Sustav za predviđanje rezultata i optimizaciju timova Fantasy Premier lige

Šošić, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 396

SUSTAV ZA PREDVIĐANJE REZULTATA I OPTIMIZACIJU TIMOVA FANTASY PREMIER LIGE

Marko Šošić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 396

SUSTAV ZA PREDVIĐANJE REZULTATA I OPTIMIZACIJU TIMOVA FANTASY PREMIER LIGE

Marko Šošić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 396

Pristupnik:	**Marko Šošić (0036526905)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Ivica Botički

Zadatak: **Sustav za predviđanje rezultata i optimizaciju timova Fantasy Premier lige**

Opis zadatka:

Fantasy Premier liga natjecateljska je igra čiji je cilj uz određeni budžet odabrati nogometaše koji će donijeti bodove u ovisnosti o stvarnom učinku na utakmici. U sklopu diplomskog rada potrebno je izraditi sustav za predviđanje rezultata i optimizaciju timova Fantasy Premier lige koji kombinira napredne tehnike analize podataka s modelima umjetne inteligencije kako bi predvidio bodove igrača Fantasy Premier lige temeljem njihovih prethodnih performansi. Aplikacija treba pružati kontinuirane statističke analize tijekom sezone, omogućujući korisnicima uvid u dosadašnje rezultate i trendove te im olakšati praćenje izvedbe pojedinih igrača. Napredna korisnička sučelja i analitički alati omogućit će intuitivno korištenje i donošenje informiranih odluka prilikom upravljanja nogometnim timom. U izradi rada koristit će se programski jezik Python i Spring Boot.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

1. Uvod

Fantasy premier league najpopularnija je fantasy igra diljem svijeta koja okuplja preko 10 milijuna igrača. Temelji se na stvarnim događajima na osnovu kojih igrači skupljaju bodove te se tako natječu. Cilj je što bolje optimizirati igrače u ograničenom budžetu za ostvarenje što boljih rezultata. Natjecanja mogu biti privatna odnosno privatne lige dok svake sezone postoje globalna liga i globalni kup.

S obzirom na to da ishodi utakmica nepredvidivi pa tako i učinci samih igrača (igrača iz stvarnog svijeta, u nastavku teksta spominjan kao nogometaš) predviđanje performansi igrača na pojedinoj utakmici predstavlja veliki izazov za implementirati. Predviđanje nikada nije moguće sa sto postotnom sigurnošću zbog razno raznih vanjskih utjecaja, a isto tako jer igrači nisu roboti za koje se točno kako će se ponašati no to i nije cilj ovog modela. Ovaj sustav samo bi trebao biti pomoćnik pri slaganju momčadi kako sama srž igre ne bi bila izgubljena te kako bi svi, moglo bi ih se nazvati menadžeri svojih ekipa, mogli pokazati svoje umijeće i taktike slaganja momčadi.

U ovom radu cilj je razviti sustav, odnosno model umjetne inteligencije, koji bi pomogao u predviđanju klase bodova koju će igrač ostvariti te sve to prikazati u web aplikaciji koja bi uz samu mogućnost predviđanja bodova imala i još poneke opcije. Za razvoj modela korišten je programski jezik Python dok su za izradu web aplikacije korišteni Spring Boot i React. Iskorišteni su podatci, koji su dodatno analizirani i pretprocesirani, iz prijašnjih sezona koji su iskorišteni za treniranje modela. U diplomskom radu koriste se razni prediktivni modeli; Logistička regresija, KNN (K Nearest Neighbours odnosno K najbližih susjeda), RFC (Random Forrest Classifier odnosno Klasifikator slučajnih šuma) te Neuronske mreže, od kojih je najbolje rezultate polučio model slučajnih šuma od 100 estimatora (stabala odluke) s 0.82 točnosti.

2. Fantasy premier liga

Fantasy je pojam koji predstavlja vrstu igre u kojoj igrači na temelju stvarnih sportskih događaja ostvaruju bodove i natječu se. Tako za mnoge sportove imamo fantasy izvedbe; nogometne za velik broj liga (UEFA Champions League Fantasy - fantasy europske lige prvaka, Fantasy premier league - fantasy engleske premier lige, Fantasy HNL - fantasy hrvatske lige, LaLiga fantasy - fantasy španjolske LaLige...), košarkaške (NBA fantasy - fantasy američkog NBA, EuroLeague fantasy - fantasy Eurolige...), formule (F1 fantasy) i mnoge druge.

Fantasy premier league se pri puta pojavila u sezoni 2002/2003. Prema navodima sa službene stranice, igricu igra preko 11 milijuna igrača što ju čini najigranijom fantasy nogometnom igrom na svijetu.

2.1. Način igranja

Svaki igrač na početku sezone, ili nekada kasnije kada se priključi, ima budžet od 100 milijuna funti. Od navedenog novca mora odabrati 15 igrača za svoj tim: 2 golmana, 5 obrambenih igrača, 5 veznih igrača te 3 napadača. Za svako kolo igrač odabire 11 igrača od navedenih 15 prije nego krene prva utakmica kola određen je tzv. deadline odnosno trenutak u kojem se zaključava postava. Dodatni uvjeti su da u 11 odabranih bude minimalno: 1 (u ovom slučaju i maksimalno) golman, 3 obrambena igrača, 3 vezna igrača i minimalno 1 napadač. Ako neki od igrača ne nastupi u kolu u kojem je izabran u početnih 11 zamjenjuje ga prvi s klupe tako da su gore navedeni uvjeti zadovoljeni, ako nisu zamjenjuje ga drugi itd. što isto vrijedi ako i više od jednog igrača iz prve postave ne nastupi.

Nakon svakog kola igrač dobije "besplatni" transfer koji mu omogućava da zamijeni

igrača iz svoje momčadi za nekog drugog. Ako se navedeni transfer ne iskoristi on se može iskoristiti u sljedećem kolu u kojem shodno tome postoje dva besplatna transfera. Maksimalni broj transfera je 2 što znači da ako igrač nekoliko kola za redom ne koristi transfere i dalje će imati samo dva besplatna transfera. Svaki transfer koji nije besplatan košta igrača -4 boda u njegovom ukupnom broju bodova skupljenim tijekom sezone.

2.1.1. Način bodovanja

Nakon što igrač izabere 11 nogometaša on skuplja bodove na temelju učinka tih nogometaša na stvarnim utakmicama. Nogometaš može bodove dobiti za postignuti gol, za asistenciju, za "clean sheet" (termin koji opisuje da nogometašev tim nije primio pogodak te ga mogu ostvariti samo određene pozicije), za nastup. Isto tako može dobiti negativne bodove: crveni karton, žuti karton, autogol, za svaka dva primljena gola (samo određeni igrači). Uz sve navedeno svako kolo odabire se kapetan kojemu se bodovi duplaju.

Detaljan opis bodovanja igrača naveden na službenoj stranici

Akcija	**Bodovi**
Za igranje do 60 minuta	1
Za igranje 60 minuta ili više (isključujući nadoknadu)	2
Za svaki gol koji postigne vratar ili obrambeni igrač	6
Za svaki gol koji postigne vezni igrač	5
Za svaki gol koji postigne napadač	4
Za svaku asistenciju za gol	3
Za čistu mrežu koju sačuva vratar ili obrambeni igrač	4
Za čistu mrežu koju sačuva vezni igrač	1
Za svake 3 obrane vratara	1
Za svaku obranu penala	5
Za svaki promašaj penala	-2
Bonus bodovi za najbolje nogometaše u utakmici	1-3
Za svaka 2 primljena gola vratara ili obrambenog igrača	-1
Za svaki žuti karton	-1
Za svaki crveni karton	-3
Za svaki autogol	-2

Tablica 2.1. Pravila bodovanja u Fantasy Premier ligi (Scout, 2024)

Čista mreža

Ako je nogometaš zamijenjen prije nego je primljen gol to neće utjecati na njegove clean sheet bodove ostvarene unutar odigranih 60 minuta.

Asistencije

Asistencija se dodjeljuje igraču koji napravi završni pas prije postizanja gola bilo to namjerno ili nenamjerno

Odbijene lopte

Ako je gol postignut iz odbijanca nogometaš koji je uputio odbijanac dobiva asistenciju. Gdje slično vrijedi i za autogolove.

Penali i slobodni udarci

U slučaju pogotka iz kaznenog udarca ili slobodnog udarca asistencija se pripisuje igraču ako je ne na njemu napravljen faul te on nije strijelac pogotka

Bonus bodovi

Na temelju bonus points systema određuje se tok dobiva 3,2 i 1 bod u određenoj utakmici Ako postoje dva (ili više) nogometaša s jednakim brojem bodova onda oba (ili svi) nogometaši dobiju broj bodova za tu poziciju dok se onda sljedeća pozicija preskače. npr. Ako 2 nogometaša imaju 31 boda a jedan ima 30 tada će prva dva spomenuta dobiti 3 boda, a treći 1 bod, isto tako ako imamo podjelu da za treće mjesto u BPS-u imamo više igrača svi oni će dobiti po 1 bod.

2.1.2. Čipovi

Takozvani čipovi su određena pojačanja koja se mogu iskoristiti jednom u sezoni (ili jednom u polusezoni gledajući wildcard)

Pojačanje klupe

Čip koji omogućava da se za određeni kolo zbrajaju bodovi svih 15 igrača, a ne samo prvih 11.

Slobodan pokušaj

Čip koji omogućava da se za određeno kolo složi cijela nova ekipa koja će vrijediti samo to navedeno kolo.

Trostruki kapetan

Čip koji omogućava da se kapetanovi bodovi množe s 3 umjesto s 2.

Divlja karta

Jedini čip koji je dostupan jednom u svakoj polusezoni te omogućava da se zamijeni neograničen broj nogometaša odnosno obavi neograničen broj transfera bez oduzimanja 4 boda po transferu.

3. Treniranje modela

Podatci su preuzeti s poznate stranice kaggle.com, jedne od najpoznatijih stranica za datasetove. Prilikom treniranja modela korišten je programski jezik Python te jupyter notebook kao okruženje razvoja. Podatci kreću od sezone 2016/17 do 2022/23 uz izuzetak sezona 2018/19 i 2019/20.

3.1. Tehnologije i okruženja

Pokretanje jupyter notebooka odrađeno je koristeći PyCharm, Jetbrainsov IDE za programski jezik Python.

3.1.1. Python

Python je interpretirani, interaktivni, objektno-orijentirani programski jezik. Podržava više programskih paradigmi osim objektno-orijentiranog programiranja, kao što su proceduralno i funkcionalno programiranje. (Python Software Foundation, 2024) U kasnim 80-ima Guido Van Rossum je započeo rad na Python u Nacionalnom istraživačkom institutu za matematiku i računalne znanosti u Nizozemskoj ili Centrum voor Wiskunde e Informatica (CWI) kako je poznat na nizozemskom. (Venners, 2003) Prva verzija je objavljena 1991 godine. (van Rossum, 2009)

Iako je opći programski jezik, Pythone se često naziva skriptnim jezikom jer olakšava korištenje i upravljanje drugim komponentama, ali možda najveća prednost je njegova jednostavnost te čini razvoj softvera bržim.(Lutz, 2011)

3.1.2. Jupyter notebook

Jupyter notebook je aplikacija za stvaranje notebook-a. Nudi brze i interaktivne načine za prototipiranje i objašnjavanje koda kao i izvršavanje i vizualizaciju određenih podataka.

