ktorý meria priemerný rozsah chýb v množine predikcií, bez uváženia ich smeru. Je to priemer nad testovacou množinou absolútnych vzdialeností medzi predikciou a aktuálnou pozorovanou hodnotou, kde všetky individuálne rozdiely majú rovnakú váhu. Vypočíta sa pomocou:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{1}$$

V niektorých prípadoch je metrika RMSE (Root mean squared error) vhodnejšia, pretože dáva väčšiu váhu veľkým chybám. RMSE je založený na kvadratickom

bodovaní, ktorý podobne ako MAE meria priemernú veľkosť chyby. Je to druhá odmocnina priemeru štvorcových rozdielov medzi predikciou a skutočným pozorovaním. Vypočíta sa pomocou:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{2}$$

4 Experimenty

4.1 Predspracovanie

V rámci pred spracovania dátovej množiny som sa zameriaval na vyriešenie identifikovaných problémov z predošlej správy. Niektoré problémy sa dali odstrániť, ďalšie pribudli počas pred spracovania.

Ako bolo avizované v predošlej správe, atribút **matchType** nezodpovedá presnému počtu hráčov v hre. V dátovej množine je atribút groupId, ktorý je rovnaký pre hráčov, hrajúcich v jednom tíme. Tým pádom som vytvoril nový atribút **countOfTeamPlayers**, kde je uložený počet rovnakých identifikácií hráčov.

Ďalším identifikovaným problémom boli špeciálne hry, ktoré boli dostupné iba nejaký čas v rámci testovania novej funkcionality. Aj keď tvoria malú časť dátovej množiny, stále môžu ovplyvniť trénovanie výpočtového modelu. Vo svojej podstate tie špeciálne hry sa dajú zovšeobecniť, nakoľko sa stále jedná o zoskupenie hráčov, buď do tímu po štyroch (squad) alebo do tímu po dvoch (duo) poprípade pre jedného hráča (solo). Na oficiálnej stránke hry som teda vyhľadal pravidlá a informácie o daných špeciálnych hrách a následne som ich zovšeobecnil v dátovej množine do 3 základných skupín (solo, duo, squad).

Počas pokročilej analýzy dátovej množiny, som zistil, že existujú inštancie hráčov, ktorí nespravili ani krok (vzdialenosť, ktorú prešli sa rovná 0), je to z hľadiska hry, nemožné. Jedná sa o hráčov, ktorí nejakým spôsobom odišli z hry alebo "boli preč od klávesnice (AFK - away from keyboard)", na druhej strane, našli sa aj inštancie, kde rovnako nespravili ani krok, ale majú nadpriemerné množstvo zabití. V takom prípade sa jedná o hráčov, ktorí podvádzajú. Obe typy dát patria do redundantnej skupiny a pridávajú šum do výpočtového modelu. Riešením bolo nájdenie a vymazanie všetkých inštancií, kde sa atribút **walkDistance** rovná 0.

Niektoré atribúty v dátovej množine vyjadrujú podobné veci natoľko, že je ich možné spojiť do jedného atribútu. Vzdialenosť, ktorú spraví hráč počas hry sa rozdeľuje na tri časti **walkDistance, rideDistance, swimDistance**. V korelačnej matici sa ukázalo, že atribút, ktorý je treba predikovať najviac koreluje so vzdialenosťou, ktorú hráč spravil pešo **walkDistance**, ostatné dva atribúty vyjadrujúce vzdialenosť mali malú koreláciu, ale vyjadrujú v podstate to isté. Zo spomenutého dôvodu, som vytvoril nový atribút, ktorý je sumou všetkých troch atribútov. Podobne je to aj v prípade zozbieraných vecí použitých na uzdravenie počas hry, tie údaje sú uložené pod atribútmi **boosts a heals**. Obidva atribúty

silno korelujú s atribútom, ktorý sa má predikovať, takže sa vytvoril nový atribút **totalHeals**, kde je súčet obidvoch spomenutých atribútov.

Nakoľko sa ide pracovať s regresnými modelmi, tak je potrebné kategorické atribúty transformovať na numerické. V dátovej množine sa nachádza jeden atribút, ktorý je pre užitočný a to **matchType**. Na transformáciu atribútu matchType som použil one hot encoding.

Niektoré atribúty ako **Id, groupId, matchId, matchDuration, winPoints, killpoints, rankPoints** boli z dátovej množiny vyradené, nakoľko po pred spracovaní nemali žiadny dopad na natrénovanie výpočtového modelu (nízka alebo žiadna korelácia, kategorické atribúty boli zmenené na numerické (groupId, matchId).

