Odabir Fantasy Premier League tima pomoću metoda mašinskog učenja i linearnog programiranja

Projekat u okviru kursa Mašinsko učenje
Matematički fakultet

Luka Milošević, Stefan Lazović
lukamilosevic11@gmail.com, skefi96@gmail.com

27. avgust 2020.

Sažetak

Učinak fudbalera u engleskoj Premijer ligi (eng. *Premier league*) uveliko varira od sezone do sezone, čak i od utakmice do utakmice. Očigledno je da će metoda koja bi uspela da prognozira i analizira budućnost ovih igrača na terenu dosta pomoći menadžerima pri odabiru tima. U simuliranom okruženju poput Fantazi Premijer lige (eng. *Fantasy Premier league*) entuzijasti iz celog sveta učestvuju i upravljaju igračima tokom cele sezone pokušavajući da stvore idealni tim koji će doneti najviše poena. Zbog dinamične prirode poena, ne postoji poznat pristup kreiranja idealnog tima. U ovom projektu pokušavamo da rešimo taj problem koristeći različite tehnike mašinskog učenja i pomerajući težinski prosek (eng. *weighted moving average*) i eksponencijalni pomerajući težinski prosek (eng. exponentially weighted moving average) kao pripremu podataka za predviđanje vremenskih serija poena igrača i naknadno maksimizovanje poena igrača i odabir idealnog tima pomoću linearnog programiranja (eng. linear programming). Predviđanja se vrše u odnosu na prethodne utakmice u tekućoj sezoni.

Keywords— Time series Forecasting, Fantasy Premier League, English Premier League, Linear Optimisation

Sadržaj

1 Uvod

Fudbal (eng. *football*) je kolektivni sport koji se igra između dve ekipe, sastavljene od po jedanaest igrača. Fudbal je trenutno najpopularniji sport na svetu. Igra se u preko 200 zemalja. Mogu ga igrati ljudi svih godišta i oba pola. Često se o fudbalu govori kao o "najvažnijoj sporednoj stvari na svetu"[1]. U savremeno doba, igra je dostigla fazu u kojoj svaka od aktivnosti igrača je pod stalnim nadzorom. Samim tim, predviđanje njihovih nastupa od najvećeg je značaja koje može dovesti do uspeha pojedinih timova. Postoje različite mere kvaliteta nekog fudbalera, naime, golovi za napadače, odbrane za golmane, asistencije za vezni red ili standardizovani sistem poena. Ovaj projekat se bavi gore navedenim, prognozom poena igrača na osnovu toga kako je odigrao u poslednjih n kola.

1.1 Fantasy Premier League

Premijer liga je najviša profesionalna fudbalska liga Engleske. U njoj se takmiči 20 klubova sa ostrva za titulu prvaka Engleske. Sezona traje od avgusta do maja u kojoj svaki tim odigra po 38 mečeva. Premijer liga je nedavno postala najgledanija sportska liga [2]. Fantasy Premier League (FPL) predstavlja simulaciju Premijer lige i daje šansu entuzijastima širom sveta da naprave svoj tim od 15 igrača i takmiče se na svetskom nivou i zauzmu neko od mesta na vrhu tabele. Organizovana od strane EA Sports, FPL na kraju sezone pruža jako zadivljujuće nagrade. Projekat može pomoći početnicima da naprave svoj idealni tim dok ne steknu iskustvo.

1.2 Formulacija idealnog tima

Kao što je spomenuto u 1.1, jedini cilj FPL je da održava tim od 15 igrača čiji će učinak odrediti poziciju na tabeli. S obzirom na dat fiksni budžet od £100 miliona, tim mora sadržati 2 golmana, 5 defanzivaca, 5 veznih igrača i 3 napadača. Rezultat tima u poenima se dobija samo od igrača koji su bili na terenu i naravno cilj je postići što bolji rezultat. Poeni se sabiraju kroz kola i na taj način se rangiraju timovi na tabeli.

