Julita Ołtusek
Łukasz Pik

Klasyfikacja piłkarzy pod względem skuteczności na zbiorze danych FIFA 2019 Complete Player Dataset (Kaggle)

Dokumentacja końcowa projektu

1. Interpretacja tematu projektu

Zagadnieniem przeprowadzanej klasyfikacji jest predykcja skuteczności piłkarzy za pomocą wybranych metod uczenia maszynowego pod nadzorem. Główny cel projektu to budowa modeli klasyfikacyjnych oraz ich analiza na podstawie przeprowadzonych testów, z wykorzystaniem szeregu algorytmów, których implementacje są dostępne w pakietach języka R. Przeprowadzona analiza posłuży do oceny jakości klasyfikacji przez wytrenowane modele oraz ich porównanie. Źródłem danych użytym w projekcie jest baza danych piłkarzy FIFA 2019.

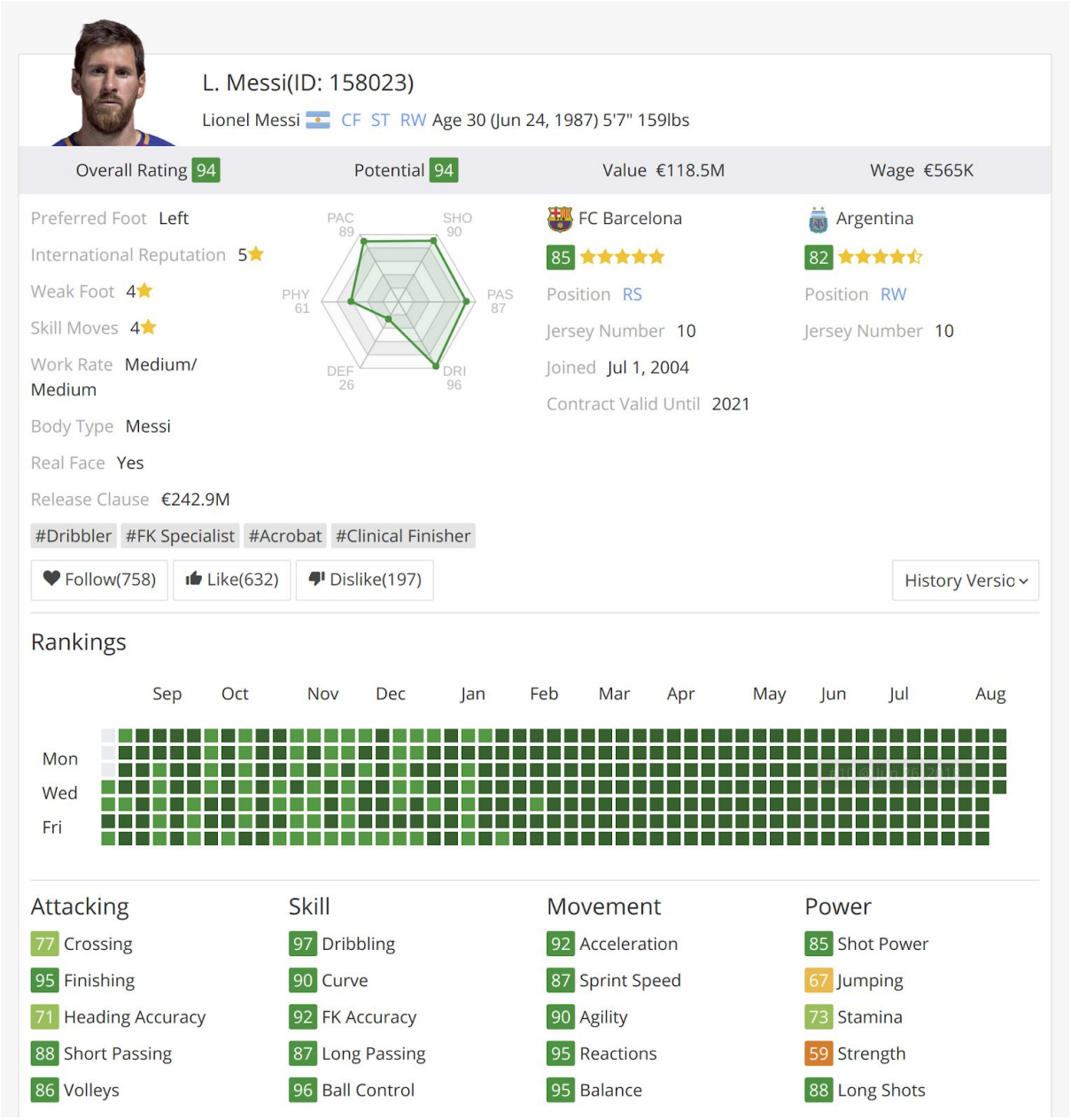
2. Wybór atrybutu reprezentującego pojęcie docelowe

Zbiór danych postanowiliśmy podzielić na klasy względem atrybutu ogólnej oceny (overall). Atrybut ten przyjmuje wartości całkowite 0-100. Rozważaliśmy podział tego zakresu na 5 przedziałów w taki sposób, aby zbiór danych był zbalansowany, a więc dobierając takie wartości granic przedziałów, aby liczba elementów należących do każdej klasy była zbliżona. Jednak okazało się, że modele przy tak postawionym problemie nie osiągają zbyt dobrej jakości. Dlatego zdecydowaliśmy się zrezygnować z takiego sposobu podziału i wybrać inną opcję, którą mieliśmy zaplanowaną w przypadku, kiedy pierwsza się nie uda.

Wyróżniliśmy zatem następujące klasy: 20% najlepszych piłkarzy oraz pozostałe 80% gorszych piłkarzy. Wartością graniczną jest 80ty centyl, który dla atrybutu “Overall” wyniósł 72. Wybranym klasom nadaliśmy etykiety “good” i “bad”. Na podstawie pozostałych cech, odpowiednio przygotowanych, przeprowadzona została klasyfikacja piłkarzy pod względem skuteczności.

