Raport

Soccer Players Clustering

Michał Iwaniuk, Anna Ostrowska

1. Wstęp

Celem projektu było zapoznanie się z pojęciem klasteryzacji danych. Projekt był podzielony na 3 kamienie milowe. KM1 – EDA, zapoznanie się ze zbiorem danych oraz zrozumienie jego cech. KM2 – feature engineering, odpowiednie przetworzenie zbioru danych, wyczyszczenie oraz wybór cech. KM3 – stworzenie finalnych modeli klasteryzujących nasze dane i ocenienie ich jakości, wyciągnięcie wniosków.

Przez cały okres tworzenia projektu walidowaliśmy projekt naszych kolegów oraz byliśmy przez kolejnych walidowani. Zadaniem walidatorów było ciągłe sprawdzanie czy np. nie został popełniony gdzieś błąd logiczny w trakcie procesu oraz ewentualna pomoc w całym zagadnieniu.

Ramka danych, z których korzystaliśmy pochodzi ze strony: <https://www.kaggle.com/datasets/spicemix/soccer-detailed-players-match-data?resource=download&select=players_away_matches.csv>, a nasze repozytorium z całym kodem z każdego etapu projektu oraz raportami i prezentacją końcową można znaleźć na Githubie pod linkiem: <https://github.com/annaostrowska03/Soccer-Players-Clustering>.

1. EDA

Nasze dane pochodzą z meczów piłki nożnej i zawierają szczegółowe statystyki meczowe dla każdego gracza. Dataset składa się z 76185 obserwacji oraz 127 kolumn.

Przykładowe znaczenia kolumn: awayScore (wynik drużyny gracza w tym meczu) homeScore (wynik drużyny przeciwnej) homeTeamName (nazwa drużyny przeciwnej) home/awayTeamTacticalSchema (schemat taktyczny drużyn) matchWeek (numer kolejki) awayTeamName (nazwa drużyny gracza) matchDate (data meczu) playerName (imię i nazwisko gracza) championshipName (nazwa rozgrywek) age (wiek gracza) assists (asysty w tym meczu) xg (oczekiwane bramki) xa (oczekiwane asysty) touches in different areas (dotknięcia piłki w różnych obszarach).

Po sprawdzeniu zawartości ramki ustalilismy nasz cel biznesowy: znalezienie jak najlepszych klastrów do analizy danych dotyczących występu graczy w meczach piłki nożnej w celu zapewnienia wglądu drużynom, trenerom, analitykom, agencjom scoutingowym i firmom bukmacherskim.

Już na tym etapie usunelismy kolumnę playerName ponieważ uznaliśmy ze to jak się nazywa dany gracz jest nieistotne.

Wiekszosc zmiennych jest numerycznych, zaledwie 8 kategorycznych. Dane zawierają dużo wartości NA bo pojawiają się w az 100 kolumnach.

W danych mamy bardzo dużo kolumn mocno skorelowanych (tj. abs(corr) >= 0.5) co widać nizej. Jest to uzyteczna wiedza w kontekście przyszłego usuwania niepotrzebnych kolumn.

Obraz zawierający tekst, zrzut ekranu, linia, numer

Opis wygenerowany automatycznie

Przebadalismy zmienne kategoryczne i okazało się ze jest dosyc sporo wartości unikalnych. Prezentowaly się one tak:

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Okazalo się ze dane pochodzą z 5 najlepszych lig europejskich oraz ze statystyki były prowadzone w latach 2019-2021. Na tym etapie postanowiliśmy usunąć kolumnę team\_name ponieważ zawiera dokładnie te same dane co kolumna homeTeamName

Nastepnie przebadaliśmy zmienne numeryczne. Oto przykladowe rozklady zmiennych:

Obraz zawierający diagram, zrzut ekranu, Prostokąt, Plan

Opis wygenerowany automatycznieObraz zawierający diagram, linia, tekst, Wykres

Opis wygenerowany automatycznie

Pomimo ze po boxplotach można by się spodziewac ze w danych mamy duzo outlierow to jednak po zrobieniu histogramow okazuje się ze wiekszosc cech ma rozklad zblizony do geometrycznego i de facto nie ma tu outlierow, po prostu dane sa „rozciagniete”.

1. Feature engineering

Zabralismy się za obrobke danych najpierw usuwając wartości NA.

Sprawdzilismy ile wartości NA zawierają poszczególne kolumny i wygladalo to tak:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Tutaj postanowiliśmy usunąć wszystkie kolumny zawierające „pct”, ponieważ jest to skrót od "percentage" (procent). Przykładowo, kolumna „dribbles\_completed\_pct” oznacza procent udanych dryblingów. Informacje zawarte w tych kolumnach są już obecne w innych kolumnach, takich jak „dribbles” i „dribbles\_completed”. Usunięcie tych kolumn jest uzasadnione, ponieważ pozwala nam także pozbyć się wartości NA.

