PA026 - Predikce výsledků zápasů NHL

Jakub Hruška

4. října 2020

Obsah

1 Úvod a podobné projekty **1**

2 Popis algoritmů a teorie **2**
2.1 Celková strategie 2
2.2 Základní datasety 3
2.3 Finální datasety 3
2.4 Křížová validace 3
2.5 Použité algoritmy strojového učení 4
2.6 První verze datasetu (v1) 5
2.7 Druhá verze datasetu (v2) 5
2.8 Třetí verze datasetu (v3) 6

3 Popis implementace **7**
3.1 Návod na instalaci a spuštění 8
3.2 Průběh projektu a výsledky 10

4 Závěr **13**

1 Úvod a podobné projekty

V oblasti predikce výsledků sportovních zápasů pomocí statistických modelů a strojového učení patří lední hokej (a konkrétně NHL) spíše k méně zastoupeným sportům, respektive ligám. Mnohem více pozornosti bylo v posledních 20 letech věnováno fotbalu, basketbalu či americkému fotbalu. Několik projektů aplikujících strojové učení na data z ledního hokeje sice vzniklo, velmi se však liší v mnoha aspektech – od zisku a zpracování dat, přes volbu proměnných a použitých metod a algoritmů, až po celkový koncept a definici řešeného problému.

Většina projektů minimálně zmiňuje možnost použití pro kurzové sázení, žádný s ním však nepočítá jako se svou pevnou součástí – vždy jde o okrajovou zmínku – což ovlivňuje celkové pojetí problému.

Co se úspěšnosti týče, projekty a články z oblasti ledního hokeje nedosahují tak vysokých čísel jako například ve fotbalu či basketbalu. To je jednak způsobeno menším zájmem o tuto oblast ze strany odborné komunity, dále pak charakteristikou ledního hokeje, který považovaný za velmi "náhodný"sport v porovnání například s basketbalem či fotbalem. Systém platových stropů a voleb nováčků v draftech navíc udržuje NHL velmi vyrovnanou v porovnání například s evropskými fotbalovými soutěžemi.

Z tohoto trendu se extrémně vymyká studie z roku 2019 [1], která dosahuje s přehledem přesnosti nad 90 %, což je i o 30 procentních bodů více než nejlepší ostatní studie. Osobně vidím problém v tom, že Gu ve své práci predikoval výsledek (výhra/prohra domácího týmu) na základě statistik ze hry, jako jsou střely na bránu, počet vyloučení apod. Nic z toho ovšem není známo před začátkem zápasu a tento postup tak nejde použít pro účely kurzového sázení. Druhým výrazným nedostatkem této studie, dle mého názoru, je použití náhodných zápasů k trénování. Vzniká tak situace, kdy testuje na "starých"zápasech model natrénovaný na zápasech z budoucnosti.

2 Popis algoritmů a teorie

2.1 Celková strategie

Od samého začátku bylo mým záměrem cílit na kurzové sázení. Tedy postavit model tak, aby předpovídal výsledky zápasů NHL z informací známých před jejich začátkem. I výstup modelu jsem přizpůsobil sázení, tedy výhra domácích/remíza/výhra hostů po základní hrací době. Prodloužení ani nájezdy jsem nebral v potaz, jelikož sázkové kanceláře nejčastěji vypisují kurzy právě na výsledek po základní hrací době. Uvažoval jsem i o tom, zda nepojmout problém jako regresi a predikovat skóre zápasu. Klasifikace mi však nakonec přišla jako jendodušší přístup.

Na rozdíl od všech ostatních článků, projektů a studií, které používají pro predikci různé týmové statistiky, mým záměrem bylo použít primárně soupisku daného týmu. Během sezóny se často stává, že se hráči zraní, trenér pozmění sestavu a podobně. Chtěl jsem tedy zjistit, zda bude možné úspěšně předpovídat výsledky zápasů ze soupisky, která se zveřejňuje několik hodin před zápasem.

Místo binárního: hráč za tým hraje/nehraje, jsem chtěl použít čas, který za zápas hráč reálně stráví ve hře (tzv. icetime). Je logické, že hráči, kteří v

zápase odehrají přes 20 minut mají obecně větší vliv na výsledek zápasu než ti, kteří odehrají 8 minut.

2.2 Základní datasety

Nejprve bylo nutné vytvořit základní datasety, které obsahují všechny potřebné informace. Ty jsem následně upravoval pro použití v jednotlivých modelech. Tyto úpravy zahrnují například předzpracování potřebné pro různé modely, počítání aktuální formy z posledních pár zápasů či upravenou verzi křížové validace. Nelze totiž dataset náhodně zamíchat a rozdělit na bloky (jako při běžné křížové validaci). Mohlo by se jednoduše stát, že by v tréninkové části datasetu byl zápas, který se ve skutečnosti odehraje až po zápasech z testovací části.

