Univerzitet u Beogradu

Mašinski fakultet

Master akademske studije

Industrija 4.0

Mašinsko učenje

Projektni zadatak

Predviđanje ishoda košarkaške utakmice Evrolige

Ocena projektnog zadatka:	Predmetni nastavnik:	prof. dr Mladen Nikolić		
	Asistent:	Nevena Ćirić		
Potpis nastavnika:	RB	Prezime i ime	Br. indeksa:	Potpis:
	1.	Filip Marković	4001/21	

Školska godina: 2022/2023.

SADRŽAJ

1. UVOD

Cilj ovog projektnog zadatka je pronalaženje modela koji najbolje predviđa ishod košarkaške utakmice u takmičenju Evroliga. Evroliga je evropsko internacionalno košarkaško takmičenje u kojem se svake godine takmiče najbolji evropski košarkaški klubovi sa ciljem da ponesu titulu najboljeg košarkaškog tima Evrope. Takmičenje je napravljeno 1958. godine i uskoro kreće 66. sezona takmičenja.

Slika 1 – Logo Evrolige

Podaci koji će biti korišćeni u ovom projektu su preuzeti sa sajta "Kaggle". U pitanju je skup podataka sa rezultatima Evrolige od 2003 do 2020. godine.[1] Skup podataka se sastoji od 33 atributa i 3831. instance. Instance predstavljaju pojedinačne utakmice Evrolige, a atributi različite podatke o tim mečevima.

Atributi su sledeći:

- DATE – datum kada se utakmica odigrala
- HT, AT – domaći, gostujući tim
- WINNER – pobednik meča, u ovom projektu ciljna promenljiva
- HS, AS – broj poena domaći, broj poena gosta na utakmici
- Q1H, Q2H, Q3H, Q4H – broj poena domaćina po četvrtinama
- Q1A, Q2A, Q3A, Q4A – broj poena gosta po četvrtinama
- OTH, OTA – broj poena domaćina i gosta u produžecima
- P1H, P2H – broj poena domaćina po poluvremenima
- P1A, P2A – broj poena gosta po poluvremenima
- P1T – ukupan broj poena u prvom poluvremenu na utakmici
- P2T – ukupan broj poena u drugom poluvremenu na utakmici
- HTT – ukupan broj poena na kraju prvog poluvremena
- FTT – ukupan broj poena na kraju meča
- GAPA1Q, GAPA2Q, GAPA3Q, GAPA4Q – maksimalna razlika u poenima između timova po četvrtinama

[1] https://www.kaggle.com/datasets/vadimgladky/euroleague-basketball-results-20032019?resource=download

2. PRETPROCESIRANJE

Pre nego što se se krene sa primenom različitih modela na podatke, potrebno ih je obraditi, tj. pretprocesirati.

Prvo što je potrebno je uraditi je uzeti samo podatke iz poslednjih 5 sezona u skupu, jer podaci iz 2003 teško da su imalo relevantni za predviđanje budućih rezultata. Da bi se to uradilo, biće zadržani samo podaci kod kojih se u datumu odigravanja nalazi godina “2016” ili kasnija godina. Time će u skupu podataka ostati 1018 instanci. Zatim će atribut “DATE” biti izbačen jer nema više potrebe za njime. Zatim je potrebno sve timove koji imaju više različitih naziva u skupu podataka svesti na jedan naziv i takođe izbaciti kineske timove koji ne učestvuju u Evroligi, a nekom greškom su se našli u skupu podataka (Slika 1).

```
In [7]: df['DATE'] = pd.to_datetime(df['DATE'], format='%d.%m.%Y')

In [8]: # Keep only instances from the last 5 seasons
        df = df[df['DATE'].dt.year >= 2016]

In [9]: # Drop the DATE column
        df = df.drop(columns=['DATE'])

In [10]: # Replace secondary teams names in 'HT' column
         df['HT'] = df['HT'].replace({'Crvena zvezda mts': 'Crvena Zvezda',
                                      'Anadolu Efes SK': 'Anadolu Efes',
                                      'Fenerbahce Ulker': 'Fenerbahce',
                                      'Khimki M.': 'Khimki'})

         # Replace secondary teams names in 'AT' column
         df['AT'] = df['AT'].replace({'Crvena zvezda mts': 'Crvena Zvezda',
                                      'Anadolu Efes SK': 'Anadolu Efes',
                                      'Fenerbahce Ulker': 'Fenerbahce',
                                      'Khimki M.': 'Khimki'})

         # Replace secondary teams names in 'WINNER' column
         df['WINNER'] = df['WINNER'].replace({'Crvena zvezda mts': 'Crvena Zvezda',
                                      'Anadolu Efes SK': 'Anadolu Efes',
                                      'Fenerbahce Ulker': 'Fenerbahce',
                                      'Khimki M.': 'Khimki'})

In [11]: # Delete instances with Chinese teams as they are not part of the EuroLeague
         df = df[(df['HT'] != 'Guandong') & (df['HT'] != 'Sichuan')]

In [12]: # Get a list of all unique teams
         unique_teams = set(df['HT'].unique()).union(set(df['AT'].unique()))
         unique_teams
```