2 Podaci

Podaci korišćeni za treniranje modela mašinskog učenja preuzeti su sa *GitHub* repozitorijuma [7], podaci sadrže različite statistike i pojedinačne učinke za svakog igrača i svaki tim u Premijer ligi. Postoji dosta različitih datoteka, one koje smo mi koristili su:

- gw.csv 1

  Direktorijum za svakog igrača sadrži datoteku gw.csv u kojoj se nalazi učinak igrača u svim kolima koja su odigrana.

assists	clean_sheets	ict_index	minutes	...	saves	selected	total_points	was_home
1	0	6.6	90		6	26492	10	False
0	1	2.2	90		3	90239	8	False
0	1	2.0	90		4	108434	9	True

Tabela 1: gw.csv

- teams.csv 2
  Sadrži informacije o timovima, njihovu jačinu.

id	name	strength	...	strength_overall_away	strength_overall_home
10	Liverpool	5		1350	1340
11	Man City	5		1340	1330
12	Man Utd	4		1300	1220

Tabela 2: teams.csv

- players_raw.csv 3
  Sadrži dodatne informacije za igrače, kao na primer na kojoj poziciji igra.

element_type	first_name	goals_scored	...	now_cost	second_name	total_points
3	Kevin	11		108	De Bruyne	226
1	David	0		53	de Gea	126
3	Nemanja	0		48	Matic	49

Tabela 3: players_raw.csv

Za predviđanje vremenskih serija podaci od ranijih godina su jako bitni, u ovom radu korišćeni su samo podaci iz tekuće godine jer postoji preko 600 igrača za koje postoje podaci iz 38 kola čime se dobije blizu 20000 instanci.

Korišćeni atributi:

assists broj ostvarenih asistencija na jednoj utakmici

bonus nagradni poeni koje dobijaju najboljih pet u timu

clean_sheets dodatni poeni za golmane, odbranu i sredinu ako ekipa ne primi gol

creativity mera kreativnosti igrača

element_type pozicija na kojoj igrač igra

goals_conceded koliko je golova primio igračev tim

goals_scored koliko golova je dao igrač

influence uticaj igrača na igru

minutes koliko minuta je igrač odigrao na meču

strength jačina protivničkog tima

own_goals autogolovi

penalties_missed promašeni penali

penalties_saved odbranjeni penali

red_cards broj crvenih kartona

saves odbrane golmana

selected koliko ljudi je u igrici izabralo igrača

team_a_score broj golova koji je postigao gostujući tim

team_h_score broj golova koji je postigao domaći tim

threat predstavlja meru koliko je igrač "opasan"po gol

transfers_balance razlika broja ubacivanja i izbacivanja igrača

value novčana vrednost igrača

yellow_cards broj žutih kartona

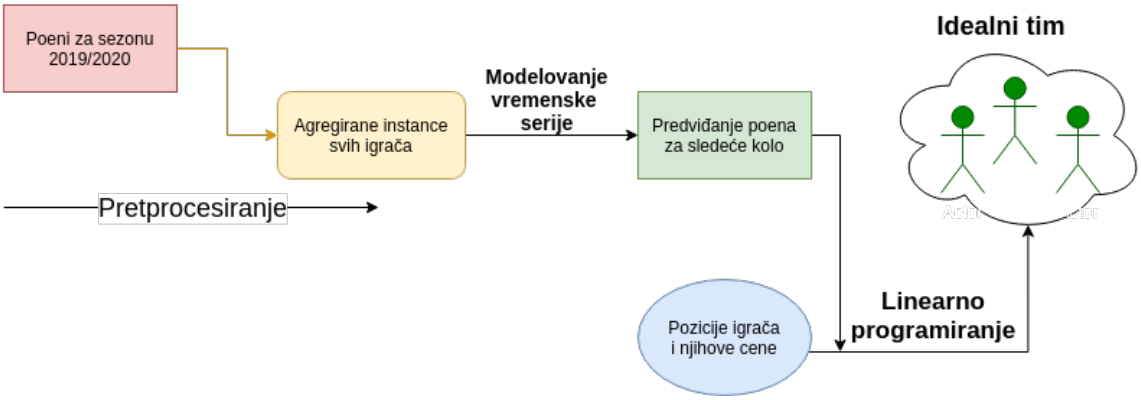
was_home da li igrač igra kod kuće

total_points ciljna promenljiva koja predstavlja broj poena ostvarenih na utakmici