Postupně jsem upravoval svou strategii a výběry proměnných. Vznikly tak tři různé verze základního datasetu. Pro první verzi (v1) a třetí verzi (v3) jsem získal data ze sezón 2013/14 až 2019/20. Druhá verze se jevila hned z počátku slabě, netestoval jsem ji proto na větším datasetu a obsahuje data pouze od sezóny 2017/18. Počet zápasů v jedné sezóně se lehce liší. Do sezóny 2016/17 hrálo v NHL 30 týmů, každý odehrál 82 zápasů a celkově tedy bylo v každé z těchto sezón odehráno 1230 zápasů. Od sezóny 2017/18 přibyl 31. tým a sezóny tak obsahují 1271 zápasů. Sezóna 2019/20 byla ukončena předčasně kvůli pandemii COVID-19 a bylo v ní odehráno pouze 1082 zápasů. Počty proměnných se velmi lišily mezi jednotlivými verzemi.

2.3 Finální datasety

Základní datasety bylo pořeba upravit tak, aby na nich mohla být aplikována upravená křížová validace (viz sekce Křížová validace) a předzpracovat data pro potřeby strojového učení. To zahrnovalo různé strategie pro vypořádání se s ordinálními a nominálními prměnnými (např. datum a jméno trenéra) – převážně byla použita metoda *dummy variables* – dále pak standardizace či jiné způsoby přeškálování proměnných.

2.4 Křížová validace

Jelikož nebylo možné použít běžnou křížovou validaci, upravil jsem ji tak, že do trénovacího datasetu je vybráno několik po sobě jdoucích sezón a jako testovací je použita následující sezóna v pořadí. Je také možné přidat prvních n zápasů z testovací sezóny do trénovací a zahrnout tak do modelu změny, které se mohly odehrát mezi sezónami.

Opět bylo cílem se co nejvíce přiblížit realitě a kurzovému sázení, tedy vycházet pouze z již odehraných zápasů. Zároveň tento přístup minimalizuje vliv náhody v podobě výběru nevhodné sezóny. Se dvěma trénovacími sezónami máme až 5 foldů takto upravené křížové validace. Při třech trénovacích sezónách máme foldy 4.

Jednotlivé foldy (finální datasety) se od sebe mohou podstatně lišit. Je logické, že v sezónách okolo roku 2014 hrálo ligu mnoho jiných hráčů, než v roce 2019. Snažil jsem se vždy dataset co nejvíce "okrájet", aby už tak velmi široký a řídký dataset nedosahoval zbytečně ještě větších rozměrů. Do finálních datasetů jsem proto zahrnul jen ty hráče, kteří opravdu naskakovali do zápasů v daném období. Navíc jsem všechny hráče, kteří měli za dané období odehráno méně než 100 minut za sezónu sloučil do jednoho sloupce (respektive dvou – domácí a hosté, pojmenované *home other players* a *away other players*).

Dále bylo nutné sjednotit schéma tréninkového a testovacího datasetu. Každý hráč z trénovacího datasetu musí být obsažen i v testovacím. Pokud se však hráč objevil až v testovacím, byl zařazen do odpovídajícího sloupce *other players*. Mělo by to odpovídat situaci, kdy je v soupisce před zápasem uveden nováček, který hraje svůj první zápas v NHL. Pokud však hráč pravidelně hrál, byl například vyměněn a má nastoupit do prvního zápasu za nový tým, v datasetu již figuruje a není potřeba jej zařazovat do *other players* sloupce.

2.5 Použité algoritmy strojového učení

Pro explorativní analýzu datasetu jsem používal převážně *PCA, LDA a t-SNE*. Klasifikátory jsem použil následující: *Random Forest, Extreme Random Forest, Neuronové sítě* (feed forward variantu), *SVM, Naive Bayes* a *LDA*. U klasifikátorů, které mají nějaké parametry jsem aplikoval gridsearch pro nalezení nejvhodnějších hodnot. Zkoušel jsem i použít PCA a LDA na extrakci proměnných a Random Forest na selekci proměnných a pro vizualizaci jejich důležitosti.

U klasifikátorů, které to umožňují (Random Forest a neuronové sítě), jsem implementoval i rozhodování na základě jistoty daného modelu. Když by měl být tento systém použit pro reálné sázení, je velmi žádaná také informace o tom, jak moc si je model svou predikcí jistý. Umožní to vsázet pouze na jistější výsledky a neztrácet peníze na zápasech, kde se model jen lehce přiklonil na jednu ze stran velmi vyrovnaného duelu. U Ranodm Forest se jedná o práh počtu hlasujících stromů, zatímco u neuronových sítí jde o práh hodnoty vítězného výstupního neuronu s aktivační funkcí softmax ve výstupní vrstvě.