Slika 2 – Pretprocesiranje podataka, prvi deo

Kolone OTH i OTA imaju većinski NaN vrednosti pa će njihove vrednosti (gde postoje) biti dodate u poene četvrte četvrtine, kao i drugog poluvremena, a zatim će biti izbačene iz skupa podataka. Takođe je potrebno atribute HT i AT, koji predstavljaju nazive timova u “string” formatu, numerički predstaviti. To će se obaviti “One-hot enkodiranjem”, čime će se svakom unikatnom nazivu tima dodeliti binarna vrednost. To će nažalost značajno povećati dimenzionalnost podataka, ali je za neke metode mašinskog učenja neophodno da bi uopšte funkcionisale, dok druge bolje rade sa binarnim vrednostima, nego sa rečima. Zatim se odbacuju svi atributi koji predstavljaju zbir nekih drugih atributa, kako bi se smanjila dimenzionalnost podataka bez

gubitka neke informacije. Atributi koji su ostali predstavljaju skup atributa X koji će biti korišćen, dok će kolona “WINNER” biti ciljna promenljiva (Slika 3).

```
In [13]: # Replace NaN values with 0 in 'OTH' and 'OTA'
         df['OTH'].fillna(0, inplace=True)
         df['OTA'].fillna(0, inplace=True)

         # Add 'OTH' to 'Q4H' and 'OTA' to 'Q4A'
         df['Q4H'] = (df['Q4H'] + df['OTH']).astype(int)
         df['Q4A'] = (df['Q4A'] + df['OTA']).astype(int)

         # Add 'OTH' to 'Q4H' and 'OTA' to 'Q4A'
         df['P2H'] = (df['P2H'] + df['OTH']).astype(int)
         df['P2A'] = (df['P2A'] + df['OTA']).astype(int)

In [14]: # One-hot encode 'HT' and 'AT'
         df = pd.get_dummies(df, columns=['HT', 'AT'], drop_first=True)

In [15]: # Drop columns that are sums of other columns, overtime columns as they are now calculated in the 4th quarter column
         # and the target variable column

         X = df.drop(columns=['WINNER', 'OTH', 'OTA', 'Q1T', 'Q2T', 'Q3T', 'Q4T', 'P1T', 'P2T', 'HTT', 'FTT', 'GAPT'])

         # 'WINNER' column is the target variable
         Y = df['WINNER']
```

Slika 3 – Pretprocesiranje, drugi deo

Sledeći korak je podela celog skupa podataka na trening i test skup u odnosu 70:30. Skupovi će biti stratifikovani. Takođe će biti izvršena standardizacija na svim numeričkim atributima, kako bi modeli mašinskog učenja bolje funkcionisali (Slika 4).

```
In [18]: # Split the data set on training and test sets
         x_train, x_test, y_train, y_test = model_selection.train_test_split(X, Y, train_size=0.7, stratify = Y, random_state = 42)

In [19]: # Create standardization scaler
         scaler = StandardScaler()

In [20]: # List all the numerical_columns
         numerical_columns = ['HS', 'AS', 'Q1H', 'Q1A', 'Q2H', 'Q2A', 'Q3H', 'Q3A', 'Q4H', 'Q4A', 'P1H', 'P1A', 'P2H', 'P2A', 'GAPA1C

In [21]: # Fit the scaler on training set
         scaler.fit(x_train[numerical_columns])

Out[21]: ▾ StandardScaler
         StandardScaler()

In [22]: # Transform training and test set with the standardization scaler
         x_train[numerical_columns] = scaler.transform(x_train[numerical_columns])
         x_test[numerical_columns] = scaler.transform(x_test[numerical_columns])
```

Slika 4 – Podela na trening i test skup i standardizacija

3. KLASIFIKACIJA

Ciljna promenljiva koja se traži je kategorička, tako da je prirodno da se primeni klasifikacija kako bi se rešio ovaj problem. Kako bi se odredio najbolji metod za klasifikovanje datih podataka, biće upoređeni rezultati dobijeni primenom četiri različita metoda za klasifikovanje podataka:

1. K – najbližih suseda
2. Metod potpornih vektora
3. AdaBoost ansambl stabala odlucivanja
4. Potpuno povezane neuronske mreže

3.1. K-najbližih suseda

Algoritam K-najbližih suseda (*K-nearest neighbors*) koristi određen broj k najbližih tačaka nekoj tački m, kako bi odredio kojoj klasi pripada tačka m.