Następnie zdecydowaliśmy się usunąć kolumny z dużą korelacją (tj. korelacja >= 0.7) oraz te, których informacje są zawarte w innych kolumnach. Na przykład zostawiliśmy kolumnę „passes”, ale usunęliśmy „passes\_total\_distance”, ponieważ jest ona silnie skorelowana z liczbą podań. Wiele kolumn usuwaliśmy wybiórczo: najpierw wypisywaliśmy kolumny silnie skorelowane z kolumną, którą chcieliśmy zachować, a następnie wybieraliśmy spośród nich te, które uznaliśmy za nieistotne.

Następnie usunęliśmy 50 wierszy (meczów), w których nie były prowadzone szczegółowe statystyki i zawierały one tylko kilka ogólnych informacji o meczu, co pozwoliło nam również pozbyć się wartości NA z 44 kolumn. Pozostałe wartości NA, które występowały już tylko w 5 kolumnach numerycznych i 1 kategorycznej, uzupełniliśmy medianą (dla kolumn numerycznych) oraz wartością „unknown” (dla kolumny kategorycznej).

Pozbyliśmy się już niepotrzebnych kolumn i wartości NA, więc pozostało zakodować zmienne. Przypomnijmy, jak wygląda rozkład zmiennych kategorycznych: wygląda rozklad zmiennych kategorycznych:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Formacje obu drużyn (awayTeamTacticalSchema, homeTeamTacticalSchema) postanowiliśmy zakodować w taki sposób, aby zapisać liczbę obrońców, pomocników i napastników w trzech osobnych kolumnach. Na przykład, w formacji (4-4-2) mamy 4 obrońców, 4 pomocników i 2 napastników, natomiast w formacji (4-2-3-1) mamy 4 obrońców, 5 pomocników (suma środkowych wartości) oraz 1 napastnika. Kolumny awayTeamName oraz homeTeamName uznaliśmy za nieistotne, więc je usunęliśmy. Kolumnę matchDate podzieliliśmy na dwie kolumny: „year” oraz „month” danego meczu, przy czym wartości „month” przekształciliśmy na wartości cykliczne. Zastosowaliśmy OneHotEncoding dla kolumny „championship\_name”.

Następnie zastosowaliśmy automatyczną detekcję outlierow przy użyciu metody kNN z biblioteki pyod, na wszelki wypadek, gdyby w zbiorze danych pojawiły się jakiekolwiek niespodziewane odstępstwa. Usnęliśmy 4% obserwacji. Jak się okazało nie zaburzyło to nam zbioru i nie usunęło rzadkich aczkolwiek wartościowych informacji takich jak np. duża liczba goli w meczu.

Następnie zastosowaliśmy skalowanie MinMax na naszych danych, ponieważ dane w ogóle nie były zbliżone do rozkładu normalnego.

W procesie wstępnego modelowania okazało się, że najbardziej optymalna liczba klastrów dla metody KMeans wynosi 5. Okazało się, że ta liczba wynika z faktu, że klastry zostały ustalone według 5 lig, w których prowadzone są rozgrywki. Można uznać to za wstępny sukces.

1. Final models

W ostatnim etapie naszego projektu musieliśmy przetestować różne modele omówione na zajęciach laboratoryjnych i wybrac dla nich odpowiednie parametry, a następnie porównać je za pomocą różnych metryk i wybrać najlepszy dla naszego zagadnienia model.

Po puszczeniu napisanego kodu na całą noc, kod dalej się kompilował, dlatego zdecydowaliśmy się wziąć reprezentatywną próbkę 10000 wierszy z naszej ramki danych i na niej wykonywać dalsze sprawdzanie modeli.

* 1. KMeans

Na początku przetestowaliśmy model KMeans i sprawdziliśmy, które parametry dają najlepsze rezultaty. Użyliśmy w tym celu 4 metryk: metoda łokcia dla sumy kwadratów odległości punktów od ich środków klastra (WSS), Silhouette score, Calinski-Harabasz score oraz Davies-Bouldin score. W każdej metryce wyszedł jednoznaczny wynik – najlepsza liczba klastrów dla modelu K-Means wynosi 5.

A graph on a screen

Description automatically generated A graph on a screen

Description automatically generated

A graph with a line going up

Description automatically generated A graph with a line going up

Description automatically generated

Nastęnie używając tych samych metryk sprawdziliśmy wpływ innych parametrów na wyniki modelu K-Means:

A screenshot of a computer

Description automatically generated

Zauważylismy, że parametr algorithm nie zmienia wyników dla żadnej z metryk w modelu K-Means dla naszych danych, a parametr „init” zmienia, ale bardzo nieznacznie. Zdecydowaliśmy się wybrać parametry n\_clusters=5, init="k\_means++" jako najlepsze dla modelu KMeans.