4.2 DM metódy

Prvotné experimenty - Medzi prvé experimenty som zaradil model Random forest, nakoľko nie je potrebné pokročilé ladenie parametrov na odskúšanie prvotného natrénovania bez väčšieho predspracovania a nehrozí pretrenévanie. Jediné úpravy dátovej množiny, ktoré prebehli pred natrénovaním boli zmazanie hráčov, ktorí nespravili žiadnu vzdialenosť a samozrejme one-hot encoding kategorického atribútu. Výsledok som vyhodnocoval pomocou R2 score. Trénovali a vyhodnocovali sa 4 úrovne estimátorov, 1,5,6,7. Samozrejme platí, že čím je viac úrovní, tým je model presnejší. Najlepšia hodnota dosiahnutá - 91.6% Z dôvodou porovnania som aplikoval predspracovanie nad dátovou množinou a spustil znovu trénovanie na rovnakých úrovniach, kde sa úspešnosť bohužiaľ zhoršila z 91% na 89.7%

Pokročilé experimenty - V rámci ďalších experimentov som sa venoval už priamo gradientovým stromom a snažil sa zistiť, ktorý z nich je bez pokročilého ladenia hyperparametrov najlepší čo sa týka výkonu a chybovosti. V oboch prípadoch zvíťazil LGBM. Zo spomenutého dôvodu som sa počas trénovania modelov najviac zameral na daný model, nakoľko sľubuje najlepšie výsledky. Už v tejto začiatočnej fáze bez pokročilých ladení parametrov dosahoval veľmi nízku chybovosť.

Počas experimentov a následne aj počas trénovania som sa zameriaval na trénovanie modelov na viacerých tvaroch dátovej množiny. *Featured dataset* - dátová množina, ktorá prešla predspracovaním, ktoré je detailnejšie rozpísané v časti predspracovanie v sekcii experimenty. *Nonfeatured dataset* - dátová množina, ktorá sa líši od originálneho tvaru tým, že sú odstránené atribúty ako *matchId, groupId a Id*, následne použitý one-hot encoding na kategorický atribút *matchType*.

Rôzne kombinácie dátových množín vytvorených pomocou algoritmov výberu atribútov - Rôzne kombinácie dátovej množiny *Featured dataset*, pre všetky algoritmy výberu atribútov som použil spomínanú dátovú množinu, nakoľko v prvotných experimentoch dosahovali modely najlepšie výsledky na tejto dátovej množine. Všetky dátové množiny po vytvorení boli zoserializované a uložené v tvare pickle objektov, z dôvodu rýchlejšieho načítavania.

Nastavenie parametrov V rámci nastavenia a vylaďovania parametrov som sa inšpiroval z vedeckých prác, ktoré sa zaoberali s dátovou množinou PUBG, tým pádom bolo zabránené pretrénovaniu gradientových stromov. Z dôvodu nedostatkovej výpočtovej sily som parametre použité v prácach pretransformoval, aby sa modely dokázali natrénovať aj s nižším výpočtovým výkonom. V tejto časti som natrafil na viaceré slepé uličky, ktorých výsledkom je nižšia presnosť než vo vedeckých prácach. Práve z dôvodu nedostatočnej výpočtovej sily, som využil algoritmy Random Search a Grid Search na prehľadávanie hyperparametrov. Rozdiel medzi Random search a Grid search, je v presnosti prehľadávania. Grid search trvá dlhšie, ale všetky možné kombinácie sú odskúšané, kým Random search vyberá náhodne kombinácie parametrov, čo mu umožňuje rýchlejší výsledok, ale menšiu presnosť. V rámci výberu vhodného algoritmu na prehľadávanie som natrafil na slepú uličku, čo sa týka výkonu prehľadávania parametrov. Dátová množina bola príliš rozsiahla na natrénovanie v optimálnom čase, z toho dôvodu sa vybrala určitá podmnožina (100 000 inštancií) na prehľadávanie. Nakoľko oba algoritmy sú vhodné v rozdielnych prípadoch, tak som sa rozhodol použiť Grid Search pri modeloch, ktoré v prvotných experimentoch predstavovali lepší výkon t.j. gradientové stromy a Random search pri Náhodnom lese, nakoľko jeho trénovanie trvá najdlhšie. Zameriaval som sa na prehľadávanie nasledujúcich parametrov.
Pri LGBM:

- num_leaves: [31, 128, 256],
- sub_feature: [0.5,0.7],
- bagging_fraction: [0.3,0.5,0.7],
- feature_fraction: [0.3,0.5,0.7],
- learning_rate: [0.1,0.05,0.03],
- n_estimators: [10000,5000,1000]