3. Opis danych

Zbiór danych wykorzystany w tym zadaniu pochodzi z FIFA 19 complete player dataset. Dane zostały przygotowane w oparciu o serwis https://sofifa.com/, z którego za pomocą skryptu indeksującego, zostały wydobyte dane o piłkarzach znajdujących się w grze FIFA 2019. Do cech należą między innymi: wzrost, wiek, statystyki ofensywne jak i defensywne, czy też refleks. Zbiór danych zawiera 18 207 rekordów, z których każdy jest opisany za pomocą 89 atrybutów. Są to zarówno atrybuty numeryczne (np. drybling, kondycja, wytrzymałość), jak i dyskretne (np. preferowana noga, pozycja). Na poniższej grafice widoczny jest widok szczegółowy dla jednego piłkarza, czyli pojedynczej próbki z naszego zbioru danych.



W zbiorze danych znajduje się część danych tekstowych, jednak są to dane typu “Nazwisko” lub link do zdjęcia, które mają bardzo dużo różnych wartości bądź też dane binarne, jak “Preferowana noga”. Dlatego do klasyfikacji zostały wykorzystane tylko cechy numeryczne. Dodatkowo wszystkie te wartości są mierzone na skali co najmniej porządkowej, dlatego dla algorytmów drzewiastych można na nich zastosować podziały typu “mniejsze”.
W tabeli 1. przedstawiliśmy opis statystyczny danych numerycznych.

	Min.	X1st.Qu.	Median	Mean	X3rd.Qu.	Max.
Age	16	21	25	25.12112	28	45
Potential	48	67	71	71.32408	75	95
Special	731	1457	1635	1598.003	1787	2346
International.Reputation	1	1	1	1.113297	1	5
Skill.Moves	1	2	2	2.361492	3	5
Crossing	5	38	54	49.73841	64	93
Finishing	2	30	49	45.55023	62	95
HeadingAccuracy	4	44	56	52.30077	64	94
ShortPassing	7	54	62	58.69543	68	93
Volleys	4	30	44	42.91222	57	90
Dribbling	4	49	61	55.37516	68	97
Curve	6	34	48	47.17628	62	94
FKAccuracy	3	31	41	42.86604	57	94
LongPassing	9	43	56	52.72139	64	93
BallControl	5	54	63	58.37483	69	96
Acceleration	12	57	67	64.61283	75	97
SprintSpeed	12	57	67	64.72624	75	96
Agility	14	55	66	63.50129	74	96
Reactions	21	56	62	61.83915	68	96
Balance	16	56	66	63.96429	74	96
ShotPower	2	45	59	55.4652	68	95
Jumping	15	58	66	65.09103	73	95
Stamina	12	56	66	63.22158	74	96
Strength	17	58	67	65.31862	74	97
LongShots	3	33	51	47.11319	62	94
Aggression	11	44	59	55.87607	69	95
Interceptions	3	26	52	46.70276	64	92
Positioning	2	38	55	49.9622	64	95
Vision	10	44	55	53.40778	64	94
Penalties	5	39	49	48.54637	60	92
Composure	3	51	60	58.65113	67	96
Marking	3	30	53	47.28605	64	94
StandingTackle	2	27	55	47.70188	66	93
SlidingTackle	3	24	52	45.66634	64	91
GKDiving	1	8	11	16.61691	14	90
GKHandling	1	8	11	16.39384	14	92
GKKicking	1	8	11	16.23304	14	91
GKPositioning	1	8	11	16.38965	14	90
GKReflexes	1	8	11	16.71202	14	94
Value	0	300k	675k	2417729	2000k	118500k
Wage	0	1k	3k	9759	9k	565k
Release.Clause	13000	525k	1100k	4585061	3500k	228100k

Tabela 1. Statystyczny opis danych numerycznych.

4. Przygotowanie danych wejściowych

Zadany zbiór danych został przygotowany w postaci pliku csv, co umożliwia jego prostą konwersję do postaci tabelarycznej. Był już częściowo przetworzony. Usunęliśmy z niego brakujące wartości (zbiór danych i tak jest bardzo duży, więc nie musieliśmy oszczędzać na danych). W rezultacie zostało usuniętych 60 wierszy, co daje 18147 pozostałych próbek. Część atrybutów numerycznych została zapisana w postaci tekstowej, z jednostkami (np. cechy będące kwotami, takie jak szacunkowa wartość piłkarza, pensja zostały podane z symbolem waluty oraz skrótami oznaczającymi krotność (K, M), waga została podana w funtach ze

skrótem “lbs” na końcu. Zostały przez nas przetransformowane do wartości liczbowych, aby wydobyć z nich relację porządku. Niektóre atrybuty, takie jak logo klubu czy długość kontraktu, od razu zauważyliśmy, że nie są przydatne w naszym zadaniu klasyfikacji i zostały usunięte ze wstępnego zbioru danych. Większość cech liczbowych jest mierzonych w skali 0-100. Dane nie zostały poddane standaryzacji.

5. Selekcja atrybutów

Z uwagi na dużą liczbę atrybutów, jaki dany zbiór posiada, postanowiliśmy, że tylko części z nich użyliśmy do klasyfikacji. Dzięki temu ograniczyliśmy złożoność obliczeniową, a także pozbyliśmy się atrybutów, które nie są skorelowane z klasami bądź są z nimi skorelowane zbyt silnie i za bardzo by to strywializowało nasz problem. Aby zmaksymalizować jakość predykcji należy pozbyć się informacji zbędnych lub powtarzających się. Na początku odrzuciliśmy cechy wspomniane wcześniej, które od razu były podejrzane o to, że nie pomogą w klasyfikacji. Następnie, chcieliśmy dowiedzieć się, jak wiele informacji o klasie niesie ze sobą dana cecha. Przy odrzuceniu części atrybutów można posłużyć się różnymi technikami wyboru cech. Mogą one korzystać z entropii, Indeksu Giniego lub w inny sposób obliczać korelację zmiennych. W naszym przypadku zbadaliśmy korelację cech z atrybutem, od którego pochodzi pojęcie docelowe. Wykorzystaliśmy również funkcję *varimp* z pakietu *caret* środowiska R, która korzysta z różnych metod w zależności od algorytmu klasyfikacji, do jakiego jest zastosowana. Odrzuciliśmy cechy, które miały wyższy od 0.5 wsp. korelacji z atrybutem Overall, co zgadzało się także z wynikiem działania funkcji sprawdzającej ważność cech pod względem indeksu Giniego dla lasu losowego o domyślnych parametrach.