* 1. KMedoids

Następnie przeanalizowaliśmy parametry dla modelu KMedoids. Porównaliśmy wyniki za pomocą różnych metryk dla tego modelu z różnymi wartościami parametru n\_clusters:

A graph on a screen

Description automatically generated A graph with blue lines

Description automatically generated

Patrząc na te metryki nie mogliśmy zdecydować, która wartość n\_clusters jest najlepsza (5,7,11,15), dlatego sprawdziliśmy jeszcze metrykę Silhouette:

A graph with blue lines and dots

Description automatically generated

Po tych wynikach byliśmy już pewni, że wybieramy n\_clusters=11 jako najlepszy parametr dla modelu Kmedoids.

* 1. DBSCAN

W następnym kroku patrzyliśmy na model DBSCAN i jego parametry. Zgodnie z instrukcją podaną na zajęciach laboratoryjnych sprawdziliśmy wymiar naszej ramki danych (81 kolumn po KM2, gdyż usunęliśmy tam część kolumn z wysoką korelacją) i ustawiliśmy minPts na podwojoną wartość wymiaru, czyli 162. Następnie sprawdziliśmy dla każdego punktu odległość do (𝚖𝚒𝚗𝙿𝚝𝚜−1) najbliższego sąsiada (różnego od ustalonego punktu) i przedstawiliśmy wyniki na wykresie.

A graph with a line going up

Description automatically generated

Według instrukcji z zajęć następnie należało znaleźć ostatni 'duży skok' (lub łokieć) na wykresie i wybrać wartość odległości jako 𝜀.

Niestety z powyższego wykresu ciężko nam było stwierdzić, gdzie dokładnie znajduje się ten punkt, więc wybraliśmy kilka przedziałów (2.2-2.05,1.7-1.9, 1.2-1.4), w których według wykresu powinien się on znajdować i, iterując po różnych wartościach na tych przedziałach wyliczyliśmy najlepszy parametr epsilon.

Przy okazji przetestowaliśmy w tej samej pętli parametr ,,metric” – iterowaliśmy po metrykach i epsilonach szukając najlepszej kombinacji parametrów.

Przetestowane parametry:



Najpierw użyliśmy metryki Silhouette i według niej najlepsza kombinacja parametrów przedstawia się następująco:

A black screen with white text

Description automatically generated.

Następnie sprawdziliśmy to rónież za pomocą innej metryki – indeksu Davies-Bouldin:

.

Ponieważ dla obu metryk najlepsza kombinacja parametrów była ta sama, stwierdziliśmy, że na pewno dalej będziemy brali pod uwagę jedynie te parametry: eps: 1.9, metric: ,,euclidean”.

* 1. AgglomerativeClustering

W tym modelu chcieliśmy sprawdzić wszystkie mozliwe wartości parametru ,,linkage” dla różnych liczb klastrów, żeby zobaczyć, która kombinacja przedstawia się najlepiej.

Wyniki porównaliśmy ze sobą wzajemnie i z modelem KMeans (jako, że był to pierwszy przedstawiony model na zajęciach) na wykresie za pomocą metryki Davies-Bouldin score:

A graph with lines and numbers

Description automatically generated with medium confidence

Według powyższego wykresu najlepiej wypadają parametry linkage="single" i n\_clusters=4 oraz linkage="average" i n\_clusters=6.

Jako, że jedna metryka może mylić, bo nie patrzy na wszystkie właściwości klasteryzacji, to zdecydowaliśmy się jeszcze użyć metryki Silhouette:

A graph on a screen

Description automatically generated

Tutaj wyniki były całkiem inne: linkage=”average” z n\_clusters=6 w poprzedniej metryce wypadał prawie najlepiej, za to tutaj wypadł zdecydowanie najgorzej. Najlepsze w tej metryce wyniki osiągnęły linkage=”complete” z n\_clusters=6 i „ward” z n\_clusters=5 wraz z KMeans.

Zdecydowalismy się zapisać otrzymane najlepsze parametry dla obu tych metryk i na końcu przetestować i porównać z innymi modelami.

* 1. GaussianMixture (GMM)

W tym modelu również przeanalizowaliśmy dostępne parametry i przeiterowaliśmy po różnych ich wartościach w celu sprawdzenia, która kombinacja daje najlepsze wyniki.

Sprawdzone parametry:

A close up of text

Description automatically generated.

Według metryki Silhouette najlepsze wyniki dawała poniższa kombinacja parametrów:



Patrząc na matrykę Davies-Bouldin score, inna kombinacja parametrów wychodziła najlepiej:



Zapisaliśmy obie kombinacje do dalszego porównywania i analizy.