3.2. Podatci

Podatci preuzeti s kaggle-a imali su 96169 zapisa te 37 značajki. Značajke u izvornom skupu podataka su: `season_x`, `name`, `position`, `team_x`, `assists`, `bonus`, `bps`, `clean_sheets`, `creativity`, `element`, `fixture`, `goals_conceded`, `goals_scored`, `ict_index`, `influence`, `kickoff_time`, `minutes`, `opponent_team`, `opp_team_name`, `own_goals`, `penalties_missed`, `penalties_saved`, `red_cards`, `round`, `saves`, `selected`, `team_a_score`, `team_h_score`, `threat`, `total_points`, `transfers_balance`, `transfers_in`, `transfers_out`, `value`, `was_home`, `yellow_cards`, `GW`.

Popis svih značajki i brojnost te tip:

#	Column	Non-Null Count	Dtype
0	season_x	96169	object
1	name	96169	object
2	position	96169	object
3	team_x	76317	object
4	assists	96169	int64
5	bonus	96169	int64
6	bps	96169	int64
7	clean_sheets	96169	int64
8	creativity	96169	float64
9	element	96169	int64
10	fixture	96169	int64
11	goals_conceded	96169	int64
12	goals_scored	96169	int64
13	ict_index	96169	float64
14	influence	96169	float64
15	kickoff_time	96169	object
16	minutes	96169	int64
17	opponent_team	96169	int64
18	opp_team_name	96169	object
19	own_goals	96169	int64
20	penalties_missed	96169	int64
21	penalties_saved	96169	int64
22	red_cards	96169	int64
23	round	96169	int64
24	saves	96169	int64
25	selected	96169	int64
26	team_a_score	96169	float64
27	team_h_score	96169	float64
28	threat	96169	float64
29	total_points	96169	int64
30	transfers_balance	96169	int64
31	transfers_in	96169	int64
32	transfers_out	96169	int64
33	value	96169	int64
34	was_home	96169	bool
35	yellow_cards	96169	int64
36	GW	96169	int64

Tablica 3.1. Tablica sa svim značajkama i njihovim tipovima

Popis jedinstvenih vrijednosti značajki:

Column	**Unique Count**
season_x	5
name	1327
position	5
team_x	25
assists	5
bonus	4
bps	114
clean_sheets	2
creativity	830
element	778
fixture	380
goals_conceded	10
goals_scored	5
ict_index	259
influence	518
kickoff_time	1220
minutes	91
opponent_team	20
opp_team_name	31
own_goals	3
penalties_missed	2
penalties_saved	2
red_cards	2
round	38
saves	14
selected	60872
team_a_score	8
team_h_score	9
threat	145
total_points	31
transfers_balance	29780
transfers_in	22813
transfers_out	25229
value	97
was_home	2
yellow_cards	2
GW	38

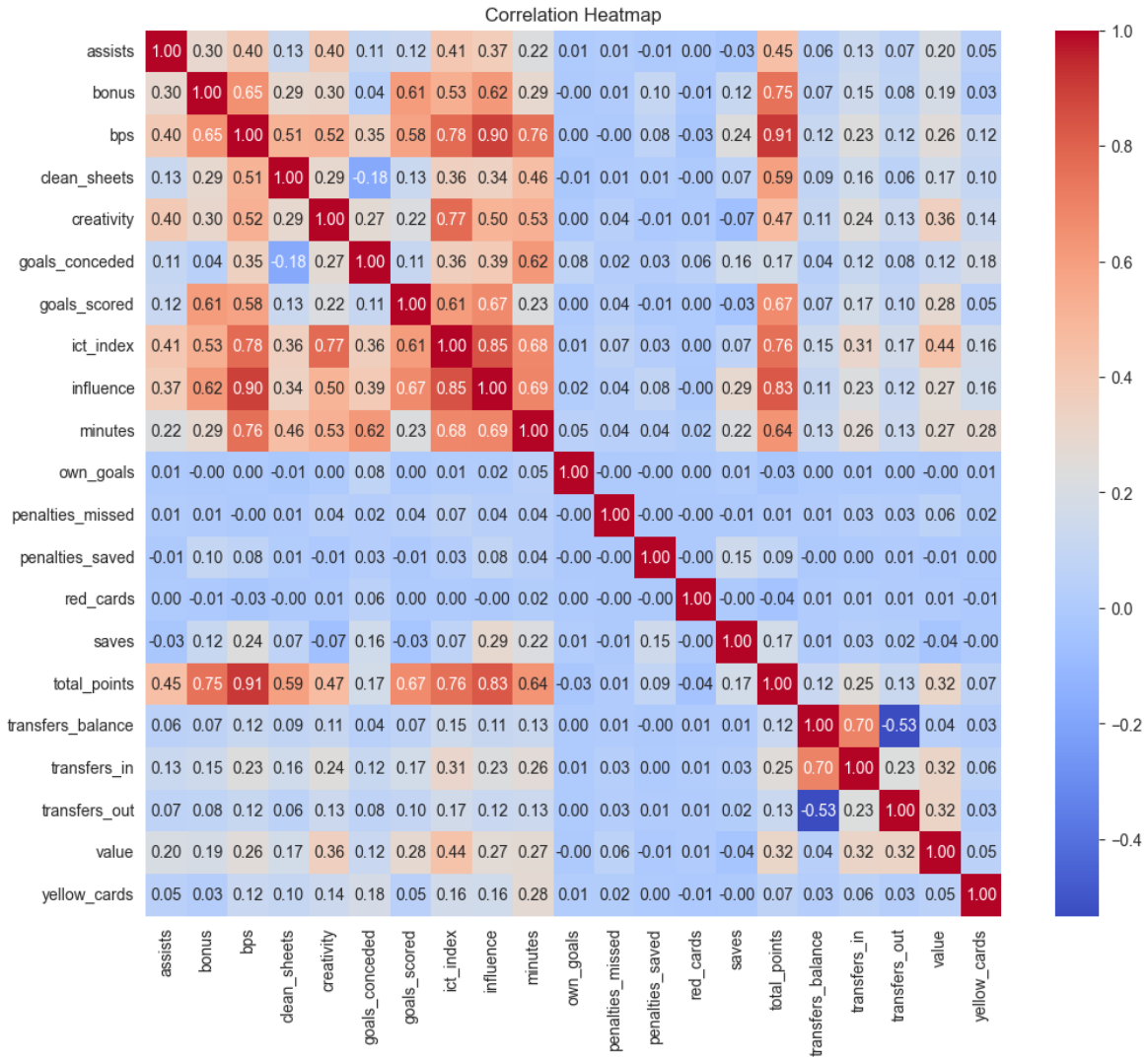
Tablica 3.2. Tablica s brojem jedinstvenih vrijednosti za svaki stupac

3.2.1. Pretprocesiranje podataka

Kako bi se provjerila koreliranost kreirana je matrica korelacije brojčanih značajki.

```
cols = ['assists', 'bonus', 'bps', 'clean_sheets', 'creativity',
                    'goals_conceded', 'goals_scored', 'ict_index', 'influence',
                    'minutes', 'own_goals', 'penalties_missed', 'penalties_saved',
                    'red_cards', 'saves', 'total_points', 'transfers_balance',
                    'transfers_in', 'transfers_out', 'value', 'yellow_cards']
sub_df = data[cols]
corr_matrix = sub_df.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

Odsječak koda 3..1: Korelacijski matrica



Slika 3.1. Korelacijska matrica

Vidi se da je bps korelirana s dvije varijable te se izbacuje jer je posljedična odnosno vrijednost joj je poznata tek nakon utakmice. Ostavljane su samo sljedeće značajke koje su relevantne za model: `fixture`, `value`, `transfers_in`, `transfers_out`, `selected`, `position`, `creativity`, `influence`, `threat`, `ict_index`, `was_home`, `total_points`, `minutes`. Ostavljani su samo oni podatci kojima je vrijednost značajke minutes veća od 0:

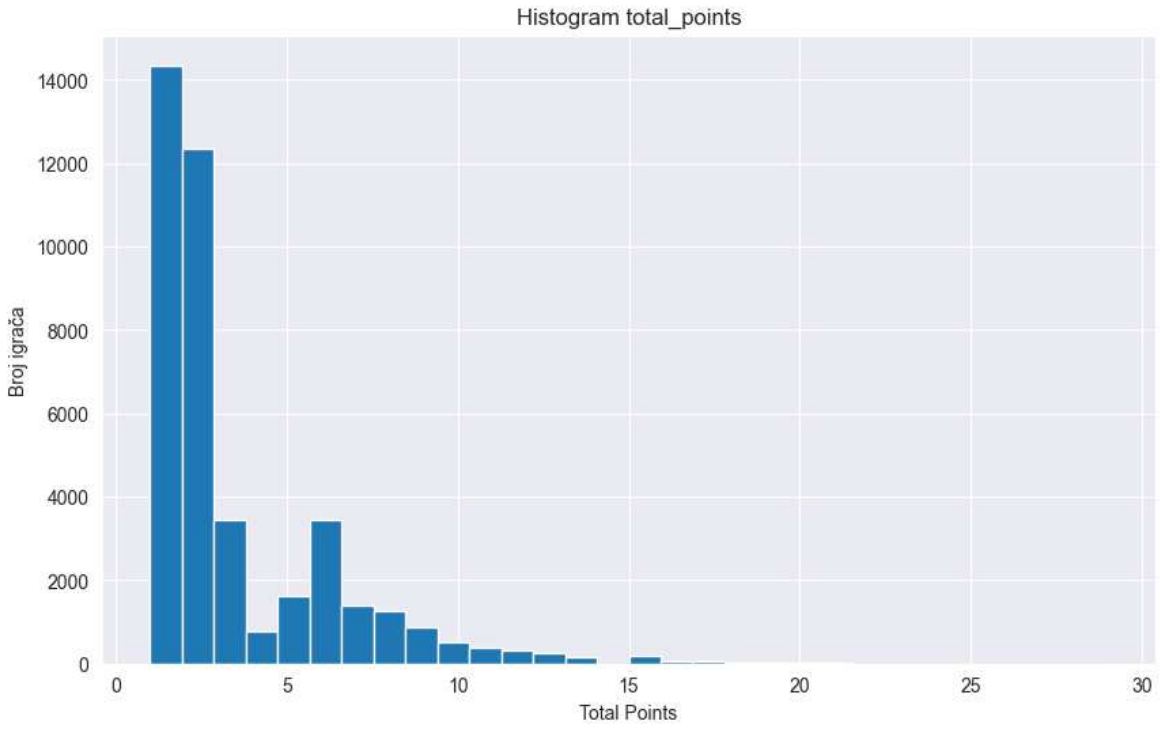
```
data = data[data.minutes > 0]
```

Odsječak koda 3..2: Uklanjanje nerelevantih redaka

Nakon inicijalne obrade odrađena je kratku vizualizacija podataka, Prikaz raspodjela `total_points` značajke te se u obzir uzimaju samo one koji su ostvarili više od 0 bodova:

```
plt.figure(figsize=(10, 6))
plt.hist(data.total_points[data.total_points > 0], bins=30)
plt.xlabel('Total Points')
plt.ylabel('Broj igraca')
plt.title('Histogram total_points')
plt.show()
```

Odsječak koda 3..3: Histogram broja bodova

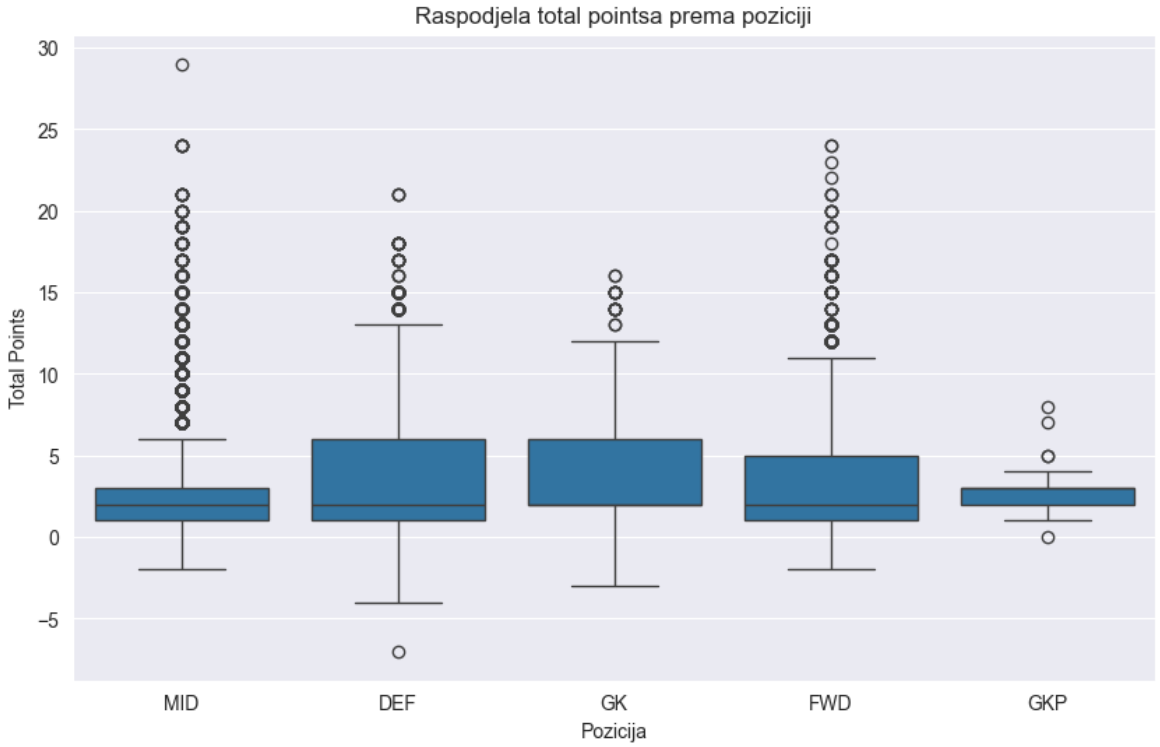


Slika 3.2. Histogram bodova

Isto tako razmotren je raspored varijable total_points varijable prema poziciji. To je napravljeno koristeći boxplot koji jasno prikazuje raspodjelu.

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='position', y='total_points', data=data[data.minutes > 0])
plt.xlabel('Pozicija')
plt.ylabel('Total Points')
plt.title('Raspodjela total pointsa prema poziciji')
plt.show()
```

Odsječak koda 3..4: Boxplot total_points varijable prema poziciji

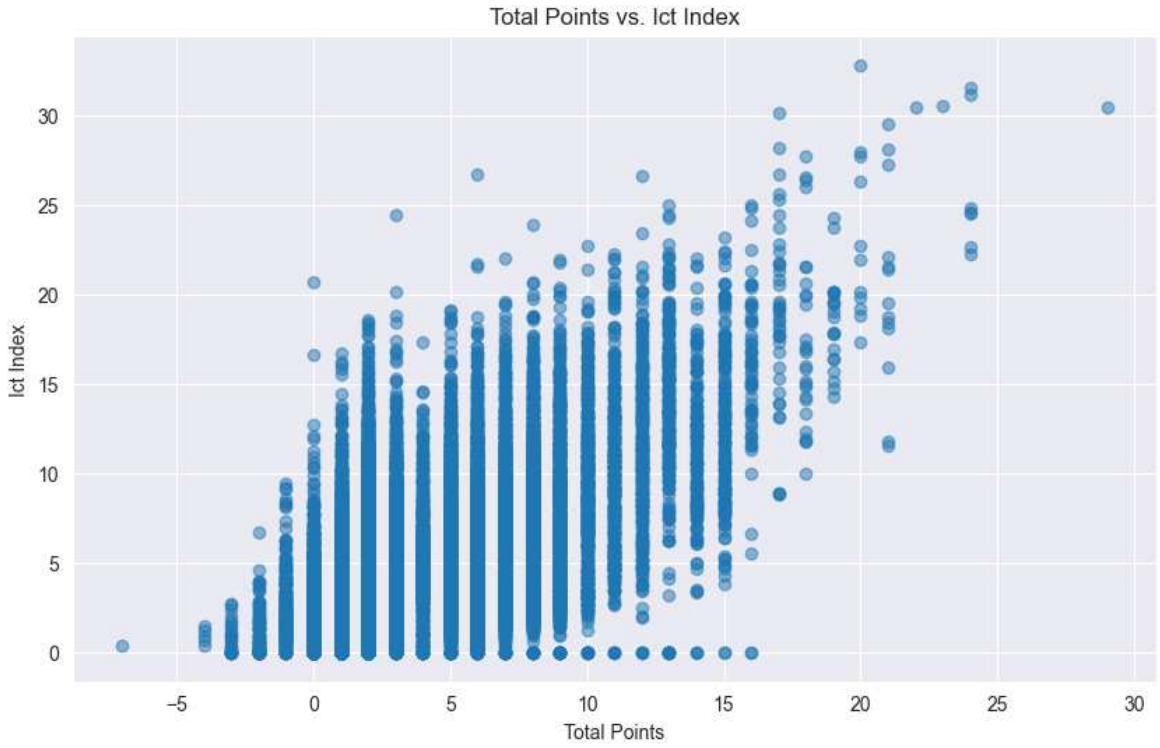


Slika 3.3. Boxplot total_pointsa prema poziciji

Bilo je za pretpostaviti kako bi značajka ict_index mogla biti korelirana sa značajkom total_points pa je odlučeno to i provjeriti vizualno scatter plot-om te potvrditi osnovanu pretpostavku:

```
plt.figure(figsize=(10, 6))
plt.scatter(data.total_points, data.ict_index, alpha=0.5)
plt.xlabel('Total Points')
plt.ylabel('Ict Index')
plt.title('Total Points vs. Ict Index')
plt.show()
```

Odsječak koda 3..5: Scatter plot total_pointsa i ict_indexa



Slika 3.4. Scatter plot total_pointsa i ict_indexa

Mapirane su vrijednosti position značajke sa stringa na brojčanu vrijednost kako bi ju bilo lakše koristili.

```
data['position'] = data['position'].map({'GKP': 0, 'DEF': 1, 'MID': 2, 'FWD': 3})
```

Odsječak koda 3..6: Mapiranje pozicije iz string-a u brojčanu varijablu

Nakon odrađenog mapiranja, sve null vrijednosti zamijenjene su medijanom značajke:

```
data = data.fillna(data.median())
```

Odsječak koda 3..7: Zamjena null vrijednosti medijanom

S obzirom na to da je u diplomskom radu odabrana klasifikaciju u 4 određene grupe, mapirana je svaka od značajki `total_points` u odgovarajuću grupu te je maknuta ta značajka.

```
def group_points(points):
    if points <= 1:
        return 0
    elif points <= 4:
        return 1
```

```
    elif points <= 7:
        return 2
    else:
        return 3

data['points_group'] = data['total_points'].apply(group_points)

data = data.drop('total_points', axis=1)
```

Odsječak koda 3..8: Pretvorba u kategoričku varijablu

S obzirom na to da su podatci bili nejednako raspoređeni korištene su određene metode stvaranja sintetičkih podataka. Iskorišten je postojeći ADASYN algoritam za generiranje novih sintetičkih podataka koji na temelju susjednih podataka generira nove sintetičke podatke:

```
from imblearn.over_sampling import ADASYN

X = data.drop('points_group', axis=1)
y = data['points_group']

adasyn = ADASYN(sampling_strategy='minority', random_state=42)
X_res, y_res = adasyn.fit_resample(X, y)

data_res = pd.concat([X_res, y_res], axis=1)
```

Odsječak koda 3..9: Sintaktički generirani podatci

Kako su podatci grupe 2 bili znatno (duplo) manje zastupljeni odlučeno je dupplicirati svaki red te tako stvoriti sintetičke podatke.

3.3. Treniranje prediktivnih modela

U diplomskom radu koristi se više modela umjetne inteligencije te odabrati onaj s najvećom preciznosti:

1. KNN - K nearest neighbors
2. Logistic regression
3. Random Forest Classifier
4. Neural network

3.3.1. KNN - K nearest neighbors

KNN - K nearest neighbors temelji se na ideji da najbliži obrasci ciljnom obrascu x, za koji tražimo oznaku, pružaju korisne informacije o oznaci. KNN dodjeljuje klasnu oznaku većine od K najbližih uzoraka u prostoru podataka. Za tu svrhu, moramo biti u mogućnosti definirati mjeru sličnosti u prostoru podataka. (Kramer, 2013) Dakle cilj al algoritma je pridružiti svaku točku (ili primjerak) jednoj od k grupa tako da udaljenost (najčešće Euklidova) točaka do centroida najmanja moguća. To možemo razložiti u nekoliko ponavljajućih koraka:

1. Izaberemo broj grupa k tako da je prosjek silueta najveći.

$$S = \frac{1}{N} \sum_{i=1}^{N} \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

   Gdje su:

   - N - ukupan broj primjeraka u skupu podataka,
   - a_i - prosječna udaljenost primjerka i do svih drugih primjeraka u istoj grupi (klasteru),
   - b_i - najmanja prosječna udaljenost primjerka i do svih primjeraka u bilo kojoj drugoj grupi (klasteru) kojem i ne pripada.

2. Slučajno odaberi k centara klaster
3. Svaki od n objekata pridruži najbližem centroidu
4. Promijeni centre klastera tako da se nađe srednja vrijenost svih klastera

```
from sklearn.neighbors import KNeighborsClassifier

model_knn = KNeighborsClassifier(n_neighbors=5)
model_knn.fit(X_train, y_train)
y_pred = model_knn.predict(X_test)
accuracy = model_knn.score(X_test, y_test)
```

Odsječak koda 3..10: KNN

Točnost ovog modela je 0.6687145934732307.

3.3.2. Logistička regresija

Iako najčešće namijenjena binarnoj klasifikaciji, isprobana je i logističku regresiju. Logistička regresije je nadzirani algoritam strojnog učenja koji obavlja zadatke binarne klasifikacije predviđanjem vjerojatnosti ishoda, događaja ili opažanja. Logistička regresija analizira odnos između jedne ili više nezavisnih varijabli i klasificira podatke diskretne klase. Koristi se u prediktivnom modeliranju, gdje model procjenjuje matematičku vjerojatnost da li primjerak pripada određenoj kategoriji ili ne.(Kanade, 2022)

```
from sklearn.linear_model import LogisticRegression

model_lr = LogisticRegression(random_state=42)
model_lr.fit(X_train, y_train)

y_pred = model_lr.predict(X_test)
accuracy = model_lr.score(X_test, y_test)
```

Odsječak koda 3..11: Logistic regression

Točnost ovog modela je 0.6148567781058006.

3.3.3. Model slučajnih šuma

Model slučajnih šuma, odnosno Random Forrest Classifier, je model koji se temelji na algoritmu stabala odluke. Slučajne šume je model koji gradi veliku kolekciju nekoreliranih stabala, a zatim od njih uzima prosjek. U većini problema performanse slučajnih šuma su vrlo slične boostingu, a lakše ih je trenirati i podešavati. Kao posljedica toga, slučajne šume su popularne i implementirane u raznim paketima.(Hastie et al., 2009)

Algoritam slučajnih šuma:

- Odabir broja stabala (N) - odabire se broj stabala u slučajnoj šumi koji se koristi, veći broj stabala može povećati točnosti modela, ali može i povećati trošak pa je glavni cilj odabrati najoptimalniji N za naš slučaj.
- Treniranja svakog stabla
  - Odabir uzorka - iz skupa za treniranje uzima se podskup na kojem će se trenirati navedeni skup.