BallControl	Dribbling	ShortPassing	StandingTackle
1593.7	763.9927	680.8603	626.7392
Vision	Composure	Skill.Moves	GKDiving
543.3198	451.5075	434.5817	306.0741
SlidingTackle	Marking	Interceptions	GKReflexes
289.2099	258.0582	239.3012	201.3994
GKPositioning	GKHandling	Finishing	Positioning
185.8621	146.0512	141.9108	101.7919
GKKicking	HeadingAccuracy	Position	Reputation
54.32484	52.32194	46.32916	24.55073
Volleys	LongPassing	Aggression	Penalties
22.30532	13.09811	9.152517	8.221935
ShotPower	Curve	LongShots	FKAccuracy
7.738292	6.115002	4.296537	3.989642

Acceleration	Crossing	Jumping	Work.Rate
2.263323	2.263323	2.240093	2.240093

Stamina
1.120047

Tabela 2. Znaczenie zmiennych wyrażone za pomocą Mean Gini index

6. Algorytmy klasyfikacji

6.1. Naiwny klasyfikator Bayesa

Jest to model cech niezależnych, który korzystając z twierdzenia Bayesa wyprowadza model prawdopodobieństwa. Ta metoda bardzo dobrze sprawuje się w przypadkach dużej liczby cech obiektów oraz jest stosunkowo łatwa do wykorzystania, a także bardzo skutecznie klasyfikuje, o ile założenie o wzajemnej niezależności atrybutów jest spełnione w znacznym stopniu. W projekcie zostanie wykorzystany klasyfikator z pakietu e1071.

6.2. Drzewo decyzyjne

Drzewa służą do analizy oraz klasyfikacji danych, na podstawie serii warunków znajdujących się w węzłach. Klasyfikacja polega na przejściu drzewa od korzenia do liścia, który daje nam jednoznaczną odpowiedź. Proces klasyfikacji z wykorzystaniem drzew decyzyjnych jest efektywny obliczeniowo, wyznaczenie kategorii przykładu wymaga w najgorszym razie przetestowania raz wszystkich jego atrybutów. Wykorzystana zostanie implementacja z pakietu rpart.

Parametry do strojenia:

- minsplit - minimalna liczba obserwacji, jaka musi być wykryta w węźle, aby był on rozważany do analizy podziału;
- minbucket - minimalna liczba obserwacji, jaką musi posiadać każdy liść drzewa;
- cp - parametr odpowiedzialny za obcinanie podziałów małoznaczących. Każdy podział, który nie będzie przypasowania o co najmniej wartość parametru cp, zostanie pominięty;
- maxcompete - liczba wybieranych podziałów jakie drzewo ma posiadać;
- maxsurrogate - maksymalna liczba zastępczych podziałów;
- usesurrogate - parametr mówiący o tym, jak mają wyglądać podziały zastępcze
  dla wartości 0 - jeśli nie ma wartości podziału dla danej zasady, obiekt nie jest dalej analizowany:
  1 - używa zastępczych wartości, aby dokonać podziału na zasadzie, jeśli brakuje wszystkich zastępców, obiekt nie jest analizowany,
  2 - jeśli brakuje wszystkich zastępców, wtedy wykorzystuje ścieżkę większościową;
- xval - liczba grup w walidacji krzyżowej k-krotnej;
- surrogatestyle - parametr kontrolujący wybór najlepszych zastępców;
- maxdepth - maksymalna głębokość finalnego drzewa.

6.3. Las losowy

Las losowy jest zbudowany w wielu klasyfikatorów, którymi są właśnie drzewa decyzyjne. Dane przechodzą przez każde z drzew, a następnie gdy klasyfikacja zostanie przez nie zakończona, wyniki są agregowane i wybierany jest poprzez głosowanie ostateczny wynik klasyfikacji. Do lasu losowego wykorzystana zostanie implementacja z pakietu randomForest.
Parametry do strojenia:

- nodesize - minimalny rozmiar liści (liczba próbek ze zbioru trenującego);
- maxnodes - maksymalna liczba liści;
- ntree - liczba drzew;
- replace - parametr mówiący, czy powinny być używane zamienniki;
- mtry -liczba zmiennych losowo wybranych na kandydatów na każdym podziale. Domyślnie jest to wartość sqrt(p), gdzie p to liczba zmiennych w analizowanych danych;
- cutoff - wektor proporcji głosów, używany do obcinania;
- strata - zmienna używana do losowania warstwowego;
- sampsize - wielkość przykładów do rysowania.

7. Testowanie i ocena jakości modeli

Kiedy model będzie już zbudowany, można będzie przystąpić do oceny jego jakości. Ten etap polega na sprawdzeniu na zbiorze testowym, jaka jest skuteczność predykcji wytrenowanego modelu. Przy ocenie klasyfikatora można wziąć pod uwagę wiele kryteriów. Opisujemy dalej, jakie miary jakości oraz procedury oceny zastosujemy.