Kako bi se odredili najbolji parametri za kreiranje modela na datim podacima, koristiće se unakrsna validacija (Slika 5).

```
In [23]: # Parametes used in the GridSearchCV for the KNN
         parameters = {
             "n_neighbors":[2,6,10],
             "p":[1,2],
             "weights":["uniform", "distance"]
         }

In [24]: # Create KNN model
         clf_knn = model_selection.GridSearchCV (KNeighborsClassifier(), parameters)

In [25]: # Train KNN model
         clf_knn.fit(x_train, y_train)

Out[25]: GridSearchCV
         estimator: KNeighborsClassifier
             KNeighborsClassifier

In [26]: # Show parameters that are chosen as the best by GridSearchCV for KNN
         clf_knn.best_params_

Out[26]: {'n_neighbors': 10, 'p': 1, 'weights': 'distance'}
```

Slika 5 – Unakrsna validacija i izabrani parametri metoda K-najbližih suseda

Kao što se vidi na Slici 5, broj najbližih suseda je 10, Menhetn rastojanje se koristi za merenje distance između tačaka i bliži susedi će imati proporcionalno veću težinu.

Pomoću izabranih parametara se kreira model koji se uči na trening skupu i zatim testira na trening (Slika 6) i test skupu (Slika 7).

```
In [27]: # Predict results on the training set for KNN
         y_pred = clf_knn.predict(x_train)

In [28]: print(classification_report(y_train, y_pred))
```

	precision	recall	f1-score	support
Alba Berlin	1.00	1.00	1.00	3
Anadolu Efes	1.00	1.00	1.00	47
Bamberg	1.00	1.00	1.00	20
Barcelona	1.00	1.00	1.00	44
Baskonia	1.00	1.00	1.00	47
Bayern	1.00	1.00	1.00	13
Buducnost	1.00	1.00	1.00	4
CSKA Moscow	1.00	1.00	1.00	73
Cedevita	1.00	1.00	1.00	3
Crvena Zvezda	1.00	1.00	1.00	27
Darussafaka Ct	1.00	1.00	1.00	19
Fenerbahce	1.00	1.00	1.00	67
Galatasaray CC	1.00	1.00	1.00	8
Gran Canaria	1.00	1.00	1.00	6
Khimki	1.00	1.00	1.00	27
Lokomotiv Kuban	1.00	1.00	1.00	8
Lyon-Villeurbanne	1.00	1.00	1.00	4
Maccabi Tel Aviv	1.00	1.00	1.00	32
Olimpia Milano	1.00	1.00	1.00	27
Olympiacos	1.00	1.00	1.00	48
Panathinaikos	1.00	1.00	1.00	50
Real Madrid	1.00	1.00	1.00	64
Unicaja	1.00	1.00	1.00	11
Unics Kazan	1.00	1.00	1.00	6
Valencia	1.00	1.00	1.00	12
Zalgiris Kaunas	1.00	1.00	1.00	40
Zenit Petersburg	1.00	1.00	1.00	2
accuracy			1.00	712
macro avg	1.00	1.00	1.00	712
weighted avg	1.00	1.00	1.00	712

Slika 6 – Testiranje modela na trening skupu, dobijeni rezultati primenom metoda K-najbližih suseda

```
In [29]: # Predict results on the test set for KNN
         y_pred_test = clf_knn.predict(x_test)

In [30]: print(classification_report(y_test, y_pred_test))
```