  - Izgradnja stabla - za svaki čvor se rekurzivno ponavljaju sljedeći koraci dok se ne dođe do određene dubine stabla ili dok se ne dođe do minimalne veličine čvora:
    * slučajnim odabirom odabere se podskup značajki iz skupa značajki.
    * podjela čvora - od podskupa značajki iz prvog koraka odabire se najbolji podskup značajki odnosno onaj koji daje najveću informacijsku vrijednost.
- Predikcija - za klasifikaciju svako stablo daje svoju odluku te je konačna predikcija ona s najvećim brojem glasova.

U modelu za broj procjenitelja, odnosno broj stabala slučajne šume odabrano je 100 stabala:

```
from sklearn.ensemble import RandomForestClassifier

model_rc = RandomForestClassifier(n_estimators=100, random_state=42)
model_rc.fit(X_train, y_train)

y_pred = model_rc.predict(X_test)
accuracy = model_rc.score(X_test, y_test)
```

Odsječak koda 3..12: RFC

Točnost ovog modela je 0.8223888933216309.

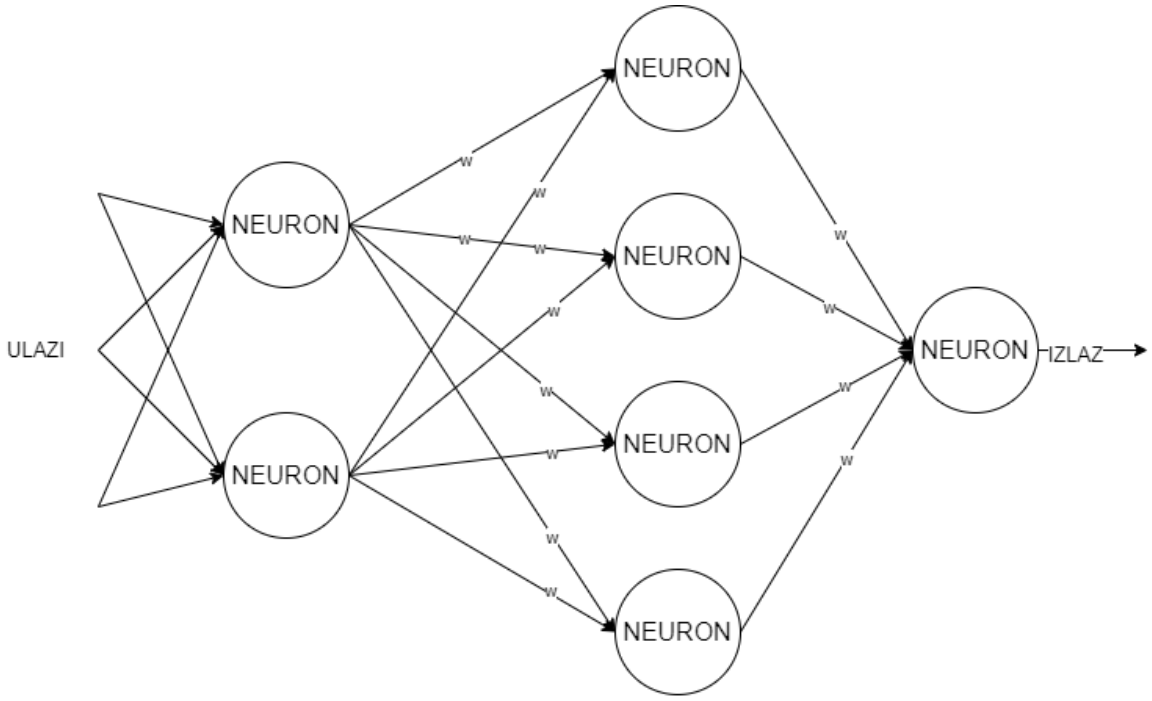
3.3.4. Neuronska mreža

Umjetne neuronske mreže su metode strojnog učenja koje simuliraju mehanizam učenja u biološkim organizmima. Ljudski živčani sustav sadrži stanice koje se nazivaju neuroni. Neuroni su povezani međusobno pomoću dendrita i aksona, a povezne regije između aksona i dendrita nazivaju se sinapse. Snaga sinaptičkih veza često se mijenja kao odgovor na vanjske podražaje. Ova promjena je način na koji se odvija učenje kod ljudi, a i svih ostalih životnih organizama.

Taj mehanizam učenja simulira se u umjetnim neuronskim mrežama, koje sadrže jedinice koje izvode određena računanja, a nazivaju se neuroni. Neuroni su međusobno

povezani težinama, koje imaju istu ulogu kao snaga sinaptičkih veza u biološkim organizmima. Svaki ulaz u neuron skalira se težinom koja utječe na funkciju u tom neuronu. Neuronska mreža izračunava funkciju ulaza propagirajući izračunate vrijednosti od ulaznih neurona do izlaznog neurona (ili više njih) koristeći težine kao međuparametre. Učenje se događa kada se težine među neuronima mijenjaju. Kao što su biološkim organizmima potrebni vanjski podražaji za učenje, vanjski podražaj u umjetnim neuronskim mrežama osigurava skup podataka za treniranje koji sadrži primjer parova ulaz-izlaz funkcije koju treba naučiti.

Usporedba s biološkim često je kritizirana kao vrlo loša karikatura načina funkcioniranja ljudskog mozga, no ipak principi neuroznanosti često su bili korisni u dizajniranju arhitektura neuronskih mreža. Drugi pogled je da su neuronske mreže izgrađene kao apstrakcije više razine klasičnih modela koji se koriste u strojnom učenju. Neuroni u mreži su inspirirani tradicionalnim algoritmima strojnog učenja poput regresije. Snaga neuronskih mreža je u spajanju mnogih jednostavnih ili osnovnih neurona te učenje težina različitih neurona kako bi se smanjila pogreška predikcije. Ako ovako gledamo, neuronska mreža se može gledati kao graf elementarnih jedinica (neurona) u kojem se veća snaga dobiva njihovim povezivanjem na određene načine.(Aggarwal, 2018)



Slika 3.5. Neuronska mreža

Kao što vidimo iz slike 3.5. neuroni su posloženi u slojeve te svaki neuron pripada određenom sloju. Nakon samog računanja unutar neurona prilikom "izlaska" iz neuronske mreže primjenjuju se aktivacijske funkcije. Aktivacijska funkcija je nelinearna funkcija koja se primjenjuju na rezultate linearne kombinacije (najčešće) ulaza i težine.

- sigmoidna funkcija - komprimira sve ulazne vrijednosti u raspon između 0 i 1

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Tanh odnosno tangens hiperbolni koji komprimira sve vrijednosti između -1 i 1

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- ReLU (Rectified Linear Unit) - najčešće korištena zbog učinkovitosti i jednostavnosti koja vraća vrijednost 0 ako je ulaz ReLU funkcije manji od 0, a inače vraća ulaz funkcije.

$$\text{ReLU}(x) = \max(0, x)$$

- Leaky ReLU - kako i samo ime kaže, "propusna" modifikacija ReLU funkcije i ona dopušta malom dijelu ulaza da prođe ako je on manji od nula

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{ako je } x \geq 0 \\ \alpha x & \text{ako je } x < 0 \end{cases}$$

gdje je α mala konstanta (najčešće 0.01)

- ELU (Exponential Linear Unit) - sličan ReLU ali boljih performansi jer koristi eksponencijalnu funkciju kod pojave negativnih vrijednosti

$$\text{ELU}(x) = \begin{cases} x & \text{ako je } x \geq 0 \\ \alpha(e^{x} - 1) & \text{ako je } x < 0 \end{cases}$$

gdje je α konstanta (najčešće 1.0)

- Softmax - koristi se za višeklasne klasifikacijske probleme, pretvarajući logite (izlaze neurona prije aktivacijske funkcije) u vjerojatnosti koje se zbrajaju do 1

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Sama neuronska mreža uči se povratnom propagacijom koja se odvija u iteracijama. Prvotno slučajno postavimo težine te zatim provučemo ulaz kroz neuronsku mrežu. Nakon što dobijemo izlaz usporedimo ga sa stvarnim izlazom i ta je razlika pogreška mreže. Nakon izračuna pogreške propagiramo pogrešku unatrag lančano od kraja do početnih neurona te se nakon toga ažuriraju težine. Iterira se ili unaprijed određeni broj puta ili kada se uvidi da je da je došlo do prestanka poboljšavanja modela ili rano zaustavljanje kako bi se zaustavilo prenaučavanje.

Za treniranje neuronske mreže odabran je Sequential odnosno linearni stack slojeva. Sastoji se od 6 slojeva i jednog izlaznog sloja. Kombinacija je Dense i Dropout slojeva. Dense sloj je potpuno povezani sloj te prva dva Dense sloja imaju 128 neurona dok 3 Dense sloj ima 64 neurona te sva tri sloja koriste ReLU aktivacijsku funkciju. Dropout slojevi služe kako bi se reguliralo učenje da ne bi došlo do prenaučavanja tj overfittinga i ona radi na principu, kako joj samo ime kaže, izbacivanju slučajno određenih neurona u iteracijama treniranja. Prva dva dropouta izbacuju 20% dok zadnji dropout izbacuje 50 % neurona. Izlazni Dense sloj ima 4 neurona (broj klasa koje predviđamo) te aktivacijska funkcija zadnjeg sloja je softmax.

```
from keras.models import Sequential
from keras.layers import Dense, Dropout

model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test
    ))

test_loss, test_acc = model.evaluate(X_test, y_test)
```

Odsječak koda 3..13: Nuronska mreža

Točnost ovog modela je 0.72142742

Sve točnosti treniranih modela:

Metoda	Točnost
KNN (K-nearest neighbors)	0.6149
Logistička regresija	0.6687
Slučajne šume	0.8224
Neuronska mreža	0.7214

3.4. Export modela

S obzirom na to da je za treniranje modela odabrano programiranje u Python programskom jeziku, a razvijanje web aplikacije u Javi na serverskoj strani model je izvezen kako bi se mogao direktno na serveru koristiti. Korišten je format pmml. Pmml, Predictive Model Markup Language, je standardni XML format za spremanje modela te omogućava interoperabilnost između aplikacija koje koriste modele umjetne inteligencije bez

potrebe za reimplementacijom istog modela.

```
from sklearn2pmml import PMMLPipeline, sklearn2pmml
pipeline = PMMLPipeline([
    ("classifier", RandomForestClassifier(n_estimators=100, random_state=42))
])
pipeline.fit(X_train, y_train)
sklearn2pmml(pipeline, "model-fpl.pmml", with_repr = True)
```

Odsječak koda 3..14: Export modela u fomratu pmml

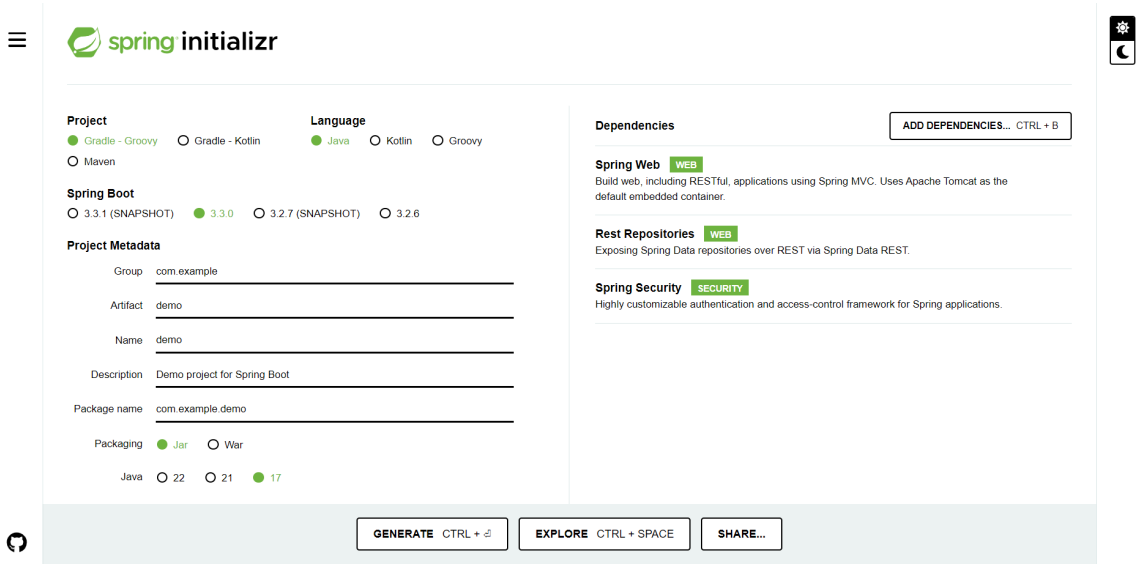
4. Web aplikacija za pomoć pri sastavljanju momčadi

Web aplikacija koja je napravljena služila bi kao pomoćnik u slaganju momčadi igračima Fantasy premier league igre. Pruža mogućnosti pregleda timova i nogometaša i njihovih nedavnih performansi te mogućnost predikcije performanse u sljedećoj utakmici. Web aplikacija na backend strani razvijena je koristeći Javu, odnosno Spring Boot framework dok je na frontend strani korišten Typescript odnosno React. Korištena baza podataka je Postgresql.