7.1. Macierz pomyłek

Przy ocenie jakości modeli bardzo pomocna jest macierz pomyłek nazywana również macierzą błędów (ang. error matrix) lub konfuzji (ang. confusion matrix) . Forma tabeli pozwala na przejrzystą wizualizację wyników testów klasyfikatora. To macierz kwadratowa o wymiarach *nxn*, gdzie *n* to liczba klas. Każdy wiersz reprezentuje klasy faktyczne, natomiast każda kolumna – klasy przewidziane. Zatem w komórce *mij* zapisuje się liczbę przypadków testowych należących do klasy *yi*, zaklasyfikowanych do klasy *yj*. Na głównej przekątnej znajdują się przypadki zaklasyfikowane poprawnie. Macierz jest wykorzystywana do wyznaczania miar jakości klasyfikatora.

cała próba		klasy przewidziane				
		klasa 1	klasa 2	…	klasa n	Σ
klasy faktyczne	klasa 1	m_{11}	m12	…	m_{1n}	$\sum_{j=1}^{n} m_{1j}$
	klasa 2	m_{21}	m_{22}	…	m_{2n}	$\sum_{j=1}^{n} m_{2j}$
	…	…	…	…	…	…
	klasa n	m_{n1}	m_{n2}	…	m_{nn}	$\sum_{j=1}^{n} m_{nj}$
	Σ	$\sum_{i=1}^{n} m_{i1}$	$\sum_{i=1}^{n} m_{i2}$	…	$\sum_{i=1}^{n} m_{in}$	$\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}$

Tabela 3. Przykładowa macierz pomyłek dla klasyfikacji wieloklasowej.

7.2. Miary jakości

W projekcie użyjemy 3 miar jakości klasyfikacji, które są jednymi z najpopularniejszych, a także dodatkowej metody oceny - krzywej ROC. W przypadku, kiedy zdecydujemy się na wariant wieloklasowy, w celu oceny modeli sprowadzimy je do zadania binarnego pod postacią OVA (one vs all - jedna kontra pozostałe). W dalszej części tego punktu użyliśmy następujących oznaczeń:
X - wektor atrybutów
k - numer klasy
K - liczba wszystkich klas
Y - etykieta klasy
d - klasyfikator
m_{ij} - wartość z pola macierzy konfuzji

Dokładność (ang. accuracy)

Dokładność to jedna z popularnych miar ocen klasyfikatorów. Jest oznaczana przez procentową część przykładów testowych poprawnie zaklasyfikowanych do odpowiadających im klas. W taki sposób określa się prawdopodobieństwo, że nowy przykład zostanie poprawnie zaklasyfikowany.

$$accuracy(d) = P(d(X) = Y) = \frac{\sum_i m_{ii}}{\sum_{i,j} m_{ij}}$$

Precyzja (ang. precision)

Precyzja to miara, która określa prawdopodobieństwo tego, że obiekt faktycznie należy do danej klasy, jeśli został do niej zaklasyfikowany. Jest to stosunek liczby obiektów poprawnie zaklasyfikowanych do danej klasy do wszystkich obiektów, które zostały zaklasyfikowane jako należące do niej (łącznie z błędnie zaklasyfikowanymi). Precyzja liczona jest oddzielnie dla każdej klasy.

$$precision_k(d) = P(Y = y_k | d(X) = y_k) = \frac{m_{kk}}{\sum_j m_{jk}}$$

Czułość (ang. recall, sensitivity)

Czułość określa stosunek liczby poprawnie zaklasyfikowanych obiektów z danej klasy do wszystkich obiektów, które rzeczywiście do niej należą. Innymi słowy jest to prawdopodobieństwo poprawnej klasyfikacji do danej klasy pod warunkiem, że obiekt rzeczywiście do niej należy. Czułość podobnie jak precyzja liczona jest oddzielnie dla każdej klasy.

$$recall_k(d) = P(d(X) = y_k | Y = y_k) = \frac{m_{kk}}{\sum_j m_{kj}}$$

Krzywa ROC (ang. Receiver Operating Characteristic curve)

Krzywa ROC to graficzna metoda oceny jakości klasyfikacji. W układzie współrzędnych (FP rate, TP rate) przedstawiana jest wizualizacja punktów pracy modeli klasyfikacji. Pole pod krzywą ROC nazywa się AUC (Area Under the Curve) i jego zakres wartości wynosi [0;1]. Im bliżej 1, tym model lepiej radzi sobie z klasyfikacją. Dla przypadku wieloklasowego sporządzimy charakterystyki dla każdego wariantu OVA bądź jedną wspólną, po uśrednieniu wartości z komórek poszczególnych macierzy pomyłek.

7.3. Procedury oceny jakości

Zbiór testowy jest zwykle częścią populacji danych, pozostałą po oddzieleniu wcześniej zbioru trenującego. Dzięki temu, że trenowanie modelu przebiega na innych danych, niż jego testowanie, można sprawdzić, jak model zachowuje się w przypadku nowych dla niego i zupełnie nieznanych danych. Pomaga to zapobiec przeuczeniu (ang. overfitting), czyli nadmiernemu dopasowaniu danych do zestawu treningowego (czasem też pośrednio testowego). Klasyfikator przewiduje klasy dla przypadków testowych, a ich wartości porównywane są z rzeczywistymi znanymi klasami, po czym wyznaczane są miary jakości opisane w poprzednim punkcie.

7.3.1. Walidacja krzyżowa

Do badania skuteczności klasyfikatorów często wykorzystuje się metodę znaną w statystyce jako walidacja krzyżowa (ang. cross-validation). Procedura w ogólnym przypadku przebiega w taki sposób, że zarówno budowa klasyfikatora, jak i jego testowanie wykonywane są wielokrotnie, za każdym razem przy innym podziale danych na zbiór treningowy i testowy. Następnie wyniki wszystkich iteracji są łączone (np. uśredniane), co pozwala oszacować skuteczność predykcji modelu. Rozróżnia się kilka rodzajów tej metody, jednak my zdecydowaliśmy się wybrać **K-krotną walidację ze zbiorem testującym i weryfikującym**. Liczność przykładów w zbiorze danych wynosi ponad 18 tysięcy, dlatego uznaliśmy, że możemy sobie pozwolić na wydzielenie zbioru weryfikującego, a nawet było to wskazane głównie na dużą zasobożerność algorytmu lasu losowego, który na pełnym zbiorze danych działał bardzo wolno i staraliśmy się ograniczyć jak najbardziej czynniki, które na to wpływały. Dlatego zbiór danych został podzielony na 2 równe części na zbiór trenujący i walidacyjny. Wartość liczby podziałów w walidacji k-krotnej ustaliliśmy jako 10.