3 Proces određivanja idealnog tima

Celokupan proces određivanja idealnog tima prikazan je na slici 1. Proces se sastoji iz tri faze:

1. pretprocesiranje
2. treniranje, evaluacija i testiranje modela
3. linearno programiranje



Slika 1: Proces dobijanja idelanog tima

3.1 Pretprocesiranje

Iako većina današnjih baza podataka ili bilo kakvo skladište podataka sadrži dosta podataka koji u sebi imaju nedostajuće vrednosti ili nekakvu vrstu nekonzistentnosti. Podaci koje smo mi koristili bili su zaista dosta dobro uređeni i nisu imali tih problema. Ono što je karakteristično za podatke koje smo koristili jeste to da je dosta bitnih atributa bilo razloženo u nekoliko razdvojenih datoteka. Dakle podaci nisu bili grupisani na jednom mestu nego su bili odvojeni u različitim datotekama i trebalo ih je spojiti u jedan fajl i spremiti za sledeću fazu pretprocesiranja. Svaka od datoteka koje je trebalo spojiti je imala vezu "strani <-> primarni ključ" sa nekom drugom datotekom i sledeći deo koda predstavlja spajanje dve datoteke:

```
1     df = pd.merge(df, teams,     left_on='opponent_team', right_on='id',\
2                 how='left').drop(['opponent_team','id'],axis=1)
3     df = pd.merge(df, payers_raw, left_on='element', right_on='id',\
4                 how='left').drop(['element','id'],axis=1)
```

Nakon spojenih datoteka potrebno je podatke parsirati na način da se koriste kao vremenska serija. Ovo predstavlja glavni deo pretprocesiranja gde se na osnovu **n** nedelja unazad kreira jedna instanca koja će se koristiti pri pravljenju modela. Takođe u ovom delu parsiranja se koristi pomerajući težinski prosek i eksponencijalni pomerajući težinski prosek. Instancama koje su dalje tj. u našem slučaju utakmicama koje su odigrane ranije dodeljuje se manja težina dok utakmicama koje su skorije dodeljuje im se veća težina. Smisao ovog pristupa je u tome da skorije utakmice utiču više nego utakmice koje su se odigrale odavno iz razloga što su informacije o spremnosti igrača tačnije i svežije kod skorijih utakmica. Formula koja se koristi za pomerajući težinski prosek je [3]:

$$\frac{1}{n * ((n+1)/2)} \cdots \frac{n}{n * ((n+1)/2)}$$

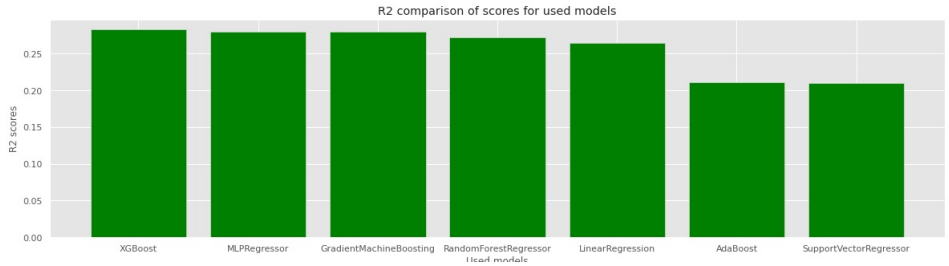
gde **n** predstavalja broj unazad gledanih nedelja. Za eksponencijalni pomerajući težinski prosek koristimo ugrađenu funkciju ***pandas.DataFrame.ewm*** [5] Python jezika iz paketa *pandas*.