4.1. Tehnologije

4.1.1. SpringBoot

SpringBoot je radni okvir, odnosno framework, programskog jezika Java koji se primarno koristi za razvoj web servera. Jako je rasprostranjen i po mnogima najbolja opcija za enterprise aplikacije. Spring Boot nadogradnja je na Spring, radni okvir koji olakšava razvijanje aplikacija u Javi. Spring boot podržava sve opcije kao Spring te dodatno olakšava razvoj aplikacija. Samo konfiguriranje projekta u spring boot-u je lakše, jednostavnije i brže. Samo konfiguriranje početnog projekta i uključivanje dodatnih paketa u početni projekt odrađeno je korištenjem Spring Initializr, poznati pomoćnik u kreiranju Spring Boota koji omogućava početni odabir verzija jave, package managera, meta podataka o projektu te raznih dependency-a. Sučelje je vrlo jednostavno i intuitivno što možemo i vidjeti iz priloženog screenshota 4.1.



Slika 4.1. Spring initializr

Java

Java, prvotnog naziva Oak, je razvijena u kompaniji Sun Microsystems 1991. godine za ugrađene čipove dok je 1995. godine preimenovana u Java te je tada redizajnirana za razvoj web-aplikacija. (Liang, 2015) Objektno-orijentirani je programski jezik koji se koristi u razne svrhe kao što su: Razvoj web aplikacija, razvoj mobilnih aplikacija, primjena u umjetnoj inteligenciji, razvoj desktop aplikacija, IOT i mnoge druge primjene.

Baš zbog svoje široke primjene i velikog broja radnih okvira dostupnih u ovom programskom jeziku mogla bi se svrstati u top 3 programska jezika. Isto tako postoji mnogo drugih prednosti koje Java kao programski jezik ima, kao što je Prenosivost - omogućeno korištenjem JVM-a (Java Virtual Machine) koji daje mogućnost da se Javin bytecode izvršava na bilo kojem operativnom sustavu koji ima JVM. Robusnost, dizajnirana je da bude sigurna i robusna, s jako dobrim upravljanjem memorijom kao i upravljanje smećem za kojeg je zadužen garbage collector. Visoke performanse , multithreading i još brojne prednosti programskog jezika dovele su ju u vrlo široku rasprostranjenost. Primjer dijela servisa napisanog u Javi (maknuti importi zbog lakšeg prikaza u radu):

```
package fplbot.player;

@Service
@RequiredArgsConstructor
public class PlayerService {

    private final PlayerRepository playerRepository;
    private final PlayerOuterService playerOuterService;
    private final ModelMapper modelMapper;

    private final ObjectMapper objectMapper;

    public PlayerDto getPlayer(Short id) throws JsonProcessingException {
        var player = getPlayerEntity(id);
        var body = playerOuterService.getPlayerFromOuterService(id);

        var playerDto = modelMapper.map(player, PlayerDto.class);
        if (body != null) {
            playerDto.setLastPoints(sortByKickoffTimeAndGetLastFive(objectMapper.
    readValue(body, ElementSummary.class), 5));
        }

        return playerDto;
    }

    public Player getPlayerEntity(Short id) {
        return playerRepository.findById(id).orElseThrow();
    }

    public History getLastHistory(Short id) throws JsonProcessingException {
        var body = playerOuterService.getPlayerFromOuterService(id);
        var elementSummary = sortByKickoffTimeAndGetLastFive(objectMapper.readValue(
    body, ElementSummary.class), 1);
        if (elementSummary.isEmpty()) {
            return null;
        }
        return elementSummary.get(0);
    }

        public List<PlayerDto> getPlayers() {
        return playerRepository.findAll().stream().map(player -> modelMapper.map(
    player, PlayerDto.class)).toList();
    }
}
```

Odsječak koda 4..1: Service napisan u javi

JPA

ORM, Object relational mapping, je proces u kojem se Java objekti pretvaraju u tablice baze podataka i obratno. To omogućuje interakciju s relacijskom bazom bez korištenja SQL-a, structured query languange koji se koristi za rad s bazama podataka. Jakarta Persistence API, prijašnjeg naziva Java Persistence API, je specifikacija koja definira kako rukovati podatcima u Java aplikacijama. Primarni fokus JPA je ORM sloj. (**?**)

Hibernate JPA je samo specifikacija pa samim time ju je potrebno implementirati. Standardna implementacija JPA specifikacije je Hibernate, Javin ORM radni okvir koja obavlja pretvorbe Java klase u modele baze podataka i obratno te sve tipove podataka iz Javinih tipova u tipove baze i obratno. JPA i Hibernate olakšavaju pristup podatcima iz Spring Boot aplikacije te su zato izabrani za korištenje. Hibernate omogućava pisanje metoda, u JPA Repozitorijima (sučelje za pristup bazi), na način da "ljudskim" jezikom definiramo što nam je potrebno te to zatim Hibernate pretvara u SQL upite prema bazi i vraća rezultat. S obzirom na to da u SQL upitima postoje mogućnosti postavljanja određenih uvjeta Hibernate također daje tu mogućnost predavanjem argumenata, a sami uvjeti definirani su u imenu. Isto tako sam JPARepository nudi mnoge, out of the box, metode poput *count, delete, exists, existsById, findById, findAll, save, saveAll, saveAllAndFlush...*

```
@Repository
public interface PlayerRepository extends JpaRepository<Player, Short> {
    Optional<Player> findByFullNameContaining(String fullName);
}
```

Odsječak koda 4..2: JPA Repository

Beanovi, context i model

Kao što je već navedeno Spring boot mnoge stvari olakšava i radi za nas pa tako postoji kontekst o kojem vodi brigu i registrira tako zvane Beanove. Bean su objekti kojima upravlja Spring IoC (inversion of control) kontejner. Vrlo često su to singletone objekti koji se injektiraju na mjesta gdje se koriste. To su uglavnom objekti koji rade neku logiku unutar same aplikacije. Postoji više vrsta beanova: *Component, Service, Repository, Controller, Configuration* koji se svi u aplikaciji mapiraju u bean-ove.

Pa tako imamo primjer beana evaluatora modela koji prima podatke i vraća izlaz s modela, a prilikom inicijalizacije se model učitava iz pmml file-a koji je exportan iz Pythona (bez importa radi lakšeg prikaza u radu).

```
package fplbot.prediction;

@Component
public class FplEvaluator {

    @Bean
    public Evaluator evaluator() {
        Evaluator evaluator;
        try {
            var file = new File("src/main/java/fplbot/prediction/model-fpl.pmml");
            //start the timer
            long startTime = System.currentTimeMillis();
            evaluator = new LoadingModelEvaluatorBuilder()
                    .load(file)
                    .build();
            //end the timer
            long endTime = System.currentTimeMillis();
            System.out.println("Time to load the model: " + (endTime - startTime) + "
    ms");

            // Perforing the self-check
            evaluator.verify();

            // Printing input (x1, x2, .., xn) fields
            List<InputField> inputFields = evaluator.getInputFields();
            System.out.println("Input fields: " + inputFields);
        } catch (IOException e) {
            throw new RuntimeException(e);
        } catch (ParserConfigurationException e) {
            throw new RuntimeException(e);
        } catch (SAXException e) {
            throw new RuntimeException(e);
        } catch (JAXBException e) {
            throw new RuntimeException(e);
        };
        return evaluator;
    }
}
```

Odsječak koda 4..3: Evaluator bean

4.1.2. React

React je besplatna Javascript biblioteka otvorenog koda koja se koristi za izgradnju korisničkog sučelja. Razvijen u Facebook-u, današnjoj tvrtki Meta, i još ga uvijek održavaju. React se najčešće koristi kao baza za singlepage aplikacije. (Arancio, 2021)

Najlakše ga se može opisati kao spoji HTML-a i JavaScript-a. Korištene su određene komponente iz MaterialUI-a. Uz komponente korišten je isto tako *čisti* CSS kao i styled komponente za stiliziranje elemenata. Iako korišteni typescript za ekstenziju datoteka ima samo ts velika većina ostalih datoteka ima tsx ekstenziju. TSX je ekstenzija koja se koristi za typescript datoteka koje sadrže JSX sintaksu. JSX sintaksa je nalik HTML-u koja se koristi za definiranje React komponenti.

HTML

Hypertext Markup Language, je, jezik označavanja koji definira strukturu web stranica.(Kolade, 2021). Razvijen je od strane fizičara na CERN-u u Europskom laboratoriju za fiziku čestica. Web stranice građene su od HTML elemenata. Svaki element označen je svojim tagom koji mu daje različitu ulogu. Tako imamo tag-ove strukturu: *<html>, <head>, <title>, <body>, <header>*, tekstualne tagove: *<h1>, <p>, <p>*, medijski tagovi: *<img>, <audio>, <video>* i mnoge druge.

4.1.3. CSS i MaterialUI

CSS, Cascading Style Sheets, stilski jezik koji služi za stiliziranje HTML dokumenata. Omogućava kreiranja pravila prema kojima će se odrediti izgled određenog HTML elementa ili cijel stranice. Može se odrediti boja, širina, visina, zaobljenost udaljenost od drugih elemenata, raspored elemenata i mnoge druge stvari.

Primjer css datoteke:

```
.text-field {
    border-color: aliceblue;
    border-radius: 5px;
    background: aliceblue;
    margin-bottom: 0px;
    width: 90%;
}

.predict-form {
    display: flex;
    flex-direction: column;
    align-items: center;
    min-width: 100%;
}

.error-message {
    color: red;
    font-size: 14px;
    margin-top: 0px;
    font-weight: lighter;

}

.submit-button {
    background: #101822FF;
    color: aliceblue;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    margin-top: 30px;
    cursor: pointer;
}
```

Odsječak koda 4..4: CSS

Material UI je open source library otvorenog koda koja nudi gotove stilizirane komponente koje se mogu iskoristiti. Uz to daje i mogućnost određenog modificiranja gotovih komponenti što olakšava programerima da elemente prilagodi svojim potrebama i željama. To uvelike ubrzava razvoj jer se *ne gubi* vrijeme na stiliziranje i brigu o izgledu nego se to vrijeme može iskoristiti za sam razvoj logike, a isto tako nudi ponovnu iskoristivost elemenata, što je jedna od bitnih ideja programiranja kako bi se uštedjelo vrijeme.