7.3.2. Makro- i mikro-uśrednianie

Po przeprowadzeniu szeregu testów przy użyciu walidacji krzyżowej i wyznaczeniu wartości miar jakości dla poszczególnych klas (precyzja i czułość), można je uśrednić, aby uzyskać ogólne rezultaty klasyfikacji. W tym celu wykorzystuje się zwykle jedną z dwóch metod: makro- i mikro- uśredniania.

Makro-uśrednianie
W metodzie makro-uśredniania obliczona średnia miara to po prostu średnia miar dla poszczególnych klas.

$$\text{measure}_{\text{macro}}(d) = \frac{1}{\text{K}}\sum_{\text{k=1}}^{\text{K}} \text{meaure}_{\text{k}}(\text{d})$$

Ważnym aspektem jest tutaj brak wrażliwości na różnice w licznościach zbiorów należących bądź zaklasyfikowanych do różnych klas, gdyż każda klasa otrzymuje jednakową wagę.

Mikro-uśrednianie
W metodzie mikro-uśredniania średnia miara jest liczona w nieco bardziej skomplikowany sposób. Ogólnie mówiąc, w każdym miejscu gdzie występują wartości dla konkretnej klasy, trzeba je zsumować po wszystkich klasach. Wzór poniżej odpowiada zarówno średniej precyzji, jak i czułości, a także dokładności, gdyż wartości w mianowniku, czyli suma próbek należących do wszystkich klas i suma próbek sklasyfikowanych są równe, w obu przypadkach to liczność całego zbioru testowego.

$$precision_{micro}(d) = recall_{micro}(d) = \text{accuracy(d)} = \frac{\sum_{k=1}^{K} m_{kk}}{\sum_{i,j} m_{ij}}$$

W tym przypadku, jeśli zbiór danych nie jest zbalansowany, uśredniona miara uwzględnia bardziej te liczniejsze klasy. Np. dla klasyfikacji binarnej, gdy do jednej klasy należy 10% wszystkich próbek, a do drugiej 90%, może się zdarzyć że uśredniona miara będzie wskazywać wysoką wartość, nawet jeśli większość przykładów reprezentujących mniej liczną klasę będzie klasyfikowana nieprawidłowo.

Z powodu niezbalansowanego zbioru danych (stosunek liczebności w klasach wynosił będzie około 1:4) uznaliśmy, że lepiej będzie posłużyć się makro-uśrednianiem, aby nie zniwelować oceny modelu dla klasy mniej licznej (w naszym przypadku 20% najbardziej skutecznych piłkarzy).

8. Wyniki testów

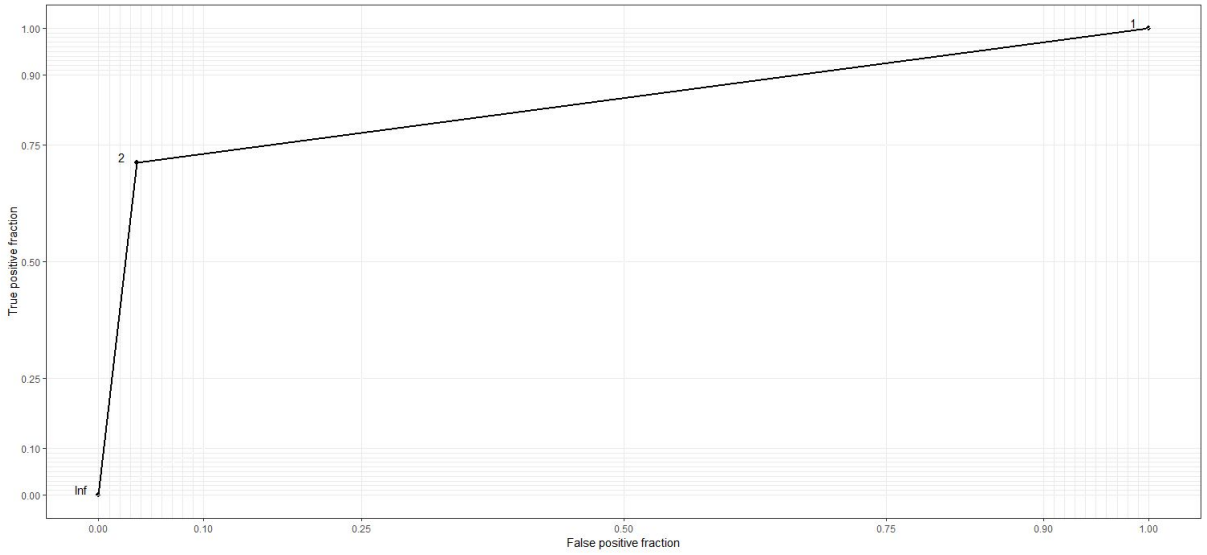
Przeprowadziliśmy dokładne testy dla modeli Drzewa Decyzyjnego i Lasu Losowego, w celu jak najlepszego dopasowania ich parametrów, które zostały opisane w punkcie 6.2. i 6.3. Następnie porównaliśmy zbudowane modele, również z modelem Naiwnego Klasyfikatora Bayesa.