2.6 První verze datasetu (v1)

Postupem času jsem v různé míře upravoval jak základní dataset, tak způsob, jakým jsem z něj vytvářel finální datasety. Vykrystalizovaly z toho tři různé verze základního datasetu, které se od sebe poměrně zásadně liší. V následujících sekcích jsou rozebrány rozlišnosti mezi základními datasety jednotlivých verzí. Důvody změn mezi verzemi jsou podrobněji rozepsané v sekci Popis implementace.

Schéma základního datasetu v1 si můžete prohlédnout na obrázku 1, ukázku datasetu v1 pak na obrázku 2. Základní dataset má 3827 proměnných a je velmi řídký, jelikož obsahuje dva sloupce (doma/venku) pro všechny hráče, kteří odehráli alespoň jeden zápas v NHL od sezóny 2013/14. Do zápasu zasáhne přibližně 20 hráčů v každém z týmů (cca 40 hráčů/zápas celkem), je tedy zřejmě, že u zbytku budou samé nuly.

```
>>> df_v1.columns
Index(['year', 'month', 'day', 'hour', 'home team', 'away team', 'result',
       'home coach', 'away coach', 'home 8476412',
       ...
       'away 8476763', 'away 8474855', 'away 8471428', 'away 8471832',
       'away 8475771', 'away 8475833', 'away 8467910', 'away 8476846',
       'away 8473932', 'away 8469798'],
      dtype='object', length=3827)
```

Obrázek 1: Schéma základního datasetu v1. Krom obecných údajů o zápase (datum a čas, domácí tým, tým hostí a výsledek zápasu) obsahuje schéma i jména trenérů a dva sloupce pro každého možného hráče – jeden pro domácí zápasy daného hráče a druhý pro venkovní.

```
>>> df_v1.head()
            year  month  day  hour home team away team result  ... away 8471832 away 8475771
2013020001  2013     10    1    23       MTL       TOR   away  ...            0            0
2013020002  2013     10    2     0       CHI       WSH   home  ...            0            0
2013020003  2013     10    2     2       EDM       WPG   away  ...            0            0
2013020004  2013     10    2    23       PHI       TOR   away  ...            0            0
2013020005  2013     10    3     0       DET       BUF   home  ...            0            0

[5 rows x 3827 columns]
```

Obrázek 2: Ukázka základního datasetu v1. Hráč, který v daném zápase nastoupil např. za domácí tým, má ve svém sloupci (home + id hráče) čas v sekundách, který strávil na ledě (icetime).

2.7 Druhá verze datasetu (v2)

Ve druhé verzi jsem se snažil zmenšit dimenzi problému. Z dostupných dat jsem se pokusil algoritmicky odvodit, na jakém postu který hráč hraje. Místo

dvou sloupců pro každého hráče hrajícího v lize jsem vytvořil dva sloupce pro každý post v týmu (domácí: brankář, 1. formace – levé křídlo, 1. formace – centr, 1. formace – pravé křídlo, 1. obránce, 2. obránce, atd.; Vždy pro domácí i hostující tým.), v jednom sloupci bylo ID hráče, který v daném zápase na daném postu v daném týmu hrál. Ve druhém sloupci jeho icetime.

Tímto způsobem se počet dimenzí snížil z 3827 na 95 a výrazně zhoustl[1]. Schéma základního datasetu v2 je na obrázku 3, ukázka datasetu v2 pak na obrázku 4.

```
>>> df_v2.columns
Index(['year', 'month', 'day', 'hour', 'home_team', 'away_team', 'result',
       'home_coach', 'away_coach', 'home_G', 'home_D1', 'home_D1_time',
       'home_D2', 'home_D2_time', 'home_D3', 'home_D3_time', 'home_D4',
       'home_D4_time', 'home_D5', 'home_D5_time', 'home_D6', 'home_D6_time',
       'home_D7', 'home_D7_time', 'home_D8', 'home_D8_time', 'home_L1-LW',
       'home_L1-LW_time', 'home_L1-C', 'home_L1-C_time', 'home_L1-RW',
       'home_L1-RW_time', 'home_L2-LW', 'home_L2-LW_time', 'home_L2-C',
       'home_L2-C_time', 'home_L2-RW', 'home_L2-RW_time', 'home_L3-LW',
       'home_L3-LW_time', 'home_L3-C', 'home_L3-C_time', 'home_L3-RW',
       'home_L3-RW_time', 'home_L4-LW', 'home_L4-LW_time', 'home_L4-C',
       'home_L4-C_time', 'home_L4-RW', 'home_L4-RW_time', 'home_F13',
       'home_F13_time', 'away_G', 'away_D1', 'away_D1_time', 'away_D2',
       'away_D2_time', 'away_D3', 'away_D3_time', 'away_D4', 'away_D4_time',
       'away_D5', 'away_D5_time', 'away_D6', 'away_D6_time', 'away_D7',
       'away_D7_time', 'away_D8', 'away_D8_time', 'away_L1-LW',
       'away_L1-LW_time', 'away_L1-C', 'away_L1-C_time', 'away_L1-RW',
       'away_L1-RW_time', 'away_L2-LW', 'away_L2-LW_time', 'away_L2-C',
       'away_L2-C_time', 'away_L2-RW', 'away_L2-RW_time', 'away_L3-LW',
       'away_L3-LW_time', 'away_L3-C', 'away_L3-C_time', 'away_L3-RW',
       'away_L3-RW_time', 'away_L4-LW', 'away_L4-LW_time', 'away_L4-C',
       'away_L4-C_time', 'away_L4-RW', 'away_L4-RW_time', 'away_F13',
       'away_F13_time'],
      dtype='object')
```

