POLITECHNIKA GDAŃSKA
WYDZIAŁ FIZYKI TECHNICZNEJ I MATEMATYKI STOSOWANEJ

Imię i nazwisko słuchacza studiów podyplomowych: **Krzysztof Pietruczuk**
Poziom kształcenia: Studia podyplomowe
Kierunek studiów: **Inżynieria Danych - Data Science**

PRACA DYPLOMOWA
Gdańsk, czerwiec 2023

Porównanie efektywności wybranych algorytmów uczenia maszynowego w określaniu fenotypu komórek PBMC metodą cytometrii przepływowej.

Opiekun pracy dyplomowej:
prof. dr hab. Józef E. Sienkiewicz

POLITECHNIKA GDAŃSKA | WYDZIAŁ FIZYKI TECHNICZNEJ I MATEMATYKI STOSOWANEJ

OŚWIADCZENIE dotyczące pracy dyplomowej:

Porównanie efektywności wybranych algorytmów uczenia maszynowego w określaniu fenotypu komórek PBMC metodą cytometrii przepływowej.

(Comparison of the effectiveness of selected machine learning algorithms in determining the phenotype of PBMC cells by flow cytometry.)

Krzysztof Pietruczuk
ur. 21 listopad 1974, Strzelno

Wydział Fizyki Technicznej i Matematyki Stosowanej
Studia podyplomowe: Inżynieria Danych – Data Science

Świadomy odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jedn. Dz. U. z 2021 r., poz. 1062) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (tekst jedn. Dz.U. z 2021 r., poz. 478 z późn. zm.), a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca zaliczeniowa została opracowana przeze mnie samodzielnie.
Niniejsza praca zaliczeniowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej ze studiami podyplomowymi.
Wszystkie informacje umieszczone w ww. pracy zaliczeniowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.
Potwierdzam zgodność niniejszej wersji pracy zaliczeniowej z załączoną wersją elektroniczną.

Gdańsk, dnia 24.06.2023

Krzysztof Pietruczuk
(data i podpis)

Spis treści

Wykaz ważniejszych skrótów

CD	-	ang. cluster of differentiation, antygen różnicowania komórkowego
FCM	-	ang. flow cytometry method, cytometria przepływowa
FCS	-	ang. flow cytometry standard, standard zapisu pliku cytometrycznego
FSC	-	ang. forward scatter channel, detektor rozproszenia światła zgodny z kierunkiem wiązki laserowej
HDBSCAN	-	ang. hierarchical density-based spatial clustering of applicatiion with noise, hierarchiczne gęstościowe grupowanie przestrzenne aplikacji z hałasem
GMM	-	ang. Gaussian Mixture Model, model mieszaniny rozkładów Gaussa
PCA	-	ang. principal component analysis, analiza głównych składowych
PBMC	-	ang. peripheral blood mononuclear cells, jednojądrzaste komórki krwi obwodowej
SSC	-	ang. side scatter channel, detektor mierzący ilość światła rozproszonego pod kątem $\sphericalangle 90°$ do lasera
t-SNE	-	ang. t-Distributed Stochastic Neighbor Embedding, stochastyczna metoda porządkowania sąsiadów w oparciu o rozkład t

Abstrakt

Znaczenie cytometrii przepływowej w naukach biomedycznych jako metody zarówno badawczej jak i szeroko stosowanej w diagnostyce nadal wzrasta. Pomimo, iż technika ta liczy juz prawie 60 lat i ciągle jest rozwijana, to nadal największe trudności sprawia analiza uzyskanych ta drogą pomiarów.

Ponieważ istnieje potrzeba ujednolicenia i stworzenia pewnych standardów analizy, a co za tym idzie zautomatyzowania całego procesu postanowiono przyjrzeć się możliwoscią jakie dają algorytmy klasteryzacji danych. Wybrano trzy rodzaje metod (k-Means, HDBSCAN, GMM) i przetestowano ich skuteczność i możliwość zastosowania do automatycznej analizy danych cytometrycznych. Przy użyciu miar walidacji wykazano największą skuteczność jednej z technik klasteryzacji jaką jest GMM.