8.1. Naiwny Klasyfikator Bayesa

W celu stworzenia modelu lasu losowego trenowaliśmy go przy pomocy funkcji train() z pakietu caret, która jako metodę klasyfikacji wykorzystywała implementację lasu losowego z pakietu randomForest. Model był trenowany z wykorzystaniem walidacji k-krotnej na wydzielonej treningowej części całego zbioru danych z parametrem k równym 10 i liczbą powtórzeń równej 3, co daje 30 testów dla każdej kombinacji parametrów.

Measure	Value
mean(acc(train))	0.9187755
acc(valid)	0.8745271
prec(valid)	0.8378405

Tabela 4. Wyniki klasyfikacji Naiwnego Klasyfikatora Bayesa.



Rysunek 1. Krzywa ROC dla Naiwnego Klasyfikatora Bayesa.

8.2. Drzewo decyzyjne

W przypadku drzewa decyzyjnego zostały przeprowadzone testy dla zmian pojedynczego parametru, który był używany w treningu. Każdy z opisanych parametrów w punkcie 6.2 został przetestowany dla różnych wartości. Wartości domyślne pochodzą z podstawowych ustawień treningu drzewa w pakiecie rpart.

	startowa	krok	maksimum	min. Najlepsza	domyślna
maxdepth	1	1	100	5	30
minsplit	20	10	2000	30	20
minbucket	20	20	2000	20	minsplit/3
cp	0,001	0,001	1	0,01	0,01
maxcompete	20	20	2000	brak wpływu	4

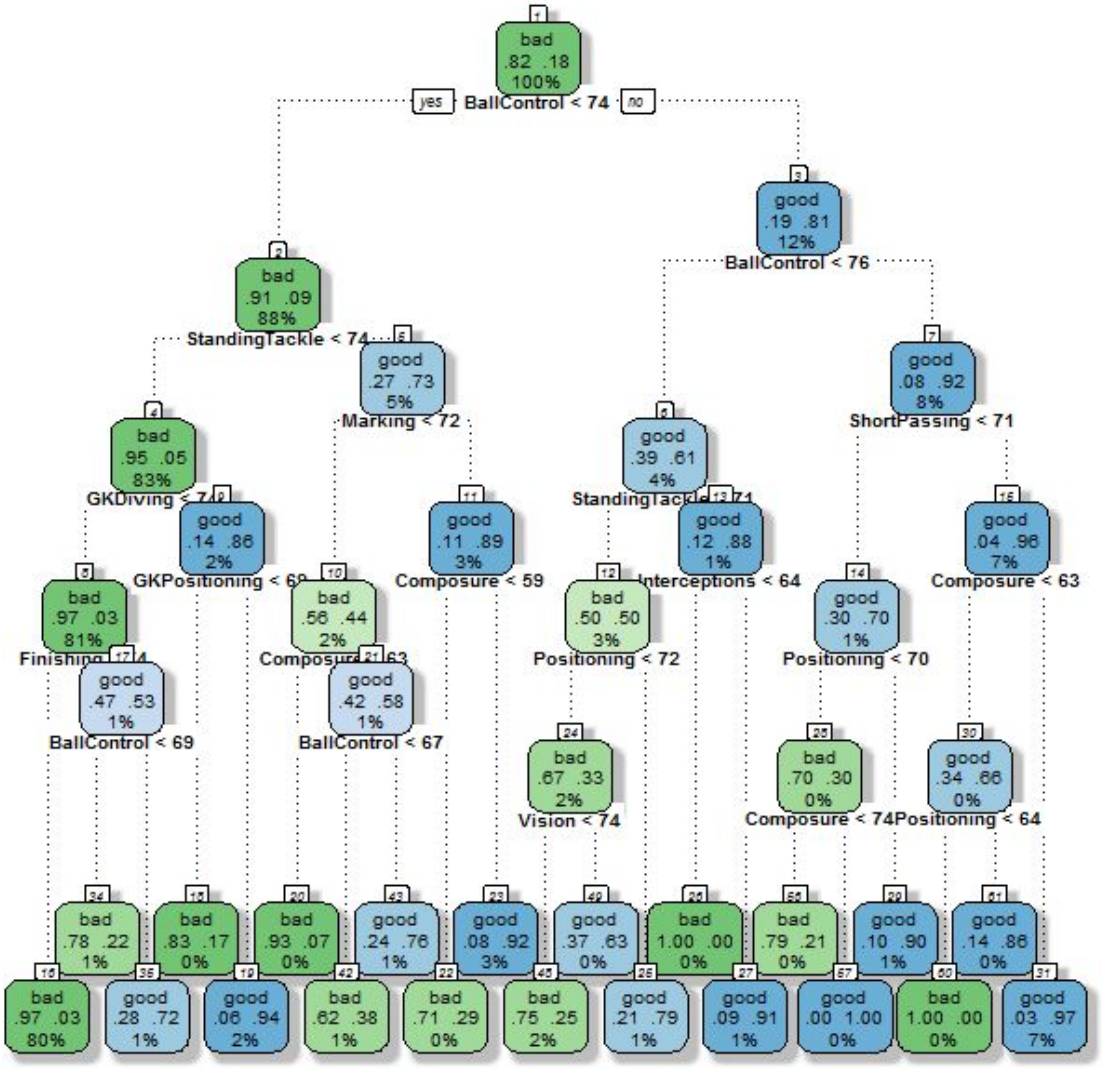
xval	1	1	100	brak wpływu	10
usersurrogate	0	1	2	brak wpływu	2
maxsurrogate	1	1	100	brak wpływu	5

Tabela 5. Wybór najlepszych wartości dla każdego z parametrów

Następnie dla najlepszych wartości dla każdego parametru oraz ich okolicznych wartości zostały przeprowadzone testy dla różnych kombinacji, gdzie część wyników wraz z najlepszymi została zamieszczona w Tabeli 3.