Obrázek 3: Schéma základního datasetu v2. Pro každý post v týmu je ve schématu jeden sloupec pro ID hráče hrajícího na daném postu a druhý pro jeho icetime v zápase. Vše ve variantě pro domácí i hostující tým.

2.8 Třetí verze datasetu (v3)

Ve třetí verzi jsem úplně opustil ideu icetimu jednotlivých hráčů a uchýlil jsem se k týmovým statistikám. Z dostupných záznamů o zápasech jsem získal všechny základní statistiky z každého zápasu[2]. Nepřidával jsem však

[1] Expandováním do dummy variables bychom ovšem o toto zlepšení přišli, jelikož cca polovina sloupců obsahuje ID hráčů, kterých je mnoho.

[2] Např.: GF – goals for – vstřelené góly; GF-PP – goals for - power play – vstřelené góly v přesilových hrách; SA – shots against – střely na brány proti; ...

```
>>> df_v2.head()
            year  month  day  hour home_team away_team  ... away_L4-C away_L4-C_time
2017020001  2017     10    4    23       WPG       TOR  ...   8475098            697
2017020002  2017     10    5     0       PIT       STL  ...   8470803            676
2017020003  2017     10    5     2       EDM       CGY  ...   8470162            630
2017020004  2017     10    5     2       SJS       PHI  ...   8477290            442
2017020005  2017     10    5    23       BOS       NSH  ...   8475714            764

[5 rows x 95 columns]
```

Obrázek 4: Ukázka základního datasetu v2.

pokročilé statistiky, jako jsou Corsi, Fenwick apod., které jsou v posledních letech mezi novináři velmi populární. Jedná se vždy o lineární kombinaci některých již přítomných statistik (většinou poměry střel na bránu), očekával jsem tedy, že to pro modely nebude žádná nová informace a budou schopné si ji vytvořit kombinací jiných[3].

Schéma základního datasetu v3 je na obrázku 5, ukázka datasetu v3 pak na obrázku 6.

```
>>> df_v3.columns
Index(['year', 'month', 'day', 'hour', 'home_team', 'away_team', 'result',
       'home_coach', 'away_coach', 'home_GF', 'home_GF-5on5', 'home_GF-PP',
       'home_GF-SH', 'home_FO-T', 'home_FO-W', 'home_SF', 'home_SF-5on5',
       'home_SF-PP', 'home_SF-SH', 'home_PP-O', 'home_PIM', 'home_BLK',
       'home_TAW', 'home_GAW', 'home_HIT', 'home_PK-O', 'home_GA',
       'home_GA-5on5', 'home_GA-PP', 'home_GA-SH', 'home_SA', 'home_SA-5on5',
       'home_SA-PP', 'home_SA-SH', 'away_GF', 'away_GF-5on5', 'away_GF-PP',
       'away_GF-SH', 'away_FO-T', 'away_FO-W', 'away_SF', 'away_SF-5on5',
       'away_SF-PP', 'away_SF-SH', 'away_PP-O', 'away_PIM', 'away_BLK',
       'away_TAW', 'away_GAW', 'away_HIT', 'away_PK-O', 'away_GA',
       'away_GA-5on5', 'away_GA-PP', 'away_GA-SH', 'away_SA', 'away_SA-5on5',
       'away_SA-PP', 'away_SA-SH'],
      dtype='object')
```

Obrázek 5: Schéma základního datasetu v3. Krom základních informací o zápase jsou pro domácí i hostující tým ve schématu obsaženy všechny týmové statistiky, které NHL oficálně uvádí.