	precision	recall	f1-score	support
Alba Berlin	0.00	0.00	0.00	1
Anadolu Efes	0.40	0.40	0.40	20
Bamberg	0.00	0.00	0.00	8
Barcelona	0.27	0.21	0.24	19
Baskonia	0.19	0.20	0.20	20
Bayern	0.00	0.00	0.00	6
Buducnost	0.00	0.00	0.00	2
CSKA Moscow	0.40	0.53	0.45	32
Cedevita	0.00	0.00	0.00	1
Crvena Zvezda	0.33	0.25	0.29	12
Darussafaka Ct	0.17	0.12	0.14	8
Fenerbahce	0.27	0.46	0.34	28
Galatasaray CC	0.00	0.00	0.00	3
Gran Canaria	0.00	0.00	0.00	2
Khimki	0.33	0.17	0.22	12
Lokomotiv Kuban	0.00	0.00	0.00	4
Lyon-Villeurbanne	0.00	0.00	0.00	2
Maccabi Tel Aviv	0.57	0.29	0.38	14
Olimpia Milano	0.38	0.25	0.30	12
Olympiacos	0.39	0.45	0.42	20
Panathinaikos	0.36	0.45	0.40	22
Real Madrid	0.45	0.61	0.52	28
Unicaja	0.00	0.00	0.00	5
Unics Kazan	0.00	0.00	0.00	2
Valencia	0.00	0.00	0.00	5
Zalgiris Kaunas	0.33	0.53	0.41	17
Zenit Petersburg	0.00	0.00	0.00	1
accuracy			0.34	306
macro avg	0.18	0.18	0.17	306
weighted avg	0.30	0.34	0.31	306

Slika 7 - – Testiranje modela na test skupu, dobijeni rezultati primenom metoda K-najbližih suseda

3.2. Metod potpornih vektora

Metod potpornih vektora (*Support Vector Machine*) se koristi za numeričke podatke i najčešće binarnu klasifikaciju. Ideja je u skupu podataka pronaći hiper-ravan koja što bolje razdvaja podatke u pripadajuće klase.

Kao i do sada, biće korišćena unakrsna validacija kako bi se izabrali najoptimalniji parametri za kreiranje modela (Slika 8). Treba napomenuti da iako je u obzir uzet relativno mali raspon parametara, pravljenje modela je trajalo značajno duže nego kod druge dve korišćene metode.

```
In [31]: # Parametes used in the GridSearchCV for the SVM
         parameters_svm = {
             'kernel':["linear","poly","rbf","sigmoid"],
             'gamma':["scale","auto"],
             'C':[0.5,1,1.5]
         }

In [32]: # Create SVM model
         clf_svm = model_selection.GridSearchCV(SVC(), parameters_svm)

In [33]: # Train SVM model
         clf_svm.fit(x_train, y_train)

Out[33]: GridSearchCV
         estimator: SVC
             SVC

In [34]: # Show best parameters for SVM chosen by GridSearchCV
         print(clf_svm.best_params_)

         {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
```

Slika 8 – Unakrsna validacija i izabrani parametri kod Metoda potpornih vektora

Za kraj je ostalo samo da se model istrenira na trening skupu i testira na trening (Slika 9) i test skupu (Slika 10).

```
In [35]: # Predict results on the training set for SVM
         y_pred = clf_svm.predict(x_train)

In [36]: print(classification_report(y_train, y_pred))
```

	precision	recall	f1-score	support
Alba Berlin	1.00	1.00	1.00	3
Anadolu Efes	0.95	0.89	0.92	47
Bamberg	1.00	1.00	1.00	20
Barcelona	0.98	0.98	0.98	44
Baskonia	1.00	0.91	0.96	47
Bayern	1.00	1.00	1.00	13
Buducnost	1.00	1.00	1.00	4
CSKA Moscow	0.94	1.00	0.97	73
Cedevita	1.00	1.00	1.00	3
Crvena Zvezda	1.00	0.96	0.98	27
Darussafaka Ct	1.00	0.95	0.97	19
Fenerbahce	0.92	0.97	0.94	67
Galatasaray CC	1.00	1.00	1.00	8
Gran Canaria	1.00	1.00	1.00	6
Khimki	1.00	1.00	1.00	27
Lokomotiv Kuban	1.00	1.00	1.00	8
Lyon-Villeurbanne	1.00	1.00	1.00	4
Maccabi Tel Aviv	0.94	0.91	0.92	32
Olimpia Milano	1.00	1.00	1.00	27
Olympiacos	0.94	0.94	0.94	48
Panathinaikos	0.96	0.96	0.96	50
Real Madrid	0.95	0.97	0.96	64
Unicaja	1.00	1.00	1.00	11
Unics Kazan	1.00	1.00	1.00	6
Valencia	1.00	1.00	1.00	12
Zalgiris Kaunas	0.93	0.93	0.93	40
Zenit Petersburg	1.00	1.00	1.00	2
accuracy			0.96	712
macro avg	0.98	0.98	0.98	712
weighted avg	0.96	0.96	0.96	712