Pri XGBM:

- max_depth: [5,10,15],
- learning_rate: [0.1,0.01,0.5,0.03],
- n_estimators: [10000,5000,1000],
- alpha: [5,7,10],
- colsample_bytree: [0.3, 0.5, 0.1]

Náhodný les:

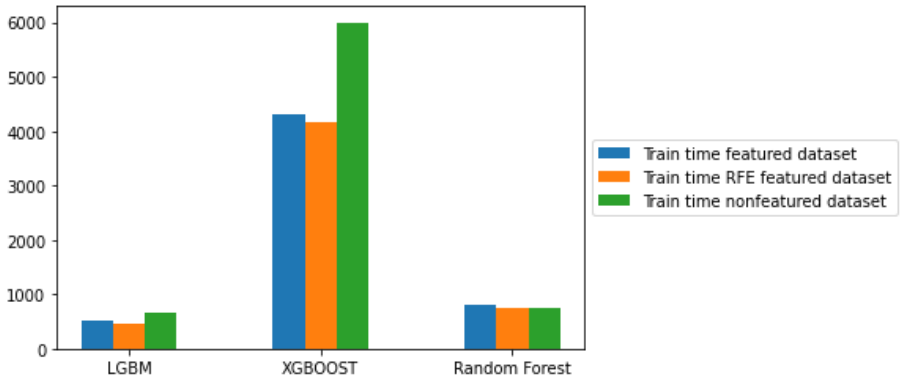
- max_depth: [80, 90, 100, 110],
- max_features: [2, 3],
- min_samples_leaf: [3, 4, 5],
- min_samples_split: [8, 10, 12],
- n_estimators: [50,100, 300, 500]

Trénovanie Po najdení najoptimalnejších parametrov,
Pre **LGBM**: 'sub_feature': 0.5, 'num_leaves': 31, 'n_estimators': 1000, 'learning_rate': 0.1, 'feature_fraction': 0.3, 'bagging_fraction': 0.7.
Pre **XGBOOST** n_estimators': 10000, 'max_depth': 5, 'learning_rate': 0.01, 'colsample_bytree': 0.5, 'alpha': 10
Pre **Nahodny les**: n_estimators: 100, min_samples_split: 8, min_samples_leaf: 3, max_features: 3, max_depth: 110

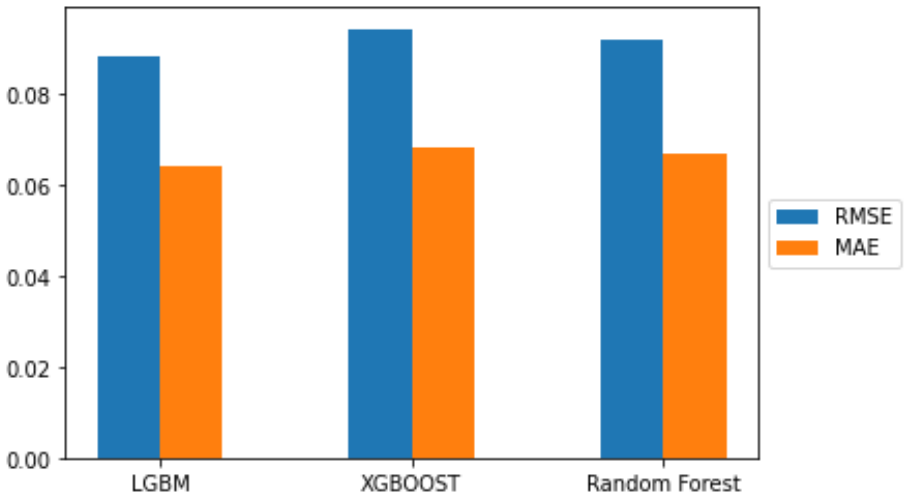
Testovanie Na testovanie som použil knižnicu train_test_split, dátovú množinu som delil v pomere 80-20, kde som model trénoval na 80% a testoval na 20%.