3 Popis implementace

Celý projekt je napsaný v *Pythonu 3.6* za použití knihovny *Pandas* pro práci s datasety, *Scikit-learn* poskytla metody strojového učení krom neuronových sítí, na které jsem použil *Keras* s *Tensorflow* backendem.

[3]Po debatě při finální prezentaci zde vidím slabé místo, které jsem podcenil a kde by mohl být prostor pro zlepšení. Manuálně vytvořit tyto proměnné a například jimi nahradit několik základních by mohla být cesta ke zlepšení úspěšnosti.

```
>>> df_v3.head()
            year  month  day  hour home_team away_team result  ... away_GA-5on5 away_GA-PP  away_GA-SH  away_SA
2013020001  2013     10    1    23       MTL       TOR   away  ...            3          0           0       37
2013020002  2013     10    2     0       CHI       WSH   home  ...            4          0           1       35
2013020003  2013     10    2     2       EDM       WPG   away  ...            2          0           2       38
2013020004  2013     10    2    23       PHI       TOR   away  ...            0          0           1       32
2013020005  2013     10    3     0       DET       BUF   home  ...            2          0           0       34

[5 rows x 59 columns]
```

Obrázek 6: Ukázka základního datasetu v3.

Významná část práce byla vytvoření frameworku, který získává data o zápasech z oficiálního NHL API, které je sice poměrně velmi rozsáhlé, nicméně mizerně zdokumentované. Existuje několik neoficiálních dokumentací z komunity. Nejsou však kompletní a místy jsou i neaktuální. Framework dále vytváří základní datasety, páruje jména hráčů s jejich ID (oficiálním NHL ID hráče), upravuje základní datasety do podoby finálních datasetů a produkuje připravené foldy (*seasonal splits*) pro křížovou validaci. Vše s možností volby několika parametrů – počty trénovacích sezón, počet úvodních zápasů přeřazených z testovací sezóny k trénovacím, použití či nepoužití dummy variables na týmy, trenéry, hráče, metoda standardizace a délka sledované formy (viz sekce Průběh projektu a výsledky).

Vznikl tak robustní framework, který poskytoval přípravu datasetů bez nutnosti složitě upravovat kód – jen změnou několika parametrů. Značně se tak zlehčila samotná učící část, jelikož porovnávání vlivu různých úprav datasetu bylo částečně zautomatizované.

3.1 Návod na instalaci a spuštění

Na adrese https://github.com/jakub-h/nhl-predict/ je přístupný repozitář s kódem. Repozitář neobsahuje datasety (ani základní ani finální), jelikož jsou příliš veliké. JSON soubory jednotlivých her získané z NHL API zahrnuty jsou, stejně tak tabulky výsledků pro v1 a v3.

Kód projektu je rozdělen do několika souborů. Jsou jimi jednak DatasetManager soubory pro každou verzi v1-v3 (`dataset_manager_*.py`). Ty obsahují třídu DatasetManager, která se stará o práci s datasety. Lze pomocí ní stáhnout a zpracovat jednotlivé zápasy z požadované sezóny z NHL API:

```
def get_season(self, season_year, preseason=False):
```

vytvořit základní dataset:

```
def create_base_dataset(self, filename="base_dataset_v3.csv"):
```

vytvořit a připravit na disk jednotlivé foldy (*seasonal splits*) upravené křížové validace podle sezón:

```
def create_seasonal_split(self, test_season,
                          first_games_to_train,
                          num_of_train_seasons, form_length,
                          dummy, home_rel, minmax):
```

a nakonec i tyto foldy zprostředkovat během průchodu křížovou validací:

```
def get_seasonal_split(self, test_season,
                       first_games_to_train,
                       num_of_train_seasons, form_length,
                       dummy, home_rel, minmax):
```

Zbytek kódu je podstatně méně strukturovaný a jedná se spíše o scripty, které během průběhu projektu živelně vznikaly, byly upravovány a některé i zanikaly. Po rozčlenění projektu do verzí jsem se snažil udržet jednotlivé verze oddělené a označené. Ve třetí verzi, které jsem se nakonec věnoval nejvíce, jsem rozdělil a zachoval i samostatné soubory pro každý z použitých klasifikátorů. Můžete v nich narazit na metody pro aplikaci křížové validace (a případné filtrování jistých výsledků), jako je např. tato pro RF ve v3:

```
def seasonal_crossval(params, first_games,
                      num_of_training_seasons,
                      form_length, dummy, home_rel,
                      minmax, n_jobs, use_class_weights):
```

Ty mají jako argument už finální parametry použitého klasifikátoru. Dále jsou zde metody pro gridsearch prohledávání prostoru parametrů:

```
def seasonal_grid_search(clf_param_grid, first_games,
                         conf_factors, form_lengths):
```

které berou jako parametr tabulku možných hodnot pro gridsearch. Např:

```
param_grid = {
    'n_estimators': [50, 200],
    'max_depth': [50, 300, None],
    'min_samples_leaf': [1, 5, 10],
    'min_samples_split': [2, 5, 10],
    'max_features': ['sqrt', None],
    'criterion': ['gini', 'entropy']
}
first_games = [0, 50, 100]
confidence_factors = [0.5, 0.65]
form_lengths = [2, 5, 10]
```

Ve složce results jsou csv soubory s některými výsledky pro v1 a v3.