Slika 9 – Testiranje modela na trening skupu, dobijeni rezultati primenom metoda Potpornih vektora

```
In [37]: # Predict results on the test set for SVM
         y_pred_test = clf_svm.predict(x_test)

In [38]: print(classification_report(y_test, y_pred_test))
```

	precision	recall	f1-score	support
Alba Berlin	0.00	0.00	0.00	1
Anadolu Efes	0.62	0.65	0.63	20
Bamberg	0.60	0.38	0.46	8
Barcelona	0.50	0.68	0.58	19
Baskonia	0.69	0.55	0.61	20
Bayern	0.60	0.50	0.55	6
Buducnost	0.00	0.00	0.00	2
CSKA Moscow	0.73	0.75	0.74	32
Cedevita	0.00	0.00	0.00	1
Crvena Zvezda	0.75	0.50	0.60	12
Darussafaka Ct	0.50	0.75	0.60	8
Fenerbahce	0.78	0.89	0.83	28
Galatasaray CC	1.00	0.33	0.50	3
Gran Canaria	0.50	0.50	0.50	2
Khimki	0.62	0.67	0.64	12
Lokomotiv Kuban	1.00	0.25	0.40	4
Lyon-Villeurbanne	1.00	0.50	0.67	2
Maccabi Tel Aviv	0.64	0.50	0.56	14
Olimpia Milano	0.83	0.42	0.56	12
Olympiacos	0.65	0.85	0.74	20
Panathinaikos	0.62	0.68	0.65	22
Real Madrid	0.63	0.61	0.62	28
Unicaja	0.75	0.60	0.67	5
Unics Kazan	0.00	0.00	0.00	2
Valencia	0.50	0.40	0.44	5
Zalgiris Kaunas	0.83	0.59	0.69	17
Zenit Petersburg	0.33	1.00	0.50	1
accuracy			0.63	306
macro avg	0.58	0.50	0.51	306
weighted avg	0.67	0.63	0.63	306

Slika 10 – Testiranje modela na test skupu, dobijeni rezultati primenom metoda K-najbližih suseda

3.3. AdaBoost ansambl stabala odlucivanja

Stabla odlučivanja (Decision trees) je metod nadgledanog učenja pri kome se skup podataka grana (deli) dok se ne dobiju dovoljno čiste grupe slogova. Cilj je da model predvidi izlaznu vrednost (klasu) na osnovu ulaznih parametara nekoliko atributa. Ansambl je skup više stabala odlučivanja koja su obučena na nezavisnim podskupovima. AdaBoost je algoritam pojačavanja koji će biti primenjen na ovom ansamblu.

Prvo će biti kreiran model AdaBoost ansambl stabala odlucivanja (Slika 11).

```
In [39]: # Create RandomForestClassifier as the base estimator
         base_estimator = ensemble.RandomForestClassifier(n_estimators=100, random_state=42)

In [40]: # Create an AdaBoostClassifier with Random Forest as the base estimator
         adaboost_classifier = ensemble.AdaBoostClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)

In [41]: # Train AdaBoostClassifier
         adaboost_classifier.fit(x_train, y_train)

Out[41]: AdaBoostClassifier
         base_estimator: RandomForestClassifier
                 RandomForestClassifier
```

Slika 11 – Kreiranje i obučavanje modela AdaBoost ansambl stabala odlucivanja

Zatim će model biti testiran na trening (Slika 12) i test skupu (Slika 33).

```
In [42]: # Predict results on the training set
         y_pred = adaboost_classifier.predict(x_train)

In [43]: print(classification_report(y_train, y_pred))
```