3.2 Treniranje, evaluacija i testiranje modela

U ovom poglavlju ćemo govoriti o različitim konfiguracijama koje smo testirali kako bismo dobili dodatne informacije odnosno pronašli model koji najbolje uči na našim podacima. Kao i kako smo dodatno menjali parametre najbolje konfiguracije kako bi ispunila naše ciljeve koje smo postavili za ovaj projekat.
Odlučili smo da rešavamo ovaj problem regresionim modelima zato što nam je takva priroda problema.
Prvi korak je bio razdvajanje skupa podataka na trening i test skup. Različite konfiguracije smo trenirali i evaluirali na odvojenom skupu podataka za trening i nakon toga najbolju konfiguraciju testirali na test skupu.
Pomoću GridSearchCV funkcije koja pripada biblioteci „sklearn“ smo mogli da od prosledjenih parametara za određenu konfiguraciju dobijemo nazad one koji su nam dali najveću preciznost odnosno koeficijent determinacije(u nastavku „R2“) pošto se bavimo problemom regresije.
GridSearchCV funkcija vrši unakrsnu validaciju nad poslatim skupom podataka za zadat broj slojeva, osim u slučaju linearne regresije gde nismo koristili unakrsnu validaciju da bismo videli koje koeficijente je dodelila kojim atributima i testirali smo je na test skupu. U svim ostalim modelima se radila unakrsna validacija sa 5 slojeva.
Takođe svi modeli koje smo trenirali su iz spomenute biblioteke.
Napravili smo funkciju „selectParamsAndEvaluateConfigurations“ kojoj kao argumente šaljemo model koji želimo da treniramo, ime tog modela i objekat koji sadrži imena i vrednosti parametara čije kombinacije će se koristiti u pronalaženju najbolje konfiguracije tog modela.

3.2.1 Korišćeni modeli i izbor najboljeg

Nakon rezultata dobijenih unakrsnih validacija navedenih modela sa slike smo uporedili njihove R2 vrednosti.

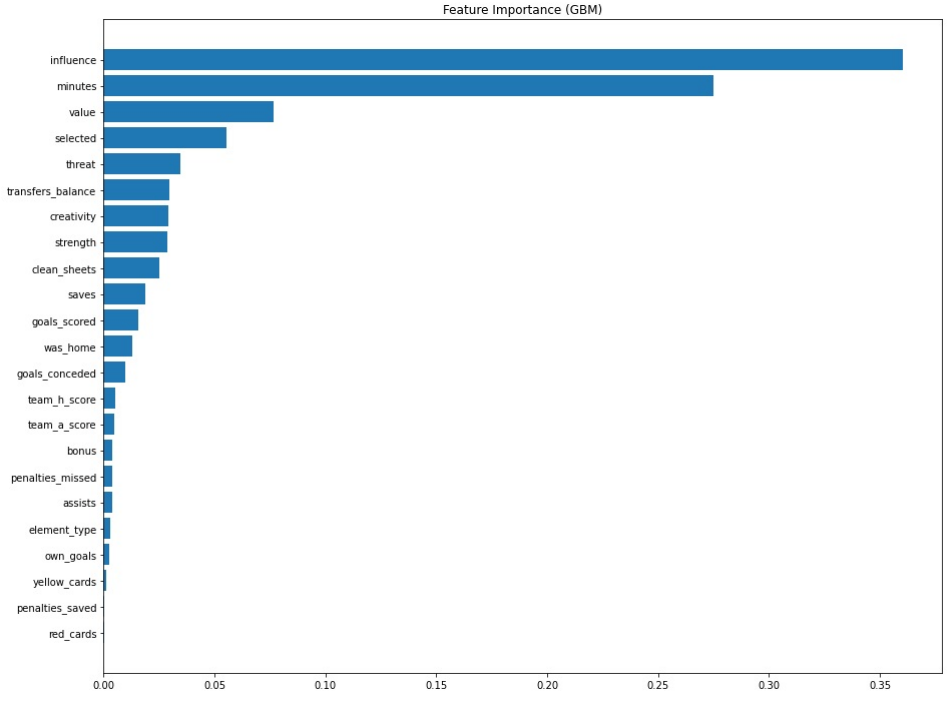


Slika 2: Poređenja vrednosti koeficijenata determinacije za modele

Pošto ima par modela koji imaju skoro identičnu preciznost, odlučili smo se za onaj koji ima mogućnost da nam prikaže koliku važnost je dao kom atributu pri treniranju učenju na trening podacima. To je bas GradientBoostingMachine konfiguracija.