4.3 Vyhodnotenie

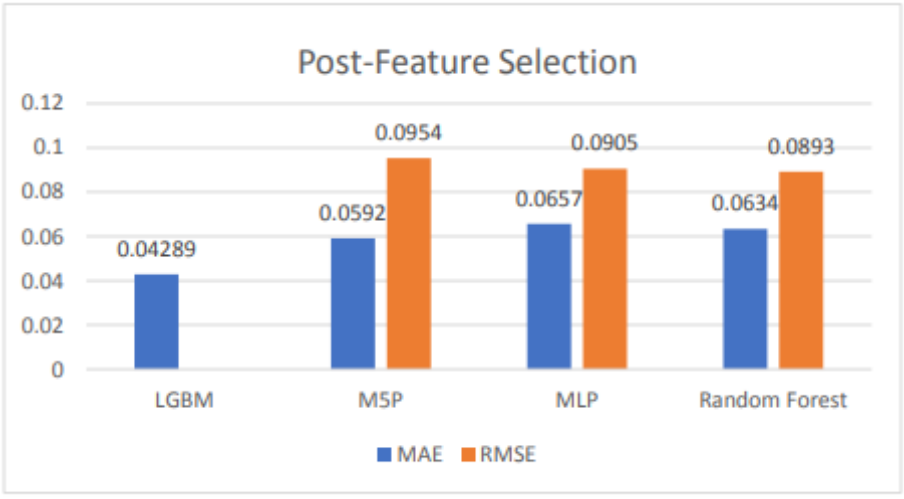
V ramci vyhodnotenia som sa zameriaval na porovnanie výkonu medzi modeli, porovnanie chybovosti medzi modelmi a zároveň medzi dátovými množinami. Z dôvodu veľkého počtu porovnaní a obrázkov do práce sú pridané dve najznačnejšie porovnania.



Obr. 1. Porovnanie výkonu medzi modelmi (čas)



Obr. 2. Porovnanie chybovosti medzi modelmi



Obr. 3. Výsledky v publikovanom vedeckom článku

5 Záver

V tejto práci som sa venoval objavovaniu znalostí v rámci dátovej množiny, ktorá sa zaoberá údajmi získanými z masívne populárnej hry PlayerUnknowns Battlegrounds. Dátová množina je robustná a poskytuje detailné informácie o každej relácií. Mojim cieľom bolo natrénovať modely, ktorých úspešnosť bude porovnateľná s publikovanými vedeckými prácami, ktoré sa tiež zaoberajú so spomínanou dátovou množinou. Po načítaní dátovej množiny som spravil úvodnú analýzu, kde som zistil, že dátová množina je takmer úplná a nie je problém s chýbajúcimi inštanciami. Na predspracovanie a výber atribútov som použil porovnanie korelácií pomocou Pearsonovej metódy, Rekurzívne eliminovanie atribútov a neskôr aj feature importance z natrénovaných modelov. Na trénovanie som použil 3 metódy, z ktorých sú dva gradientové stromy s použitím rôznych rámcov (XGBOOST, LGBM) a Náhodný les. Na vyhodnotenie som použil metriky MAE a RMSE, ale okrem chybovosti som sa zameriaval aj na výkon, ktorý som odmeral v časových jednotkách. Nakoľko je dôležité, aby sa stromy nepretrénovali a eliminovali sa podobné problémy, tak som využil Grid a Random search na prehľadávanie rôznych kombinácií hyperparametrov. Po nájdení najoptimálnejších parametrov som natrénoval spomínané modely a vyhodnotil ich. Najlepšie výsledky dosiahol model LGBM a najhoršie Náhodný les.

5.1 Ďalšia práca

Budúce práce s dátovou množinou PUBG môžu nadobudnúť viac tvarov. Jednou z najsľubnejších je použitie Neurónových sietí. Je to ďalšia oblasť objavovania znalostí, ktorá poskytuje čiernu skrinku a tým pádom aj zaujímavé výsledky na preskúmanie. Zaujímavosťou ďalšej práce by mohlo byť aj porovnanie natrénovaných gradientových stromov s neurónovými sieťami. Z dôvodu väčšej chybovosti a horšieho výkonu po porovnaní gradientových stromov s náhodným lesom, je vhodnejšie použiť na porovnanie gradientové stromy. Nakoľko gradientové stromy sa ľahšie trénujú ako neurónové siete, tak je možnosť porovnania viacerých rámcov gradientových stromov. Počas skúmania problémovej oblasti gradientových stromov som natrafil na ďalší rámec s názvom Catboost. Je to nový a vyladený rámec, ktorý v niektorých prípadoch dosahuje lepšie výsledky než iné rámce pre túto oblasť modelov.

Citácie

[1] Boming Fu et al. “Analysis upon video game Battle Royale–Prediction of Performance Ranking”. In: (2019). URL: https://about.2cni.com/doc/pubg_report.pdf.

[2] Brij Rokad et al. “Survival of the Fittest in PlayerUnknown BattleGround”. In: *CoRR* abs/1905.06052 (2019). arXiv: 1905.06052. URL: http://arxiv.org/abs/1905.06052.

[3] Amazon Web Services. *How XGBoost Works*. URL: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html (cit. 17. 04. 2020).