	precision	recall	f1-score	support
Alba Berlin	1.00	1.00	1.00	3
Anadolu Efes	1.00	1.00	1.00	47
Bamberg	1.00	1.00	1.00	20
Barcelona	1.00	1.00	1.00	44
Baskonia	1.00	1.00	1.00	47
Bayern	1.00	1.00	1.00	13
Buducnost	1.00	1.00	1.00	4
CSKA Moscow	1.00	1.00	1.00	73
Cedevita	1.00	1.00	1.00	3
Crvena Zvezda	1.00	1.00	1.00	27
Darussafaka Ct	1.00	1.00	1.00	19
Fenerbahce	1.00	1.00	1.00	67
Galatasaray CC	1.00	1.00	1.00	8
Gran Canaria	1.00	1.00	1.00	6
Khimki	1.00	1.00	1.00	27
Lokomotiv Kuban	1.00	1.00	1.00	8
Lyon-Villeurbanne	1.00	1.00	1.00	4
Maccabi Tel Aviv	1.00	1.00	1.00	32
Olimpia Milano	1.00	1.00	1.00	27
Olympiacos	1.00	1.00	1.00	48
Panathinaikos	1.00	1.00	1.00	50
Real Madrid	1.00	1.00	1.00	64
Unicaja	1.00	1.00	1.00	11
Unics Kazan	1.00	1.00	1.00	6
Valencia	1.00	1.00	1.00	12
Zalgiris Kaunas	1.00	1.00	1.00	40
Zenit Petersburg	1.00	1.00	1.00	2
accuracy			1.00	712
macro avg	1.00	1.00	1.00	712
weighted avg	1.00	1.00	1.00	712

Slika 12 - Testiranje modela na trening skupu, dobijeni rezultati primenom metoda AdaBoost ansambl stabala

```
In [44]: # Predict results on the test set
         y_pred_test = adaboost_classifier.predict(x_test)

In [45]: print(classification_report(y_test, y_pred_test))
```

	precision	recall	f1-score	support
Alba Berlin	0.00	0.00	0.00	1
Anadolu Efes	0.73	0.80	0.76	20
Bamberg	0.86	0.75	0.80	8
Barcelona	0.65	0.68	0.67	19
Baskonia	0.78	0.70	0.74	20
Bayern	0.50	0.67	0.57	6
Buducnost	0.00	0.00	0.00	2
CSKA Moscow	0.89	1.00	0.94	32
Cedevita	0.00	0.00	0.00	1
Crvena Zvezda	0.64	0.75	0.69	12
Darussafaka Ct	0.33	0.38	0.35	8
Fenerbahce	0.80	0.86	0.83	28
Galatasaray CC	1.00	0.67	0.80	3
Gran Canaria	0.00	0.00	0.00	2
Khimki	0.70	0.58	0.64	12
Lokomotiv Kuban	1.00	0.50	0.67	4
Lyon-Villeurbanne	1.00	0.50	0.67	2
Maccabi Tel Aviv	0.85	0.79	0.81	14
Olimpia Milano	0.67	0.67	0.67	12
Olympiacos	0.71	0.75	0.73	20
Panathinaikos	0.76	0.86	0.81	22
Real Madrid	0.77	0.86	0.81	28
Unicaja	0.50	0.20	0.29	5
Unics Kazan	1.00	0.50	0.67	2
Valencia	0.80	0.80	0.80	5
Zalgiris Kaunas	0.69	0.65	0.67	17
Zenit Petersburg	0.00	0.00	0.00	1
accuracy			0.74	306
macro avg	0.62	0.55	0.57	306
weighted avg	0.73	0.74	0.73	306

Slika 13 - Testiranje modela na test skupu, dobijeni rezultati primenom metoda AdaBoost ansambl stabala odlucivanja

3.4. Potpuno povezane neuronske mreže

Potpuno povezane neuronske mreže su tip neuronskih mreža kod kojih je svaki neuron povezan sa svim neuronima iz prethodnog i narednog sloja u strukturi mreže. Pretprocesiranje za potpuno povezane neuronske mreže će biti isto sem što će i za ciljnu promenljivu morati da bude primenjeno “One-hot enkodiranje” zato što neuronska mreža mora da ima numeričku vrednost kao ciljnu promenljivu (Slika 14).

```
In [16]: # One-hot encode Y
         Y = pd.get_dummies(Y, drop_first=True)
```

Slika 14 – One-hot enkodiranje ciljne promenljive

Mreža će imati ulazni sloj imati X (izabrane atribute), kao izlazni sloj će imati 26 neurona jer postoji 26 različitih košarkaških klubova u skupu podataka, dok će između njih imati četiri gusta sloja sa po 200 neurona, za koje je aktivaciona funkcija “relu” (Slika 15). Eksperimentisano je sa različitim strukturama mreže, ali dodavanje novih slojeva ili novih neurona nije davalo bolje rezultate.

```
In [22]: # Determine the size of input and output network layers
         number_of_features = X.shape[1]
         output_size = 26

In [23]: # Create the network structure
         model = Sequential([Input(shape=(number_of_features, )),
                             Dense(units=200, activation='relu'),
                             Dense(units=200, activation='relu'),
                             Dense(units=200, activation='relu'),
                             Dense(units=200, activation='relu'),
                             Dense(units=output_size, activation='sigmoid')
                            ])

In [24]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 200)               14200

 dense_1 (Dense)             (None, 200)               40200

 dense_2 (Dense)             (None, 200)               40200

 dense_3 (Dense)             (None, 200)               40200

 dense_4 (Dense)             (None, 26)                5226

=================================================================
Total params: 140026 (546.98 KB)
Trainable params: 140026 (546.98 KB)
Non-trainable params: 0 (0.00 Byte)
```