* 1. Najlepsze modele

Zgodnie z metrykami, którym się przyjrzeliśmy, najlepsze wyniki dają modele:

* KMeans z parametrami: n\_clusters=5, init=„k\_means++”
* AgglomerativeClustering z parametrami: linkage=„single” z n\_clusters=4, linkage=„average” z n\_clusters =6, linkage=ward i n\_clusters =5, linkage=complete i n\_clusters =6
* GMM z: covariance\_type: full, n\_components: 5, covariance\_type: tied, n\_components: 6
* DBSCAN z: min\_samples=162, eps=1.9, best\_metric=euclidean
* KMedoids z n\_clusters=11

W celu wybrania ostatecznego modelu i porównania ze sobą wyników tych modeli uzyliśmy kilku metryk i obliczyliśmy wyniki zwracane przez nie:

A screenshot of a computer

Description automatically generated

Aby łatwiej nam było przeanalizować wyniki, przesdtawiliśmy je w czytelniejszy sposób: za pomocą wykresów:

A screenshot of a graph

Description automatically generated

Patrząc na różne metryki ciężko było stwierdzić, który model wypada najlepiej, np. w pierwszych dwóch metrykach model DBSCAN jest najlepszy, ale za to w modelu ostatnim wypada zdecydowanie najgorzej.

Ostatecznie stwierdziliśmy, że najlepsze dla naszego celu modele to KMeans z parametrami: n\_clusters=5, init=„k\_means++” i AgglomerativeClustering z linkage=ward i n\_clusters.=5, jednak spojrzeliśmy dokładniej na tabelę wyników różnych metryk w tych dwóch modelach i, nieznacznie, KMeans wygrało w każdej metryce.

Zatem nasz ostateczny model końcowy to KMeans z parametrami: n\_clusters=5, init=„k\_means++” .

* 1. Wizualizacje

Zwizualizowaliśmy dla każdego z tych modeli, jak wygląda podział na klastry:

A screenshot of a computer screen

Description automatically generated

A red dots on a white background

Description automatically generated

A red dots on a white background

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A group of colorful dots

Description automatically generated

A group of colorful dots

Description automatically generated

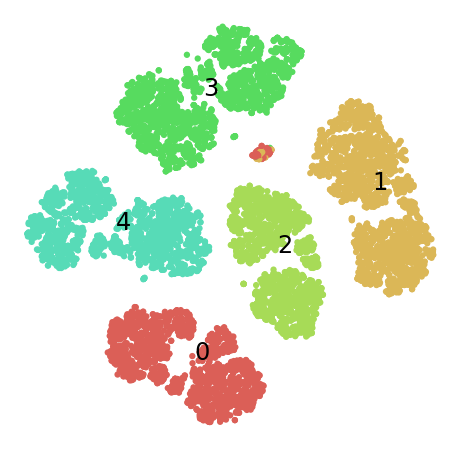
A yellow dots on a white background

Description automatically generated

A screenshot of a computer screen

Description automatically generated

* 1. TSNE dla KMeans



1. Walidacja

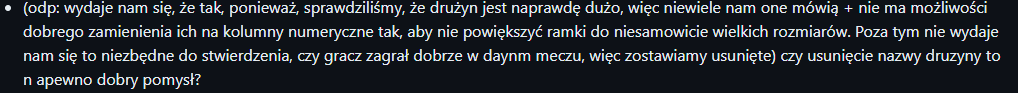
Dzięki regularnej wymianie opinii i uwag z grupą walidatorów udało nam się poprawić kilka błędów i niedopatrzeń,. Wszystkie uwagi były brane pod uwagę, realizowana większość z nich (nie wszystkie).

Jednak niezrealizowane uwagi również były przemyślane i omówione, zanim zostały odrzucone.

Przykłady uwag otrzymanych od walidatorów:







1. Podsumowanie

Analizując wszystkie modele przedstawione na zajęciach laboratoryjnych, znaleźliśmy najlepszy, naszym zdaniem, pośród nich model do realizacji naszego celu biznesowego.

Udało nam się ukończyć wszystkie etapy projektowe na czas i z sukcesem (bez większych błędów). Obecność i czujność walidatorów była tutaj kluczowa.

Wynik projektu – ostateczny model całkiem nas satysfakcjonuje i uważamy, że zrealizowaliśmy zamierzony cel projektu.

1. Bibliografia

* <https://www.kaggle.com/datasets/spicemix/soccer-detailed-players-match-data?resource=download&select=players_away_matches.csv>
* <https://github.com/annaostrowska03/Soccer-Players-Clustering>
* Platforma Leon