Jednak otrzymane wyniki i obserwacje z analizy 350 plików cytometrycznych nie pozwalają jednoznacznie stwierdzić, czy algorytm GMM stanowi najlepszy wybór spośród metod wykorzystywanych w analizie skupień. Badania tego typu powinny być rozszerzone o techniki nadzorowanego uczenia maszynowego, a nawet o koncepcję 'online learning' w którym w ramach tzw. uczenia ciągłego model jest regularnie aktualizowany na podstawie nowych danych napływających w czasie rzeczywistym.

1. Wstęp

1.1 Metoda cytometrii przepływowej

Cytometria przepływowa jest metodą laboratoryjną szeroko stosowaną zarówno w badaniach biomedycznych, jak i diagnostyce klinicznej. Jest to technika analityczna, ilościowa, która pozwala na szybką analizę dużej liczby komórek w zawiesinie. Podstawą tej metody jest pomiar światła rozpraszanego przez badane komórki oraz emitowanej przez nie fluorescencji. Jest to pierwsza w historii technika, która pozwala naukowcom analizować pojedyncze komórki. Charakteryzuje się możliwością pomiaru wielu parametrów komórek w krótkim czasie [1, 2].

Analizatory cytometryczne (tzw. cytometry) składają się z pięciu głównych elementów: źródła światła, układu hydraulicznego, układu optycznego, układu elektronicznego oraz komputera zdolnego do przetwarzania uzyskanych pomiarów [3].

Źródłem światła może być laser lub lampa łukowa. Laser wzbudza fluorochromy są przyłączone do komórek lub cząsteczek powodując emisję przez nie światła o różnych długościach fal, umożliwiając w ten sposób ich identyfikację i analizę.

Zadaniem układu hydraulicznego jest wytworzenie strumienia cieczy otaczającej zawiesinę komórek, dzięki czemu badane cząstki przechodzą przez punkt pomiarowy z odpowiednią prędkością i ustawiają się w wiązce światła jedna po drugiej.

Układ optyczny skupia wiązkę światła na badanej komórce. W jego skład wchodzi również tzw. optyka zbierająca, której zadaniem jest odfiltrowanie poszczególnych długości fal i skupienie promieni świetlnych na fotokatodzie detektorów. W cytometrze są dwa główne detektory to FSC (ang. forward scatter channel) i SSC (ang. side scatter channel). FSC wykrywa rozproszenie wzdłuż wiązki lasera, a jego intensywność jest wprost proporcjonalna do średnicy komórki, dlatego służy do określenia rozmiaru badanej cząstki. SSC mierzy rozproszenie pod kątem $\sphericalangle 90°$ do wiązki lasera [3, 4]. Dostarcza to informacji o ziarnistości komórek. Pozostałe detektory są zaangażowane w pomiary fluorescencji [3].

Układ elektroniczny służy do przetwarzania sygnałów świetlnych na impulsy elektryczne, wzmacniania sygnału i przekształcania sygnału analogowego na cyfrowy.

1.2 Zjawiska fizykochemiczne wykorzystywane w cytometrii przepływowej

Podstawowe zjawiska fizykochemiczne wykorzystywane w cytometrii to rozpraszanie światła oraz fluorescencja.

Światło emitowane przez laser jest rozpraszane na komórkach przepływających w strumieniu cieczy. Pomiar rozproszenia światła na detektorach FSC i SSC pozwala określić zarówno wielkość, jak i ziarnistość komórek [3, 4].

Zjawisko fluorescencji występuje w cytometrii przepływowej dzięki zastosowaniu barwników fluorescencyjnych, tzw. fluorochromów. Są one skoniugowane z przeciwciałami, które specyficznie reagują z określonymi antygenami/markerami komórkowymi. Przed pomiarem, komórki barwi się przeciwciałami znakowanymi fluorochromami. W świetle lasera, w fluorochromie związanym z daną strukturą komórkową następuje absorpcja energii i przeniesienie elektronu na jeden z wyższych stanów oscylacyjnych poziomu elektronowego. Następnie za pomocą przemian radialnych, w tym zjawiska fluorescencji, elektron powraca ze stanu wzbudzonego do singletowego stanu podstawowego, który jest rejestrowany na detektorach jako widmo emisyjne. Istnieje wiele rodzajów fluorochromów stosowanych w cytometrii, a każdy z nich ma charakterystyczne widmo absorpcyjne i emisyjne, więc będą wzbudzane przez różne długości fal światła [3, 5].