Slika 15 – Kreiranje strukture neuronske mreže

Zatim se obučava neuronska mreža, kao funkcija gubitka je izabrana binarna unakrsna entropija, a kao funkcija optimizacije je izabrana Adam funkcija (Slika 16). Mreža se obučava 100 epoha.

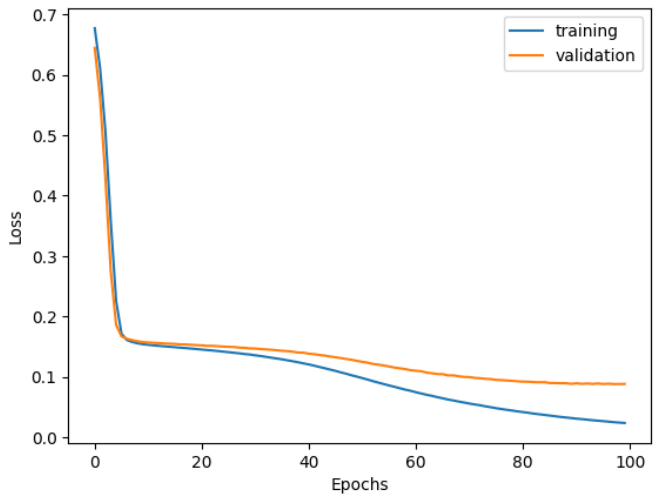
```
In [25]: from tensorflow.keras.optimizers import Adam
         from tensorflow.keras.losses import BinaryCrossentropy

In [26]: # Create the model using Adam optimizer and BinaryCrossentropy loss function
         model.compile(loss=BinaryCrossentropy(), optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])

In [27]: # Train the model for 50 epochs
         history = model.fit(x_train, y_train, epochs=100, batch_size=32, validation_split=0.2, verbose=1)
```

```
0.6503
Epoch 93/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0291 - accuracy: 0.9279 - val_loss: 0.0890 - val_accuracy:
0.6573
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0282 - accuracy: 0.9367 - val_loss: 0.0883 - val_accuracy:
0.6573
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0274 - accuracy: 0.9402 - val_loss: 0.0890 - val_accuracy:
0.6503
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0267 - accuracy: 0.9402 - val_loss: 0.0882 - val_accuracy:
0.6643
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0258 - accuracy: 0.9385 - val_loss: 0.0886 - val_accuracy:
0.6573
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.0250 - accuracy: 0.9455 - val_loss: 0.0881 - val_accuracy:
0.6434
Epoch 99/100
```

Slika 16 – Obučavanje neuronske mreže

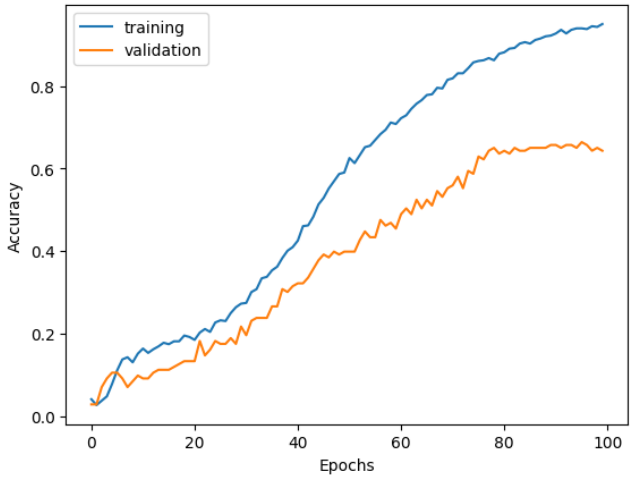
Neuronska mreža se evaluira na osnovu funkcije gubitka (Slika 17) i preciznosti (Slika 18) na trening i validacionom skupu.

```
In [28]: epochs = history.epoch
         loss = history.history['loss']
         validation_loss = history.history['val_loss']

In [29]: # Showing the change of the loss function over 50 epochs
         plt.plot(epochs, loss, label='training')
         plt.plot(epochs, validation_loss, label='validation')
         plt.xlabel('Epochs')
         plt.ylabel('Loss')
         plt.legend(loc='best')
         plt.show()
```