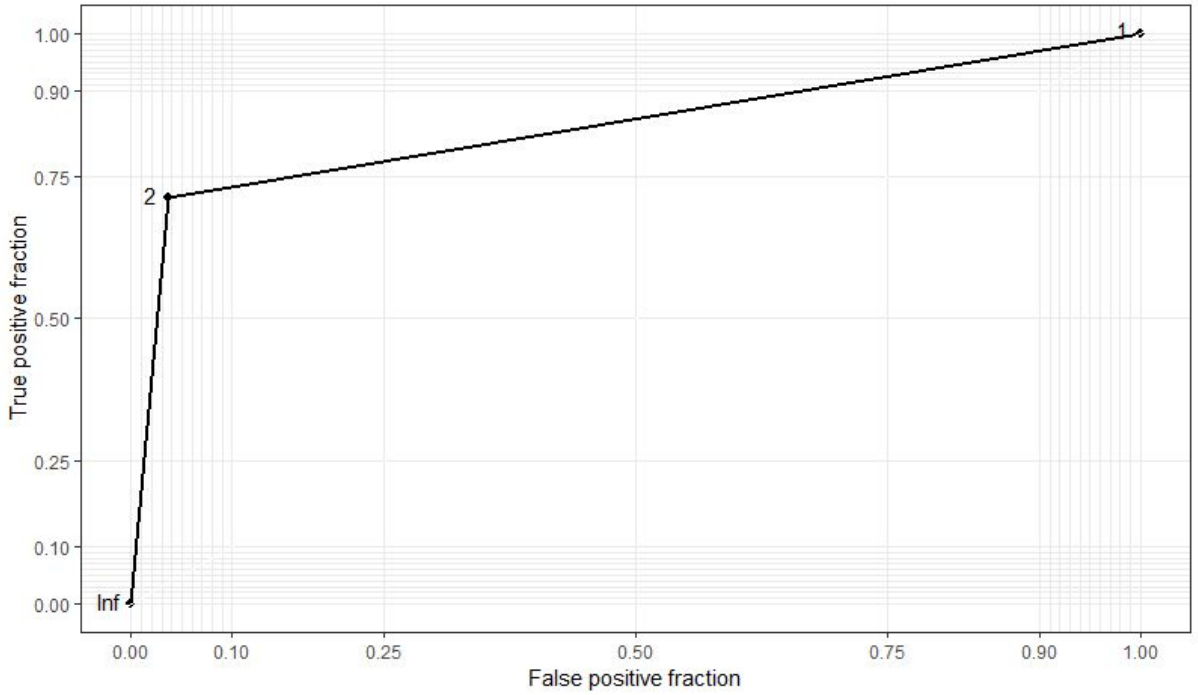
	accuracy	accuracy_micro	precision	max depth	minsplit	minbucket	cp
1	0,9009	0,9377	0,8802	8	20	3	0,001
2	0,8984	0,9372	0,8819	10	30	3	0,001
3	0,9030	0,9383	0,8801	5	15	3	0,001
4	0,8973	0,9368	0,8817	8	15	3	0,001
5	0,8931	0,9350	0,8801	10	15	3	0,001
6	0,8879	0,9350	0,8890	15	15	3	0,001
7	0,8879	0,9350	0,8890	15	15	3	0,01
8	0,9036	0,9386	0,8803	5	25	5	0,001
9	0,9036	0,9386	0,8803	5	25	5	0,01
10	0,8947	0,9368	0,8856	6	20	3	0,01
11	0,8953	0,9370	0,8858	6	25	5	0,01
12	0,9023	0,9377	0,8782	8	25	5	0,01
13	0,9028	0,9381	0,8795	8	25	3	0,01
14	0,9009	0,9377	0,8802	8	20	3	0,01
15	0,8981	0,9363	0,8784	10	25	3	0,01
16	0,8968	0,9363	0,8804	10	25	5	0,01
17	0,8933	0,9366	0,8870	15	25	5	0,01
18	**0,9236**	**0,9486**	**0,8803**	**5**	**25**	**5**	**0,01**

Tabela 6. Testowane parametry drzewa decyzyjnego



Rysunek 2. Drzewo decyzyjne dla wybranych najlepszych parametrów.

Z otrzymanego drzewa można wywnioskować jakie główne cechy piłkarza determinują czy będzie on skategoryzowany jako dobry czy słaby. Otrzymane wyniki pokrywają się z tabelą znaczenia zmiennych, zatem można uznać, że zostały wybrane poprawne cechy do klasyfikacji.



Rysunek 3. Krzywa ROC dla drzewa o wybranych parametrach.

8.3. Las losowy

W celu stworzenia modelu lasu losowego trenowaliśmy go przy pomocy funkcji train() z pakietu caret, która jako metodę klasyfikacji wykorzystywała implementację lasu losowego z pakietu randomForest. Model był trenowany z wykorzystaniem walidacji k-krotnej na wydzielonej treningowej części całego zbioru danych z parametrem k równym 10 i liczbą powtórzeń równej 3, co daje 30 testów dla każdej kombinacji parametrów. Zalecaną wartością parametru mtry jest sqrt(p), gdzie p to liczba cech w analizowanym zbiorze. Taką wartość tego parametru ustaliliśmy, dokładniej mtry = 6.480741.
Ze względu na bardzo powolne trenowanie modelu, zdecydowaliśmy się modyfikować pojedyncze parametry.
Następnie postanowiliśmy dobrać odpowiednią wartość parametru ntree. W tabeli 3. przedstawiliśmy statystyki dokładności klasyfikacji przykładów przez las losowy w zależności od parametru ntree. Najlepsza średnia dokładność została osiągnięta dla 500 drzew i dla tego parametru stroiliśmy dalej kolejne parametry.