3.2.2 Testiranje modela i distribucija predviđenih poena

Nakon što smo istrenirali najbolji model hteli smo da vidimo koju je važnost dodelio kom atributu što se može videti na narednoj slici:

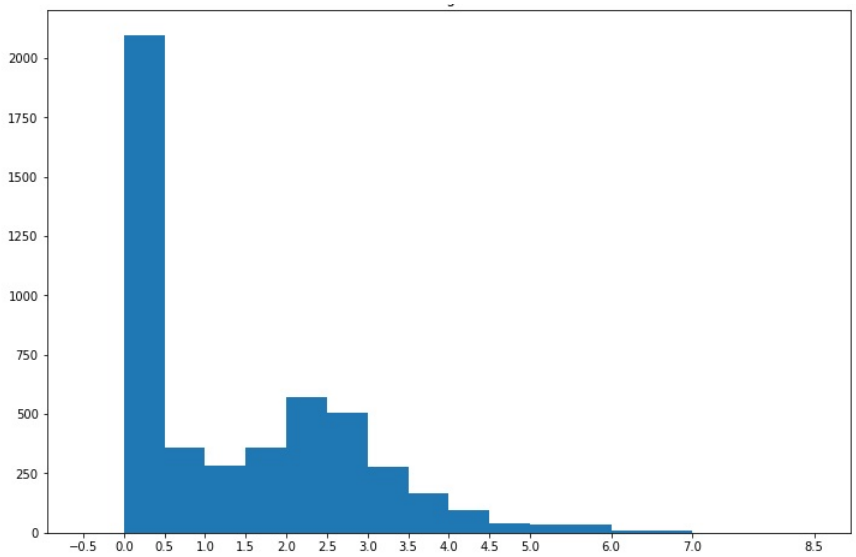


Slika 3: Važnosti atributa koje je odredio GBM model

Na osnovu ove opservacije smo menjali „max_features“ parametar modela zato što je uspevao da zadrži preciznost a u zavisnosti od vrednosti koje smo mu davali je malo drugačije birao optimalne timove.
Na primer, za manje vrednosti je gledao koji igrači imaju više minuta, manje utakmica sa primljenim golom, osrednju kreativnost i pretnju u napadu. Dok je pri povećavanju spomenutog parametra počeo vise da gleda pretnju u napadu, učešća u opasnim napadima, vrednost tog igrača kao i broj igrača u samoj igrici koji ga imaju u svom timu - što je bilo baš ono što je nama bilo potrebno za optimalni tim.
Zato što navedeni atributi u drugom slučaju zapravo maksimizuju igrače na sredini i u napadu, odnosno one koji donose najviše poena.

Zatim smo testirali model i dobili sledeće rezultate:
R2 score: 0.2866050684708288
MSE: 4.21236766907493

Na narednoj slici možete videti distribuciju poena koje taj model predvideo na test skupu:



Slika 4: Distribucija poena dobijenih posle testiranja GBM modela

Dok je:
Najmanji izračunat broj poena: 0.032267868481287106.
Najveci izračunat broj poena: 9.763231304600286.

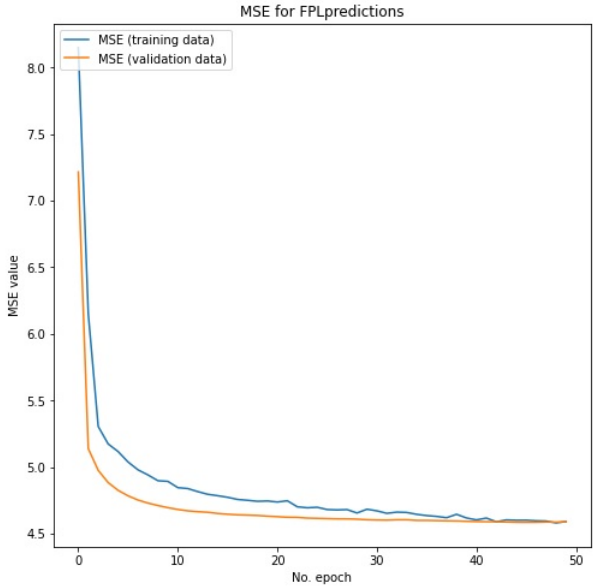
3.2.3 Rezultati koje je dala Keras neuronska mreža