Javscript

Skriptni jezik koji daje funkcionalnost web aplikacijama (i web stranicama). Razvijen u rujnu 1995. godine u američkoj tvrtki Netscape, kada ga je Brandan Eich razvio u samo 10 dana. Prvotnog imena Mocha, ubrzo poznatijeg kao LiveScript, a kasnije Javascript. (DeGroat, 2019)

Typescript Typescript je tipizirani jezik koji se prevodi u Javascript. Razvijen u Microsoftu s prvim izdanjem 2012ġodine dok 2014ġodine izlazi prva verzija. (Turner, 2014) Prednost Typescripta nad Javascriptom je, kako samo ime kaže, je to da je tipizirani jezik te je tako manja vjerojatnost grešaka.

S obzirom na to da je većina datoteka tsx, izdvajamo dio koda koji je čisti typescript:

```
import {Team} from "../Teams/team";
import {Player} from "../Players/player";
import axios from "axios";
import {Evaluation} from "../Predict/prediction";

const serverApi = process.env.SERVER_API === undefined ? 'http://localhost:8080/api/1'
    : process.env.SERVER_API;
const TEAMS_ENDPOINT = `${serverApi}/teams`;
const PLAYERS_ENDPOINT = `${serverApi}/players`;
const PREDICTION_ENDPOINT = `${serverApi}/prediction`;
export const getTeams = async (): Promise<Team[]> => {
    const response = await axios.get(TEAMS_ENDPOINT);
    return response.data;
}

export const getPlayer = async (playerId: number): Promise<Player> => {
    const response = await axios.get(`${PLAYERS_ENDPOINT}/${playerId}`);
    return response.data;
}

export const getPrediction = async (playerId: number, minutes: number, transfersIn:
    number, transfersOut: number): Promise<Evaluation> => {
    const response = await axios.post(`${PREDICTION_ENDPOINT}`, {
        playerId,
        minutes,
        transfersIn,
        transfersOut
    });
    return response.data;
}
```

Odsječak koda 4..5: Typescript

Dok s druge strane imamo tsx datoteku sa JSX sintaksom:

```
import {useParams} from "react-router-dom";
import {useContext} from "react";
import {TeamsContext} from "../App";
import {ColumnDiv} from "../components/Components";
import TeamDetails from "./TeamDetails";
import TeamPlayers from "./TeamPlayers";

export default function TeamPage() {
    const {teamId} = useParams();
    const { teams, loading } = useContext(TeamsContext);
    const team = teams.find(team => teamId ? team.id === parseInt(teamId) : false);

    return (
        loading ? <div>Loading...</div> :
        <ColumnDiv style={{marginBottom: '10px', minWidth: '100%'}}>
            <TeamDetails team={team!}/>
            <TeamPlayers players={team?.players!}/>
        </ColumnDiv>
    )
}
```

Odsječak koda 4..6: Tsx komponenta

4.1.4. PostgreSQL

PostgreSQL je open-source objektno-relacijski sistem koji koristi i proširuje SQL jezik kombiniran s mnogim značajkama koji sigurno sprema i skalira najkompliciranija podatkovna opterećenja. Sami početci dolaze iz 1986godine kao projekt POSTGERS na *University of California at Berkeley*.(The PostgreSQL Global Development Group)

4.1.5. Maven

Maven je alat koji pomaže u upravljanju softverskim projektom, najčešće Javinim. Može upravljati *buildom* projekta, izvještajem i dokumentacijom. Upravlja ovisnostima definiranim u pom.xml-u (Project Object Model). Prilikom spomenutog Maven *builda* izgrađuje se projekt te se isto tako može prolaziti kroz određene faze, koje je moguće i sam definirati isto tako imamo pluginove koji se definiraju i proširuju funkcionalnosti izgradnje projekta. Isto tako vrlo se jednostavno integrira s alatima za kontinuiranu integraciju i kontinuiranu isporuku (CI/CD) kao što je npr. Jenkins, Babmboo, Gitlab CI itd.

```
<dependencies>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>
              <groupId>org.projectlombok</groupId>
              <artifactId>lombok</artifactId>
            </exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

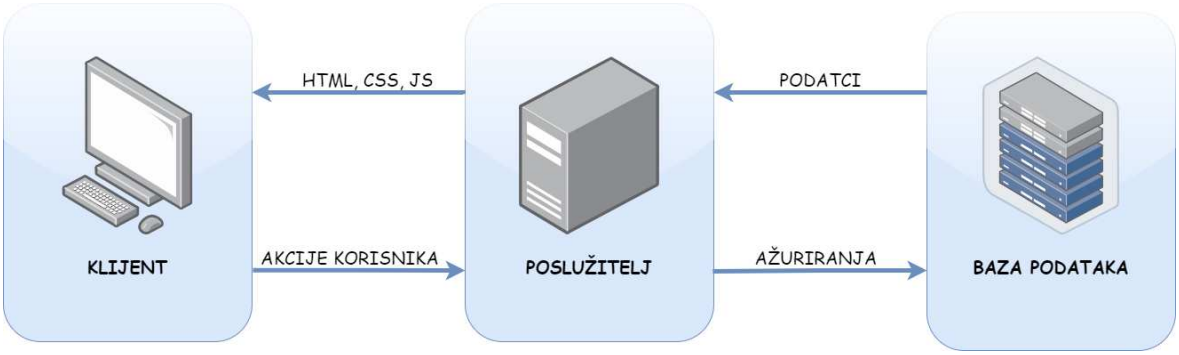
Odsječak koda 4..7: Definiranje ovisnosti i plugina

4.2. Arhitektura i dizajn sustava

Arhitektura aplikacije može se podijeliti na tri dijela, klijentska strana odnosno web-preglednik, poslužiteljska strana odnosno server te baza podataka. Korišten je MVC (Model-View-Controller) obrazac koji razdvaja poslovnu logiku od prikaza. Poslovna logika i rukovanje podatcima smještana je u modelu, prikaz podataka odrađen je na strani pogleda (view) dok se kontroler brine za upravljanje i usmjeravanje naredbi između pogleda i modela

4.2.1. Web preglednik

Za implementaciju web preglednika odabran je React.js, po mnogima najpoznatija i najkorištenija Javascript knjižnica. Web preglednik zadužen je za prikaz podataka, interakciju s web-poslužiteljom te interakciju s klijentom. S web-poslužitelja dohvaća podatke ili šalje određene podatke na obradu (najčešće dobivene od klijenta). Komunikacija sa serverom odvija se preko HTTP protokola, odnosno sigurne verzije tog protokola; HTTPS-a.



Slika 4.2. Arhitektura sustava

Prilikom komunikacije web-poslužitelj vraća odgovore gdje svaki ima određeni status kod koji određuje način na koji je zahtjev odrađen odnosno rezultat obrade pa tako postoje:

1xx: informacijski odgovor

2xx: uspješan zahtjev

3xx: preusmjeravanje

4xx: greška na klijentskoj strani

5xx: greška na poslužitelju

4.2.2. Web poslužitelj

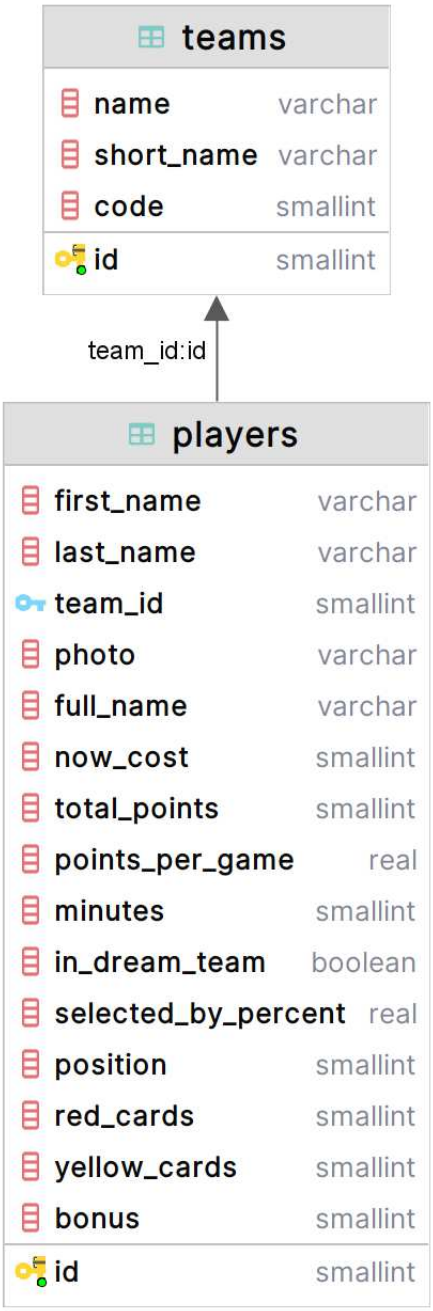
Na poslužiteljskoj strani korišten je programski jezik Java, odnosno Spring Boot. Podjela na poslužiteljskoj strani je na kontrolere, servise i repozitorije. Kontroleri su klase koje definiraju API odnosno primaju zahtjeve iz vana i prosljeđuju na obradu. U njima se može odrađivati određena validacija dobivenih podataka kao i autentifikacija/autorizacija. Api je razvijen prema REST konvenciji pa je samim time to restful api. Servisi su klase u kojima se obrađuju podatci i u kojim je smještena poslovna logika, a repozitoriji su klase (odnosno sučelja) za pristup podatcima. Primjer kontrolera s izostavljenim importima radi lakšeg prikaza:

```
package fplbot.player;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/1/players")
public class PlayerController {

    private final PlayerService playerService;
    @GetMapping
    public List<PlayerDto> getPlayers() {
        return playerService.getPlayers();
    }

    @GetMapping("/{id}")
    public PlayerDto getPlayer(@PathVariable Short id) throws JsonProcessingException
    {
        return playerService.getPlayer(id);
    }