1.3 Zastosowanie cytometrii przepływowej

Analizatory wykorzystywane są w laboratoriach badawczych i klinicznych do oceny ekspresji antygenów powierzchniowych (CD) i wewnątrz komórkowych antygenów (np. cytokin, hormonów), aktywności enzymów (np. kinazy, kaspazy) czy ekspresji genów i transkrypcji mRNA dla różnych komórek produkty [6]. Pozwalają na ocenę cyklu komórkowego, stanu mitochondriów czy procesów takich jak apoptoza, autofagia czy starzenie się komórek. Za pomocą tego typu cytometrów możliwe jest również ilościowe określenie zawartości niektórych substancji biologicznych w różnych płynach ustrojowych (np. surowicy, płynie mózgowo-rdzeniowym, cieczy wodnistej).

Sortery komórek mają możliwość nie tylko zbierania i analizowania danych o przepływających komórkach, ale także ich sortowania, czyli odchylania przepływających cząstek zgodnie z różnicami w ich potencjale elektrycznym [6]. Stopień czystości izolacji danych komórkowych wynosi ponad 99 procent. Dzięki dużej wydajności sortery służą do izolowania bardzo rzadkich komórek, takich jak komórki nowotworowe zawieszone we krwi, erytrocyty płodu czy komórki modyfikowane genetycznie [6, 7]. Istnieją również cytometry przeznaczone do określonych zadań, takich jak wykrywanie, charakteryzacja i liczenie mikroorganizmów zawieszonych w wodzie czy komórek somatycznych i bakteryjnych w mleku [8].

W medycynie największą popularnością cieszy się metoda cytometrii przepływowej, przede wszystkim w takich dziedzinach jak hematologia i onkologia, zarówno w diagnostyce chorób nowotworowych, ich klasyfikacji, jak i w monitorowaniu postępów leczenia [7].

1.4 Problemy i wyzwania cytometrii przepływowej

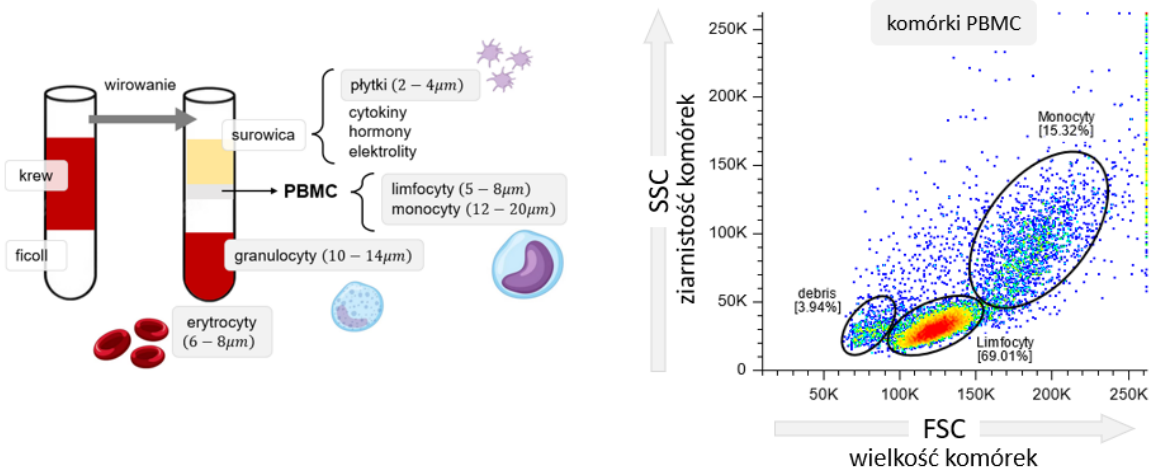
Obecnie można jednocześnie analizować około 30 parametrów komórek. Jest to prawdopodobnie górna granica tradycyjnych metod cytometrii przepływowej wynikająca zarówno z kosztów, jak i ograniczeń metody [9].
Widma emisji fluorescencji poszczególnych fluorochromów częściowo się pokrywają, co może prowadzić zarówno do wyników fałszywie dodatnich, jak i fałszywie ujemnych, dlatego konieczne jest zastosowanie tzw. kompensacji, czyli matematycznej korekty mierzonych sygnałów. Każdy fluorochrom ma charakterystyczny stopień nakładania się widma emisyjnego z widmami innych fluorochromów [3]. Stanowi to utrudnienie w analizie pomiaru. Inne źródła błędów w pomiarach cytometrycznych to: autofluorescencja, nieswoiste wiązanie przeciwciała oraz czynnik ludzki [10].