Najbolje se pokazao Adam optimizator, međutim model se preprilagođavao za sve vrednosti stope učenja koje su veće od 0.000001 odnosno 10^{-6}. Na narednoj slici se može videti kako je izgledala mreža:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 128)               3072
_________________________________________________________________
dense_7 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_8 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_10 (Dense)             (None, 256)               65792
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 257
=================================================================
Total params: 233,729
Trainable params: 233,729
Non-trainable params: 0
```

Slika 5: KerasNN Model summary

Dok se na narednom grafiku može videti kako je teklo učenje mreže na trening podacima gde se za funkciju gubitka odnosno učenja koristila srednjekvadratna greška i validacioni skup koji je iznosio 20% trening skupa:



Slika 6: Grafik koji prati MSE na trening i validacionom skupu prilikom učenja mreže

3.3 Linearno programiranje

Mnogi problemi mogu biti formulisani kao maksimizacija ili minimizacija neke ciljne funkcije za date ograničene resurse i uzajamna ograničenja. Ako možemo definisati ciljnu funkciju kao linearnu funkciju određenih promenljivih i ako možemo zadati ograničenja u resursima kao jednakosti ili nejednakosti ovih promenljivih, onda dobijamo problem linearnog programiranja. Linearni programi se javljaju u raznim praktičnim primenama [4].

Nakon dobijenog modela mašinskog učenja o čemu je pričano u delu 3.2, potrebno je predvideti poene za svakog od igrača i primeniti linearno programiranje koje će maksimizirati ukupne poene za idealni tim od 15 igrača. Implementacija linearnog programiranja urađena je pomoću ***Pulp*** biblioteke [6].

Definicija linearnog problema kojim se bira idealni tim od 15 igrača:

- z_i, predviđeni poeni i-tog igrača za sledeću utakmicu
- c_i, cena i-tog igrača
- y_i, indikator da li je i-ti igrač u idealnom timu ili (vrednost 0 znači da ne pripada, vrednost 1 znači da pripada idealnom timu)
- n, ukupan broj igrača
- F, skup igrača koji su napadači (FWD)
- M, skup igrača koji su u veznom redu (MID)
- D, skup igrača koji su odbrana (DEF)
- G, skup igrača koji su golmani (GLK)

$$Max. \sum_{i=1}^{n} z_i * y_i$$

ograničenja:

$$\sum_{i=1}^{n} c_i * y_i <= 100.0$$

$$\sum_{i \in F} y_i = 3 \qquad \sum_{i \in M} y_i = 5$$

$$\sum_{i \in D} y_i = 5 \qquad \sum_{i \in G} y_i = 2$$

$$y_i \in \{0, 1\}; \qquad 1 \leq i \leq n$$

Rezultat linearnog programiranja je dat u formi binarnih vrednosti promenljivih y_i koji su dalje mapirani u određene igrače. S obzirom da broj igrača nije veći od 700 rešavaču nije bilo potrebno mnogo vremeda da odabere idealni tim od 15 igrača.

Linearno programiranje je dosta pomoglo pri odabiru idealnog tima od 15 igrača nakon kreiranja modela mašinskog učenja.

4 Prikaz rezultata

Što se tiče krajnjeg rezultata, izvršili smo linearno programiranje da predvidimo najbolju formaciju i postavu igrača za poslednju nedelju na osnovu 5 nedelja koje su joj prethodile, kao što je rečeno na početku rada. Izvršeno je predviđanje za prvu postavu koja se sastoji od 11 igrača, koju smo birali za sumu od 85 miliona zato što smo izračunali da bi najjeftinija „klupa“ iznosila 15 miliona a mi želimo da maksimizujemo poene koje bi tih 11 igrača ostvarilo.
Rezultati koje smo dobili su bili bolji nego što smo očekivali jer se više od polovina igrača poklapala sa najboljim igračima te nedelje ili igračima koji nisu „prazni“ što je izraz za one igrače koji su te nedelje igrali utakmice a nisu uradili ništa odnosno nisu osvojili nikakve poene.