    @GetMapping("/search")
    public ResponseEntity<PlayerDto> getPlayerByFullName(@RequestParam String fullName
    ) {
        return playerService.getPlayerByFullName(fullName)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }
}
```

Odsječak koda 4..8: Player kontroler

4.2.3. Baza podataka

Prilikom razvoja odlučeno je da će se koristiti relacijska baza te PostgreSQL sistem za upravljanje podatcima. Baza podataka poprilično je jednostavna jer je cijela ideja diplomskog rada prediktivni model pa tako imamo samo dvije tablice baze: players i teams. Iz samog imena jasno je kako players tablica pohranjuje podatke o nogometašima dok teams tablica pohranjuje podatke o timovima nogometaša.



Slika 4.3. ER dijagram

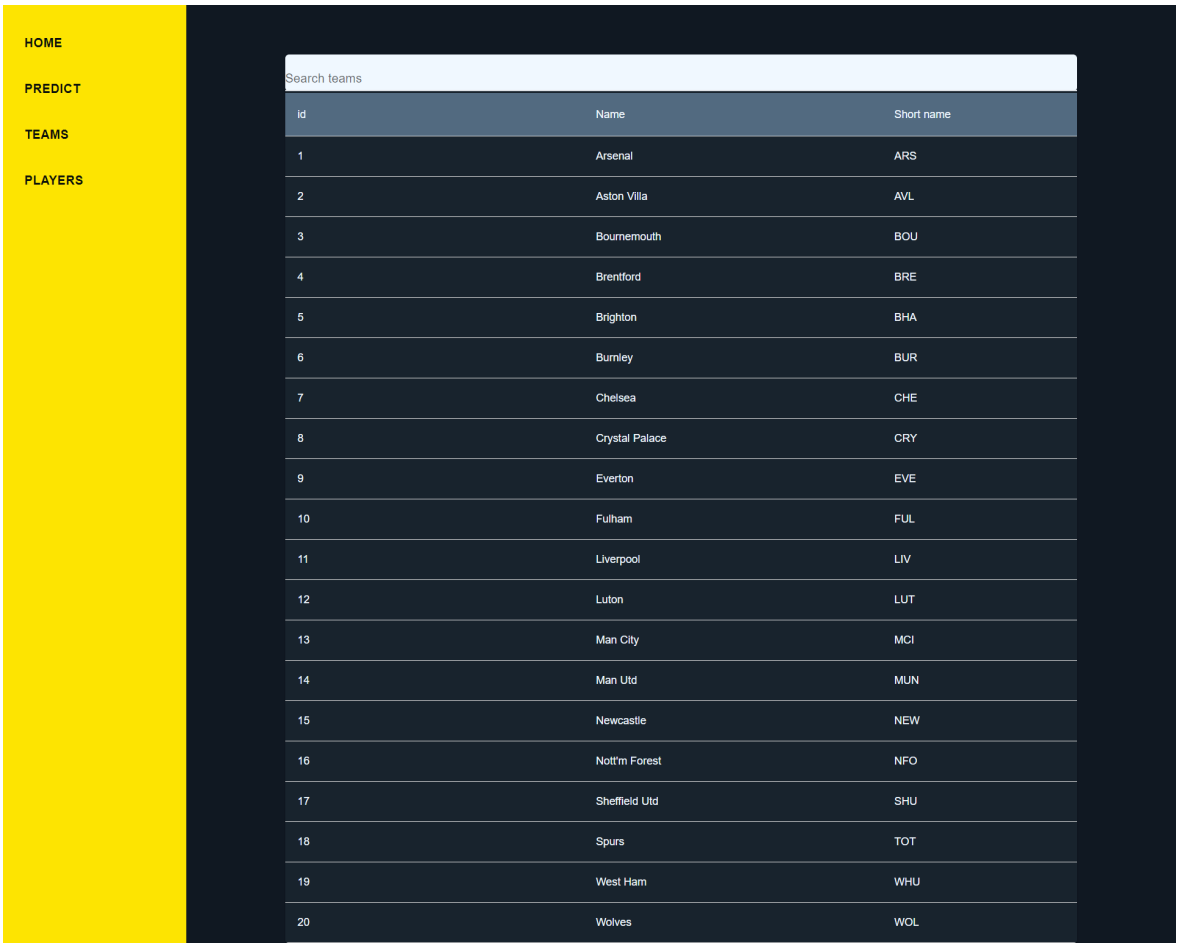
4.3. Opis rada

Web aplikacija služi kako bi igrači igre Fantasy premier league dobili pomoć prilikom sastavljanja svoje momčadi. Korištenjem aplikacije igrači mogu pretraživati i pregledavati informacije o nogometašima i timovima. Isto tako mogu vidjeti trenutnu formu igrača te zatražiti predikciju za određenog igrača.

Prilikom dolaska na web aplikaciju igrače dočekuje početna stranica koja ima određene novosti o ligi. Na lijevoj strani nalazi se navigacija koja služi za pristup ostalim dijelovima aplikacije.

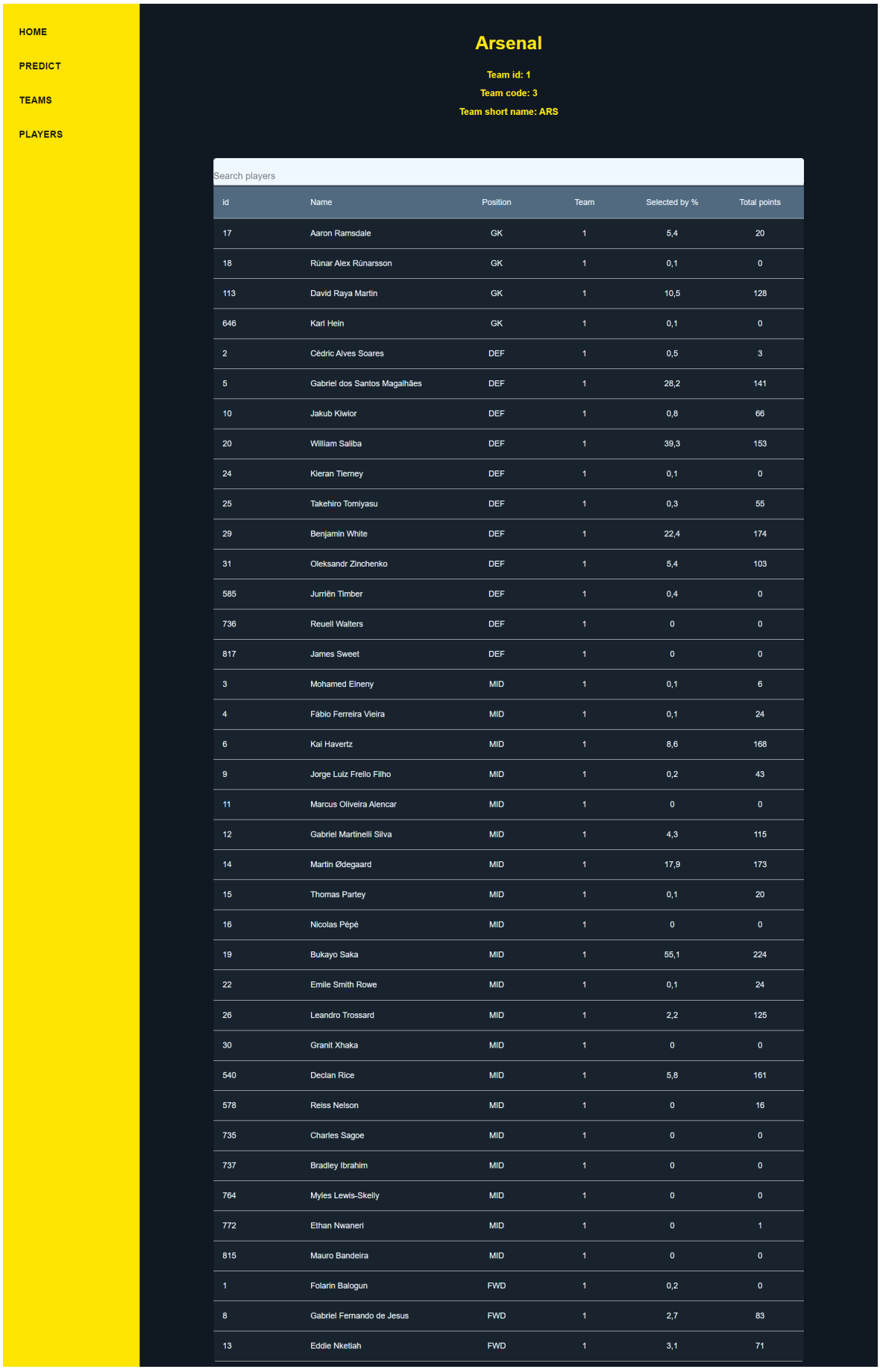
Slika 4.4. Početna stranica

Odabirom *Teams* opcije iz navigacije otvara se popis svih timova iz Engleske Premier lige gdje je svaki red iz tablice moguće odabrati koji vodi na podatke o timu. 4.6.



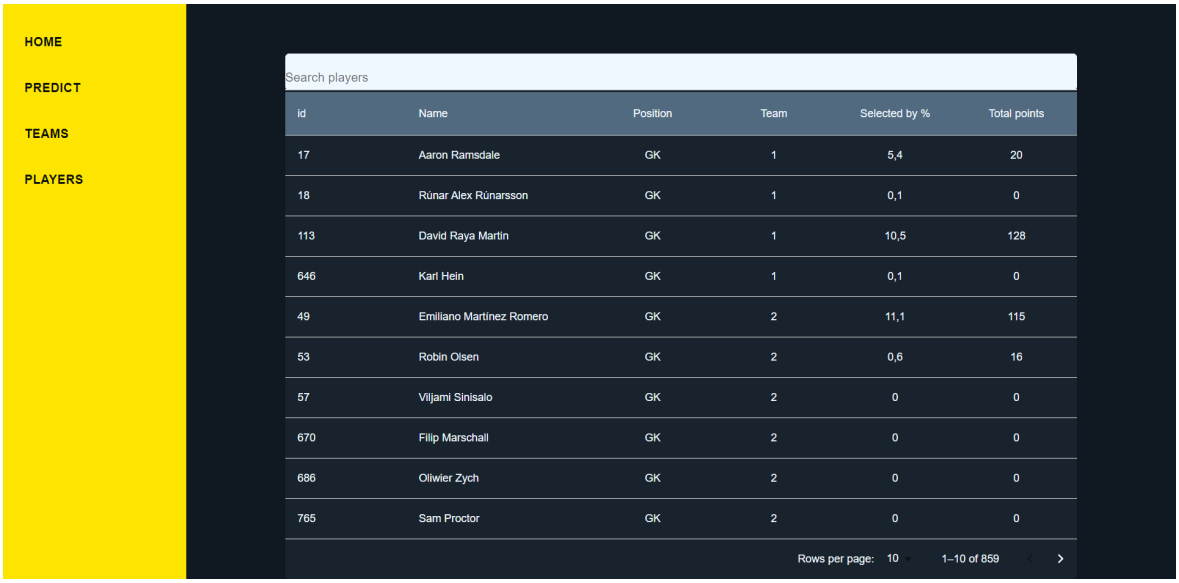
Slika 4.5. Pregled timova

Podatci o timu, osim osnovnih podataka o timu nudi i popis svih nogometaša toga tima. Svaki red s tog popisa moguće je odabrati koji vodi na podatke o nogometašu te prikaz njegove forme. 4.8.



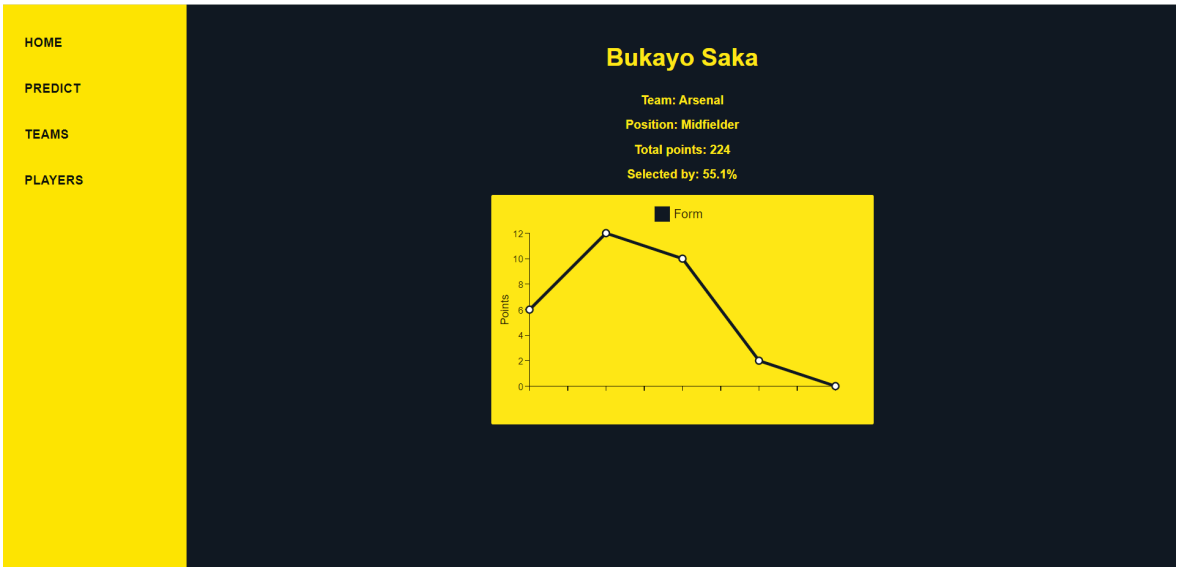
Slika 4.6. Podatci o timu

Ako se u navigaciji odabere *Players* opcija to nas odvodi na popis svih nogometaša u paginiranoj opciji uz mogućnost odabira broja nogometaša po stranici. Isto tako omogućena je i pretraga nogometaša. Odabirom bilo kojeg nogometaša otvara se nova stranica s podatcima nogometaša i njegovoj formi. 4.8.



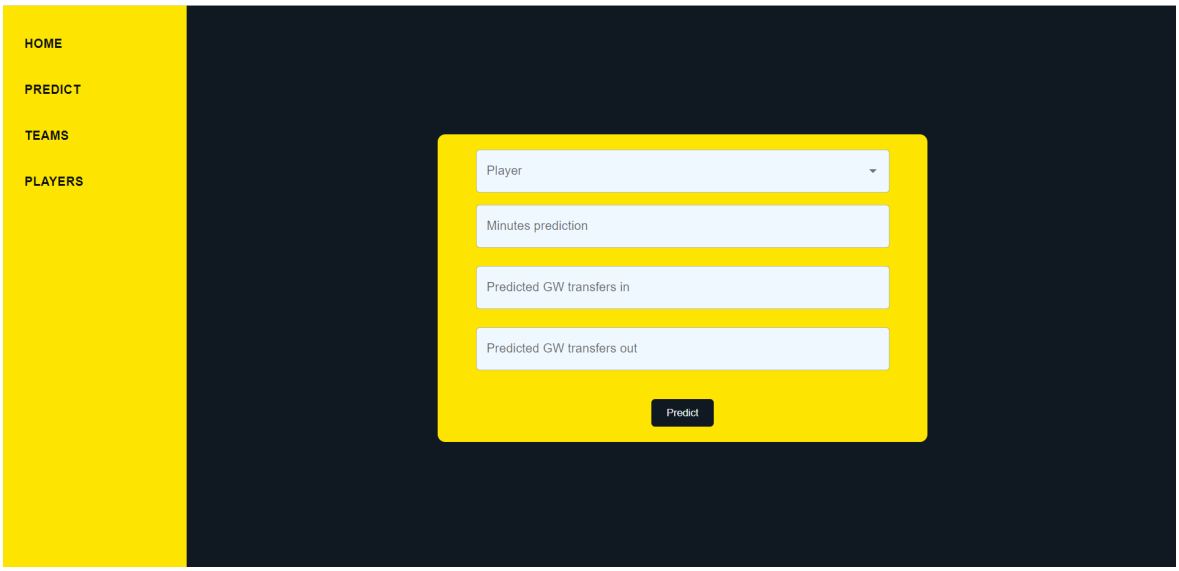
Slika 4.7. Pregled nogometaša

Podatci o nogometašu prikazuju podatke iz stvarnog svijeta (kao npr. tim za koji igra) te podatke iz igre Fantasy premier league (kao npr. ukupan broj bodova koji je ostvario). Isto tako moguće je vidjeti formu u zadnjih 5 utakmica igrača.



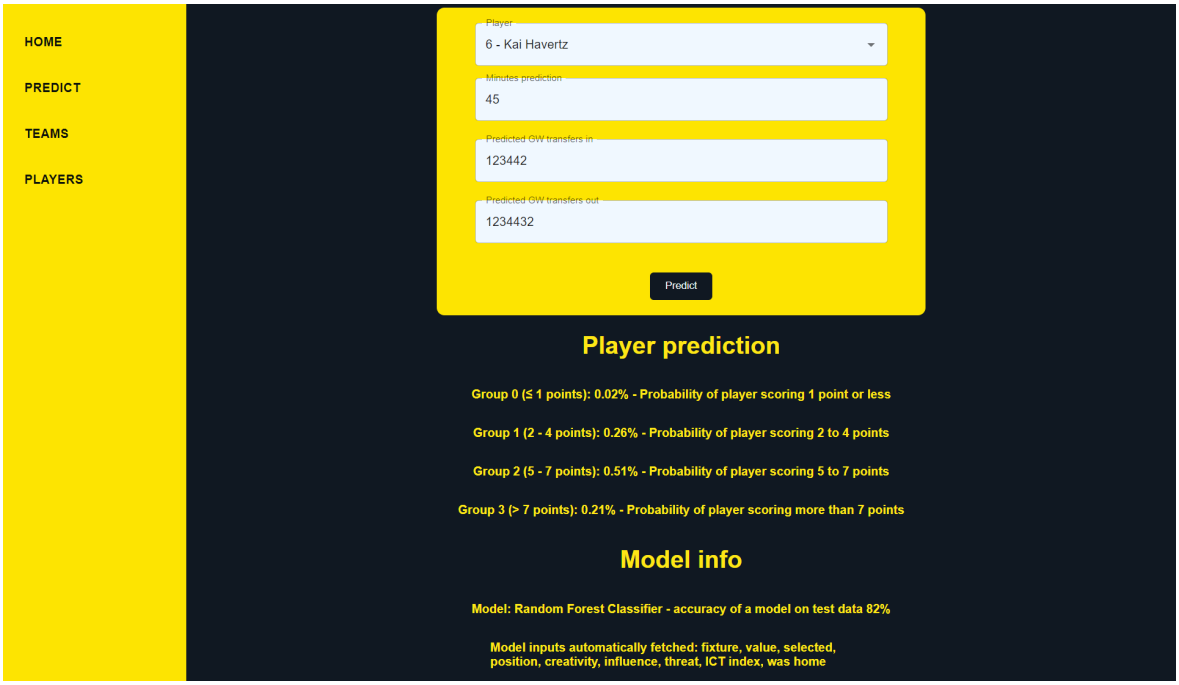
Slika 4.8. Podatci o nogometašu

Odabirom *Predict* opcije iz navigacije otvara se forma za popunjavanje potrebnih podataka za predikciju, a to su nogometaš za kojeg se želi predikcija, očekivani broj ulaznih transfera, očekivani broj izlaznih transfera te očekivani broj minuta koje će nogometaš odigrati.



Slika 4.9. Forma za predikciju

Nakon što se odradi predikcija na serveru dobije se povratni odgovor prikaz o predikciji te informacije o vjerojatnosti svake od predikcija kao i informacije o modelu.



Slika 4.10. Forma za predikciju

5. Zaključak

Cijelim procesom izgradnje modela, od prikupljanja podataka, analize podataka te procesiranja istih, a na kraju i treniranjem modela isprobano je puno novih metoda i tehnologija. Isto tako prilikom izgradnje web aplikacije dotaknuto je puno područja softverskog razvoja; baza podataka, izgradnja web poslužitelja, dizajniranje i izgradnja web klijenta. Umjetna inteligencija prešla je iz tehničkog termina kojim se koriste samo inženjeri u svakodnevicu većine ljudi. Ovim radom spojen je rad na nečemu novom a isto tako zanimljivom te je spojena tehnologija s igrom koja se bazira na stvarnosti.

Razvoj modela umjetne inteligencije uvelike ovisi o podatcima koji su dostupni što se moglo iskusiti i tijekom izrade ovog modela, ali čak uz podatke koji nisu bili najbolji, određene modifikacije dovele su do uspješno kreiranog modela te isto tako do vrlo dobrog rezultata. Web aplikaciji također je predana značajna količina vremena kako bi što bolje podržala model za predikciju.