Slika 17 – Evaluacija funkcije greške

```
In [30]: acc = history.history['accuracy']
         validation_acc = history.history['val_accuracy']

In [31]: # Showing the change of the accuracy over 50 epochs
         plt.plot(epochs, acc, label='training')
         plt.plot(epochs, validation_acc, label='validation')
         plt.xlabel('Epochs')
         plt.ylabel('Accuracy')
         plt.legend(loc='best')
         plt.show()
```



Slika 18 – Evaluacija preciznosti

Na osnovu evaluacije, zaključuje se da je model dovoljno dobar i on će se koristiti kao finalni model za ovaj skup podataka. Ono što se može videti iz rezultata na trening i test skupu (Slika 19) je da je model preprilagođen podacima, ali i pored pokušaja da se model regularizuje primenom "Dropout" slojeva i l2 regularizacije, dobijeni su samo gori rezultati i na trening i na test skupu.

```
In [32]: # Final model structure after evaluation
         final_model = Sequential([Input(shape=(number_of_features, )),
                                   Dense(units=200, activation='relu'),
                                   Dense(units=200, activation='relu'),
                                   Dense(units=200, activation='relu'),
                                   Dense(units=200, activation='relu'),
                                   Dense(units=output_size, activation='sigmoid')
                                  ])

In [33]: # Creating final model
         final_model.compile(loss=BinaryCrossentropy(), optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])

In [34]: # Training final model
         history = final_model.fit(x_train, y_train, epochs=100, batch_size=32, verbose=0)

In [35]: # Final model evaluation on training set
         train_scores = final_model.evaluate(x_train, y_train, batch_size=32)

         23/23 [==============================] - 0s 706us/step - loss: 0.0162 - accuracy: 0.9789

In [36]: # Final model evaluation on test set
         test_scores = final_model.evaluate(x_test, y_test, batch_size=32)

         10/10 [==============================] - 0s 778us/step - loss: 0.0776 - accuracy: 0.7190
```

Slika 19 – Rezultati primene modela neuronske mreže na trening i test skupu

4. ZAKLJUČAK

Kada se uporede dobijeni rezultati korišćenjem četiri metoda na datom skupu podataka (Tabela 1), može se zaključiti da je kod svih metoda došlo do preprilagođavanja podacima skupa. To je posledica samog skupa podataka, koji nije dovoljno veliki, a pritom ima 26 različitih klubova kao klase ciljne promenljive. Kod svih metoda je pokušan neki vid regularizacije kako bi smanji preprilagođavanje, ali to je samo rezultiralo lošijim rezultatima modela na test skupu. Rešenje bi bilo ili da se obogati sam skup novim podacima ili nekim boljim atributima ili da se smanji broj klubova koji predstavljaju klase ciljne promenljive. Uzevši u obzir trenutno stanje, najbolje se pokazao Adaboost ansambl stabala odlučivanja kao metod klasifikacije za ovaj zadatak, jer ima najbolje rezultate, kao i manje vreme izvršavanja od Metoda potpornih vektora i Potpuno povezane neuronske mreže.

	Skup	Preciznost
K-najbližih suseda	Trening	1.00
	Test	0.34
Metod potpornih vektora	Trening	0.96
	Test	0.63
Adaboost ansambl stabala odlučivanja	Trening	1.00
	Test	0.75
Potpuno povezana neuronska mreža	Trening	0.98
	Test	0.72

Tabela 1 – Poređenje rezultata dobijenih korišćenjem različitih metoda za klasifikaciju

5. KORIŠĆENI PAKETI

Za izradu ovog projekta korišćeni su sledeći paketi Python-a, koje je potrebno instalirati na raučunar:

1. pandas
2. NumPy
3. scikit-learn
4. Matplotlib
5. TensorFlow

6. LITERATURA

[1] M. Nikolić, Nevena Ćirić, Izvodi sa predavanja i vežbi na predmetu Mašinsko učenje, Univerzitet u Beogradu, Mašinski fakultet, 2023.

[2] M. Nikolić, A. Zečević, Skripta za predmet Mašinsko učenje, Univerzitet u Beogradu, Matematički fakultet, 2019.

[3] https://www.kaggle.com/