	Accuracy					
ntree	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
50	0.9371555	0.9503859	0.9548209	0.9531258	0.9570485	0.9636564
100	0.9493392	0.9581152	0.9630650	0.9626027	0.9677599	0.9768722
200	0.9515419	0.9592511	0.9636364	0.9631552	0.9666850	0.9713341
500	0.9514884	0.9581498	0.9647189	0.9637424	0.9680617	0.9757709
1000	0.9504405	0.9586895	0.9636564	0.9632654	0.9677335	0.9713656
1500	0.9536424	0.9595155	0.9625344	0.9634850	0.9669604	0.9812569
2000	0.9449339	0.9592174	0.9647189	0.9636324	0.9688445	0.9757709
2500	0.9504405	0.9584251	0.9630656	0.9637060	0.9680617	0.9779493

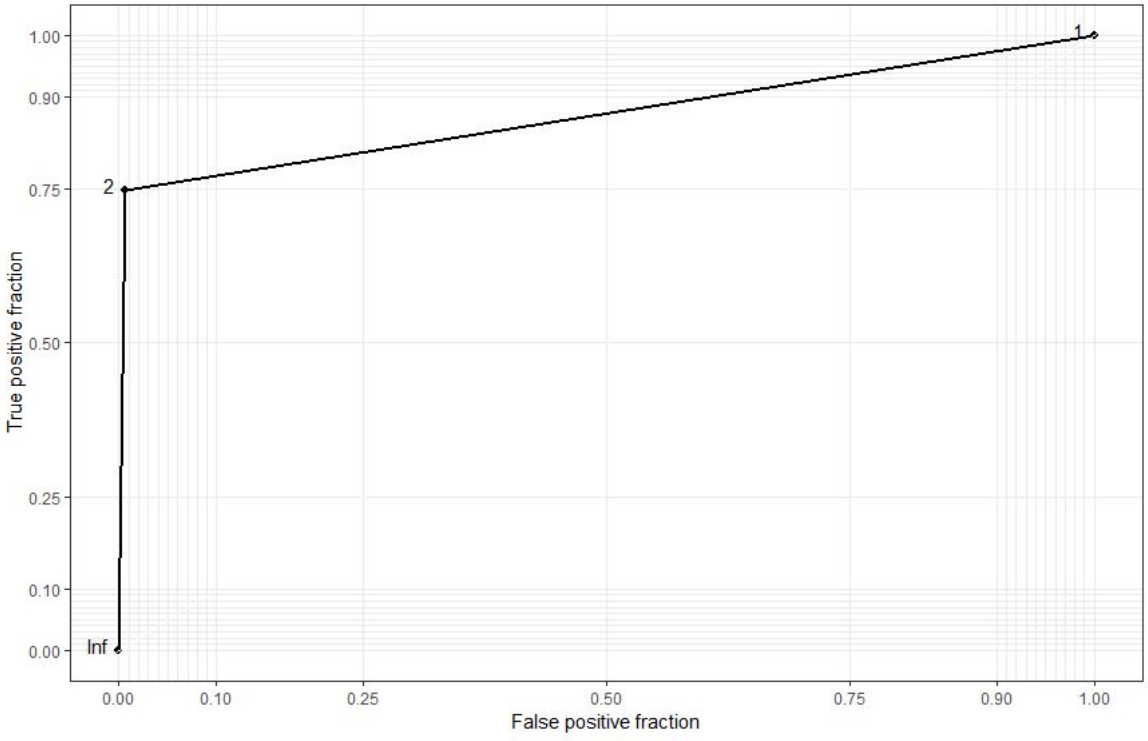
Tabela 7. Statystyki dokładności klasyfikacji lasu losowego dla różnych wartości parametru *ntree*.

W tabeli 8. zaprezentowaliśmy dokładność klasyfikacji dla ustalonego parametry ntree = 500 i różnych wartościach parametru maxnodes. Najlepszą dokładność uzyskuje dla maxnodes = 21, dalej zaczyna znowu spadać.

maxnodes	mean(Accuracy)
5	0.9224143
7	0.9213503
9	0.9327387
11	0.9357903
13	0.9364122
15	0.9360815
17	0.9415554
19	0.9428401
21	0.9439415
23	0.9436121

Tabela 8. Dokładność klasyfikacji dla różnych wartości parametru maxnodes.

Najlepszy model lasu losowego powstał przy następujących wartościach parametrów:
ntree = 500
maxnodes = 21
minsplit = 20
mtry = 6.480741.
W przypadku pozostałych parametrów pozostały domyślne wartości tego modelu.
Dla najlepszego zbudowanego modelu wielkości miar dla całego zbioru trenującego są następujące:
accuracy = 0.9556199
precision = 0.8705974.



Rysunek 4. Krzywa ROC dla Lasu Losowego.

8. Podsumowanie

Metody klasyfikacji, które wymieniliśmy zostały poddane testom, podczas których dopasowaliśmy możliwe do strojenia parametry. Miało to na celu uzyskanie jak najwyższej jakości klasyfikacji, przy jednoczesnych staraniach uniknięcia nadmiernego dopasowania modeli do danych (zarówno bezpośredniego do danych trenujących, jak i pośredniego do testowych).

W tabeli 9. prezentujemy porównanie wyników dla wszystkich modeli, które udało się uzyskać dla zbiorów walidacyjnych. Zgodnie z oczekiwaniami, Naiwny Klasyfikator Bayesa radzi sobie w tym zestawieniu najgorzej, ze względu na niespełnienie warunku o całkowitej niezależności cech. Najlepiej wypada Las losowy. Tak wysokie wyniki świadczą o tym, że udało się w znacznym stopniu uniknąć nadmiernego dopasowania modeli do danych trenujących, ale także pokazują, że dane użyte w tym zadaniu były nieco specyficzne, o dość jednorodnej strukturze i stosunkowo dobrej możliwości klasyfikacji.

Measure	Naive Bayes	Decision Tree	Random Forest
Accuracy	0.8745271	0.9133113	0.9489925
Precision	0.8378405	0.8715321	0.8559731

Tabela 9. Wyniki modeli na zbiorze walidacyjnym.