Inicijalna zamišljena ideja, kao i svi popratni ciljevi i zadatci su uspješno odrađeni, no uvijek može bolje te postoje mjesta za poboljšanja. Podatke koji su korišteni za izradu modela moglo bi se proširiti podatcima o zadnjim sezonama te pokušati istražiti mogućnost prikupljanja podataka o prošlim sezonama. Isto tako mogli bi se probati neki drugi prediktivni modeli ili drugačija konfiguracije neuronske mreže. S obzirom na to da su LLM-ovi sve zastupljeniji neki od njih mogli bi se iskoristiti za komunikaciju s klijentima te im olakšati sastavljanje momčadi.

Literatura

C. C. Aggarwal. *Neural Networks and Deep Learning A Textbook*. Springer, 2018.

S. Arancio. Reactjs: A brief history, 2021. URL https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f. [Pristupljeno: 2024-06-15].

T. J. DeGroat. The history of javascript: Everything you need to know, 2019. URL https://www.springboard.com/blog/data-science/history-of-javascript/. [Pristupljeno: 2024-06-16].

T. Hastie, R. Tibshirani, i J. Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction, Seconedition*. Springer, 2009.

Vijay Kanade. What is logistic regression? equation, assumptions, types, and best practices, 2022. URL https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/. [Pristupljeno: 2024-06-13].

C. Kolade. What is html – definition and meaning of hypertext markup language, 2021. URL https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/. [Pristupljeno: 2024-06-08].

O. Kramer. *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Springer, 2013.

Y. D. Liang. *Introduction to Java programming (10th ed.)*. Prentice Hall, 2015.

M. Lutz. *Programming Python*. O'Reilly Media, Inc., 2011.

Python Software Foundation. General python faq, 2024. URL https://docs.python.

org/3/faq/general.html#why-was-python-created-in-the-first-place. [Pristupljeno: 2024-06-09].

The Scout. Fpl basics: Scoring points, 2024. URL https://allaboutfpl.com/2024/03/highest-fpl-gameweek-scores-in-fpl-history/. [Pristupljeno: 2024-06-08].

The PostgreSQL Global Development Group. (postgresql) about. URL https://www.postgresql.org/about/. [Pristupljeno: 2024-06-16].

J. Turner. Announcing typescript 1.0, 2014. URL https://devblogs.microsoft.com/typescript/announcing-typescript-1-0/. [Pristupljeno: 2024-06-16].

Guido van Rossum. A brief timeline of python, 2009. URL https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html. [Pristupljeno: 2024-06-09].

Bill Venners. The making of python a conversation with guido van rossum, part i, 2003. URL https://www.artima.com/articles/the-making-of-python. [Pristupljeno: 2024-06-09].

Sažetak

Sustav za predviđanje rezultata i optimizaciju timova Fantasy Premier lige

Marko Šošić

Tema završnog rada bila je razvoj sustava za predviđanje rezultata i optimizaciju timova Fantasy Premier lige. Sustav se sastoji od dva dijela, modela koji je izgrađen u programskom jeziku Python te od web aplikacije koja pruža mogućnost korištenja podataka te pomoćne informacije o nogometašima i timovima. Web aplikacija izrađena je koristeći Spring Boot na serverskoj strani te React na klijentskoj strani, a odabrana baza podataka je PostgreSQL. Model služi za predikciju budućeg broja bodova nogometaša na osnovu ulaznih parametara, od kojih je dio automatski dohvaćen, a dio je popunjen od strane klijenta. Web aplikacija pruža funkcionalnost pregleda podataka o nogometašima i timovima te mogućnost upotrebe modela za predviđanje bodova.

Ključne riječi: prediktivni model; Fantasy premier league; Web aplikacija

Abstract

System for Fantasy Premier League results prediction and team optimization

Marko Šošić

The topic of the thesis was the development of a system for Fantasy Premier League results prediction and team optimization. The system consists of two parts: a model built in the Python programming language and a web application providing access to footballer data and supplementary information about footballers and teams. The web application was developed using Spring Boot on the server-side and React on the client-side, with PostgreSQL selected as the database. The model serves to predict the future points of football players based on input parameters, some of which are automatically fetched while others are filled in by the client. The web application offers functionality for viewing player and team data, as well as using the model to predict points.

Keywords: prediction model; Fantasy premier league; web application