3.2 Průběh projektu a výsledky

V první verzi datasetu (icetime každého hráče, velmi široký a řídký dataset) vycházel ze všech klasifikátorů nejlépe Random Forest. Dokonce až velmi dobře. Accuracy s přehledem přesahovala 80 % pro většinu metaparametrů modelu. Při detailnějším průzkumu bylo navíc patrné, že nejlépe model predikuje remízy. Ty jsou však často mezi odborníky považovány za nejproblematičtější pro předpověď. Po odfiltrování malé části nejvíce nejistých předpovědí (viz Použité algoritmy strojového učení) se precision u remíz blížila téměř ke 100 %.

Začal jsem se proto pídit po důvodech tohoto chování. LDA prostor potvrdil jednoznačnou diverzitu mezi remízami a zbytkem zápasů. Zatímco výhry domácích i hostů tvořily jeden velký a těžko oddělitelný mrak, remízy tvořily klastr, který se s tímto mrakem překrýval jen minimálně.

Při pohledu do významnosti jednotlivých proměnných podle Random Forestu byli všichni brankáři seřazeni na vrchních pozicích. Pod nimi byly hvězdy svých týmů, jako jsou Alexandr Ovečkin, Sidney Corsby nebo Patrice Bergeron. Pak následovali ostatní hráči a seznam uzavíraly základní informace o zápase (datum, čas začátku, týmy, ...).

Po dlouhém pátrání jsem zjistil, že jsem udělal chybu a vkládal jsem do modelů reálný čas, který hráči v daném zápase odehráli. Jednak tuto informaci samozřejmě neznáme před začátkem zápasu. Navíc je z toho v drtivé většině zápasů zřejmé, zda došlo k remíze, či nikoliv. Pokud brankář má odehráno více než 60 minut, je jasné, že došlo na prodloužení. Proto byli všichni brankáři tak důležití pro rozhodování Random Forestu.

Upravil jsem tedy způsob, jakým se vytváří finální dataset ze základního. Reálně odehraný čas jsem nahradil průměrným icetimem z doposud odehraných zápasů v aktuální sezóně. Jde o předzápasový odhad icetimu hráčů uvedených na soupisce. Následoval drastický propad accuracy ke 42 %.

Pro takovýto dataset se mi nepodařilo nalézt klasifikátor, jeho parametry či předzpracování, které by jakkoliv významě zvýšilo accuracy. V LDA či PCA prostorech dataset vypadal úplně náhodně (jeden rovnoměrně promíchaný mrak) a Random Forest ukazoval u všech proměnných skoro stejnou (extrémně malou) významnost.

Ve druhé verzi jsem chtěl zmenšit dimenzionalitu datasetu, nicméně jsem chtěl stále zachovat přístup k problému přes soupisky týmů. Jak je popsáno v sekci Druhá verze datasetu (v2), nahradil jsem sloupce pro konkrétní hráče sloupci pro jednotlivé posty. Přesný post hráče v daném zápase není z NHL API dostupný, proto jsem musel posty odhadovat na základě odehraného času v zápase. Předpokládal jsem, že hráči ze stejné formace budou mít podobné icetimy a že první formace odehraje více času než druhá, ta více

než třetí atd.

Nejsem si jistý, zda to bylo způsobeno samotným návrhem datasetu, nebo alogritmem přiřazování lidí do formací, ale výsledky byly ještě horší než u v1. Již od prvních testů byla accuracy pod 35 % a k výsledkům v1 se vůbec nepřiblížila. Přestal jsem se jí proto věnovat poměrně brzy.

Po zklamání ze zamýšleného alternativního přístupu k problému pomocí soupisek a icetimu jsem se uchýlil k tradičnějšímu pojetí – týmovým statistikám. Z NHL API jsem získal statistiky jednotlivých zápasů, dopočítal jsem z dostupných základních statistik některé další (počty střel při hře 5 na 5 a v přesilových hrách apod.) a vytvořil jsem základní dataset s pozápasovými statistikami z každého zápasu. Ve finálních datasetech figuroval kromě průměru dané statistiky z dosavadního průběhu sezóny i průměr za období posledních několika zápasů – uživatelsky volitelný parametr. Ten měl reflektovat aktuální formu týmu. Bylo rovněž možné si přepnout mezi statistikami v absolutních hodnotách pro oba týmy a v relativních vztažených k domácímu týmu.