Ovde je linearno programiranje napravljeno tako da maksimizuje postave igrača po svim formacijama koje je moguće postaviti i da izdvoji onu koja bi donela najviše poena.
Najbolja formacija i postava za poslednju nedelju:

Best team
3-5-2:

Najboljih 11 igrača

Ime i prezime igrača	cena	ostvareni poeni	pozicija igrača
Che_Adams	5.3	10.210934564320262	'FWD'
Danny_Ings	7.6	9.918711258778774	'FWD'
Christian_Pulisic	7.4	10.78262020967625	'MID'
Kevin_De_Bruyne	10.6	10.597844584663173	'MID'
Bernardo_Silva	7.6	10.752334195689984	'MID'
Raheem_Sterling	12.0	10.752334195689984	'MID'
Rodrigo_Hernandez	5.3	10.597844584663173	'MID'
Reece_James	4.9	10.123557205522951	'DEF'
João_Cancelo	5.1	10.597844584663173	'DEF'
Ryan_Bertrand	4.8	10.078702237883336	'DEF'
Jordan_Pickford	5.2	8.341621273551166	'GLK'

Takođe smo uradili linearno programiranje za najboljih 15 igrača pošto u igrici postoji čip koji se zove „Bench-boost“ koji može da se iskoristi jednom u sezoni kada se računaju i poeni koje su ostvarili igrači na „klupi“ odnosno oni koji su nama bili u izmeni a zapravo su igrali i uspeli da osvoje neke poene.
U nastavku se na tabeli mogu videti najboljih 15 igrača:

Najboljih 15 igrača

Ime i prezime igrača	cena	ostvareni poeni	pozicija igrača
Che_Adams	5.3	10.210934564320262	'FWD'
Chris_Wood	6.2	9.918711258778774	'FWD'
Danny_Ings	7.6	9.918711258778774	'FWD'
Christian_Pulisic	7.4	10.78262020967625	'MID'
Kevin_De_Bruyne	10.6	10.597844584663173	'MID'
Bernardo_Silva	7.6	10.752334195689984	'MID'
Raheem_Sterling	12.0	10.752334195689984	'MID'
Rodrigo_Hernandez	5.3	10.597844584663173	'MID'
Reece_James	4.9	10.123557205522951	'DEF'
Marcos_Alonso	6.1	9.94600980852834	'DEF'
João_Cancelo	5.1	10.597844584663173	'DEF'
Phil_Bardsley	4.4	9.918711258778774	'DEF'
Ryan_Bertrand	4.8	10.078702237883336	'DEF'
Jordan_Pickford	5.2	8.341621273551166	'GLK'
Kasper_Schmeichel	5.5	7.574938197333164	'GLK'

5 Zaključak

Fantazi premijer liga pružila je mogućnost običnom čoveku da oseti kako je to biti menadžer nekog tima i da sam kreira i određuje svoju ekipu. Projekat pruža pomoć novim igračima u odabiru njegovog prvog savršenog tima, takođe može pomoći i iskusnijim igračima da izvrše korekcije nad svojim timom. Korišćene su različite tehnike mašinskog učenja koje su predviđale poene igrača u želji da se dostigne što veća i donekle dovoljna preciznost. Na kraju je korišćeno linearno programiranje koje je uz poštovanje svih ograničenja Fantazi premijer lige biralo savršen tim na osnovu rezultata predviđanja poena. Dalji rad bi se odnosio na to da se probaju još neke od tehnika mašinskog učenja, da se uradi drugačije pretprocesiranje i da se drugačije pripreme podaci u vremensku seriju, dodavanje nekih novih atributa koji bi pomogli da se bolje predvide poene. Idealan sistem ne postoji i ne treba očekivati da će projekat uspeti da nadmaši sve rezultate i pobedi sve ljude na planeti jer postoje različiti faktori koji ne mogu da se predvide jedan od njih je i sreća ali projekat može dosta dobro da vas postavi na pravi put u odabiru svog savršenog tima.

Literatura

[1] Dostupno na:. https://sr.wikipedia.org/sr-el/Fudbal.

[2] Dostupno na:. https://sr.wikipedia.org/sr-el/Premijer_liga.

[3] What is the weighted moving average (wma)? https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/weighted-moving-average-wma/.

[4] dr Boban Stojanović. Linearno programiranje. https://imi.pmf.kg.ac.rs/index-old.php?option=com_docman&task=doc_view&gid=554.

[5] pandas. Dostupno na:. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.ewm.html.

[6] pypi. Dostupno na:. https://pypi.org/project/PuLP/.

[7] vaastav. Fpl podaci. https://github.com/vaastav/Fantasy-Premier-League.