Výsledky opět nebyly příliš oslnivé, nicméně lehký posun oproti v1 zde byl. Accuracy se přiblížila u Random Forestu, který opět vycházel lehce lépe než ostatní, ke 45 %. Jen o trochu horších výsledků dosahovaly neuronové sítě a SVM. Prohledávání metaparametrů pomocí grid search nepřineslo výrazné zlepšení. Pro parametry Scikit-learn implementace Random Forestu platí následující:

- *n_estimators*: Nepřekvapivě, větší množství stromů dává lepší výsledky za cenu vyššího času.
- *criterion*: Entropy lehce předčilo gini, za cenu většího času (viz obr. 7). Rozdíl je však dostatečně výrazný pouze u profiltorvaných zápasů.

```
>>> rf.groupby('criterion').mean().iloc[:, [10, 11, 12, 13, 23, 34, 35]]
           cv acc mean  cv acc ci  cv prec mean  cv prec ci   ovr acc  sure ovr acc  train time
criterion
entropy       0.435918   0.015140      0.379250    0.046788  0.435821      0.535697   24.052959
gini          0.434782   0.015044      0.376537    0.044679  0.434701      0.508571   13.210263
```

Obrázek 7: Výsledky gridsearch křížové validace zgrupované podle parametru *criterion*.

- *max_depth*: Nemá na výsledek téměř žádný vliv (viz obr. 8).
- *min_samples_split*: Vliv opět není příliš výrazný. Vyšší hodnoty vykazují lehce lepší accuracy po vyfiltrování nejistých výsledků (viz obr. 9).

```
>>> rf.groupby('max depth').mean().iloc[:, [9, 10, 11, 12, 22, 33, 34]]
           cv acc mean  cv acc ci  cv prec mean  cv prec ci   ovr acc  sure ovr acc  train time
max depth
10.0          0.441464   0.016395      0.368779    0.063570  0.441450      0.554978    3.331316
50.0          0.435965   0.017231      0.370483    0.044579  0.435895      0.534244    3.694146
300.0         0.435642   0.016077      0.374526    0.046820  0.435609      0.532330    3.690372
None          0.436046   0.016446      0.375827    0.051534  0.435987      0.530595    3.735619
```

Obrázek 8: Výsledky gridsearch křížové validace zgrupované podle parametru *max_depth*.

```
>>> rf.groupby('min samples split').mean().iloc[:, [8, 9, 10, 11, 21, 32, 33]]
                  cv acc mean  cv acc ci  cv prec mean  cv prec ci   ovr acc  sure ovr acc  train time
min samples split
2                    0.434944   0.015175      0.377294    0.044344  0.434831      0.514842   18.725613
5                    0.435088   0.015011      0.377694    0.045234  0.435022      0.515697   18.688176
10                   0.436019   0.015090      0.378692    0.047623  0.435930      0.535863   18.481044
```

Obrázek 9: Výsledky gridsearch křížové validace zgrupované podle parametru *min_samples_split*.

- *min_samples_leaf*: Zde to vypadá, že čím větší listy, tím lepší accuracy – jak v rámci křížové validace, tak celková normální i celková na vyfiltrovaných jistých zápasech. Oproti tomu precision v křížové validaci s velikostí listů klesá. Při pohledu na intervaly spolehlivosti však nemůžeme z tohoto pozorování vyvodit žádný jednoznačný závěr (viz obr. 10).

```
>>> rf.groupby('min samples leaf').mean().iloc[:, [8, 9, 10, 11, 21, 32, 33]]
                 cv acc mean  cv acc ci  cv prec mean  cv prec ci   ovr acc  sure ovr acc  train time
min samples leaf
1                    0.429245   0.015019      0.380294    0.034782  0.429157      0.492409   21.299123
5                    0.436155   0.015246      0.378852    0.046041  0.436074      0.524692   18.645965
10                   0.440651   0.015011      0.374535    0.056377  0.440551      0.549300   15.949746
```

Obrázek 10: Výsledky gridsearch křížové validace zgrupované podle parametru *min_samples_leaf*.

- *max_features*: Rozdíly v accuracy a precision nejsou dostatečně výrazné, aby vyvážily obrovksý nepoměr v čase tréninku. Omezení počtu proměnných při vytváření nového uzlu stromu se jeví jako výrazně časově výhodnější bez prokazatelných kvalitativních úhon (viz obr. 11).

Popis vybrané části tabulky výsledků: *cv_acc_mean* – průměrná accuracy získaná z upravené křížové validace (viz Křížová validace); *cv_acc_ci* – interval spolehlivosti průměru z křížové validace; *cv_prec_mean* – průměr precision z křížové validace; *cv_prec_ci* – interval spolehlivosti průměru precision z křížové validace; *ovr_acc* – celková accuracy – agregovaná přes všechny foldy křížové validace; *sure_ovr_acc* – celková accuracy ze zápasů, kde jistota modelu přesahuje požadovaný práh; *train_time* – čas potřebný k naučení

```
>>> rf.groupby('max features').mean().iloc[:, [9, 10, 11, 12, 22, 33, 34]]
              cv acc mean  cv acc ci  cv prec mean  cv prec ci   ovr acc  sure ovr acc  train time
max features
None             0.432695   0.015745      0.381025    0.041524  0.432589      0.528096   31.965851
sqrt             0.438005   0.014440      0.374761    0.049943  0.437933      0.516171    5.297372
```

Obrázek 11: Výsledky gridsearch křížové validace zgrupované podle parametru *max_features*.

modelu. Na obrázcích 7-11 jsou vždy zobrazeny průměry daných hodnot pro vybranou skupinu (např. průměrný čas tréninku pro hodnotu *sqrt* parametru *max_features*).

4 Závěr

Ukázalo se, že předpověď výsledků sportovních zápasů a hokeje (NHL) obzvláště patří mezi náročné klasifikační úlohy. Obzvláště, pokud se snažíme predikovat výsledek opravdu pouze na základě informací a statistik známých před začátkem zápasu. Z porovnání mnou aplikovaných přístupů vyplývá, že samotný icetime a složení soupisky nenesou dostatek informací k úspěšnému rozhodování. Tradiční a běžně používanější přístup pomocí týmových statistik se zdá být perspektivnější. Nicméně do stavu, kdy by se dalo na základě předpovědí modelu vydělávat na kurzovém vsázení, chybí ještě hodně práce.

Oblast, kde je podle mého názoru nejvíce prostoru ke zlepšení, je feature engeneering. Výběr a úprava proměnných, které pro klasifikaci použijeme, a hlavně zisk potřebných dat pro tyto proměnné jsou klíčové faktory pro úspěch. Krom vytváření frameworku na práci s datasety mi tato oblast zabrala jednoznačně nejvíce času a stále mám pocit, že jsem neprozkoumal a nevyzkoušel ani polovinu toho, co mne napadlo a chtěl bych vyzkoušet.

Jako nejlepší se nakonec jevila verze v3 s týmovými statistikami a Random Forest s následujícími parametry:

- počet stromů: 1000
- criterion: entropy
- maximální hloubka stromů: neomezeno
- minimální velikost listů: 8
- počet zápasů ze začátku testovací sezóny převedený k trénovacím: 0
- počet trénovacích sezón: 6
- počet posledních odehraných zápasů (délka aktuální formy): 5

- dummy variables: ne
- hodnoty relativně vztažené k domácímu týmu: ne
- minmax škálování: ne

```
## DatasetManager V3 ##: Initialization
## Seasonal CV ##: RandomForest ({'n estimators': 1000, 'criterion': 'entropy', 'm
## form: 5; dummy: False; home rel: False; minmax: False; class weights: False
test season: 2017 (first 0 games to train); train seasons: 6 ---> done in 16.09 s
test season: 2018 (first 0 games to train); train seasons: 6 ---> done in 20.60 s
test season: 2019 (first 0 games to train); train seasons: 6 ---> done in 24.92 s
========== y true ==========
home    1568
away    1239
draw     817
Name: true, dtype: int64
========== y pred ==========
home    2738
away     886
Name: pred, dtype: int64
              precision    recall  f1-score   support

        away     0.4424    0.3164    0.3689      1239
        draw     0.0000    0.0000    0.0000       817
        home     0.4635    0.8093    0.5894      1568

    accuracy                         0.4583      3624
   macro avg     0.3020    0.3752    0.3195      3624
weighted avg     0.3518    0.4583    0.3812      3624

========== y true sure (conf f = 0.50) ==========
home    141
away     67
draw     54
Name: true, dtype: int64
========== y pred sure (conf f = 0.50) ==========
home    257
away      5
Name: pred, dtype: int64
              precision    recall  f1-score   support

        away     0.2000    0.0149    0.0278        67
        draw     0.0000    0.0000    0.0000        54
        home     0.5447    0.9929    0.7035       141

    accuracy                         0.5382       262
   macro avg     0.2482    0.3359    0.2438       262
weighted avg     0.3443    0.5382    0.3857       262
```

Obrázek 12: Výsledky nejlepšího klasifikátoru.

Reference

[1] W. Gu, K. Foster, J. Shang, and L. Wei, "A game-predicting expert system using big data and machine learning," *Expert Syst. Appl.*, vol. 130, pp. 293–305, 2019.