Błędna może być także analiza wyników. Możemy mieć do czynienia z występowaniem dubletów, nieprawidłowymi strategiami bramkowania populacji komórek, czy błędną interpretacją immunofenotypu [10]. Przyszłym wyzwaniem jest opracowanie zunifikowanego systemu opracowywania uzyskanych danych pomiarowych, nad czym czuwa International Society for the Advancement of Cytometry (ISAC) [11].

1.5 Manualna analiza pomiarów cytometrycznych

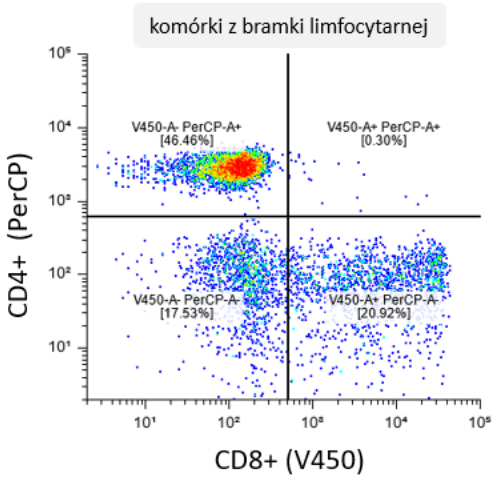
Wszystkie dane z dotyczące pomiarów zapisywane są w plikach, które później podlegają analizie w specjalnie opracowanych programach zarówno komercyjnych jak i do użytku freeware.

Pomimo ciągłego rozwoju cytometrii przepływowej, nadal podstawową metodą analizy wyników z wykorzystaniem dedykowanego oprogramowania jest manualne bramkowanie [12]. W przypadku leukocytów i/lub komórek PBMC analiza taka rozpoczyna się od wybrania na wykresie utworzonym z wyników pomiaru rozproszenia światła mierzonego przez detektory FSC i SSC, grupy komórek mieszczących się w określonym, charakterystycznym przedziale wielkości i stopnia ziarnistości.
Jest to pierwszy etap analizy przez który należy przejść chcąc uzyskać interesujące nas informacje na temat badanych komórek, obojętnie czy będziemy badać fenotyp danej populacji czy rodzaj i funkcje struktur wewnętrznych w komórkach.



Rys.1 Metoda uzyskania komórek PBMC z krwi pełnej oraz ich obraz na wykresie typu dot plot wartości FSC i SSC. Na wykresie został pokazany manualny sposób bramkowania grup komórek wykonany przy użyciu bezpłatnego programu Floreada.io, który jest dostępny on-line na stronie: https://floreada.io/analysis.

W następnym etapie wybieramy komórki o danym fenotypie z zaznaczonej grupy na podstawie pomiarów fluorescencji światła emitowanego przez użyte przez nas fluorochromy sprzężone z przeciwciałami o powinowactwie do konkretnych markerów - antygenów róznicowania komórkowego, charakterystycznych dla badanej populacji.



Rys.2 Wykres typu dot plot komórek bramki limfocytarnej. Rozdział komórek na posiadające na sobie antygen różnicowania CD4 i antygen CD8.

Na rysunku 2 pokazany jest przykład manualnej analizy bramki limfocytarnej pod katem obecności komórek posiadających na swojej powierzchni marker CD4 charakterystyczny dla limfocytów pomocniczych i obecności CD8 charakterystycznego dla limfocytów cytotoksycznych. Komórki CD4+ są znakowane fluorochromem PerCP a CD8+ fluorochromem V450. W lewym-górnym kwadrancie zaznaczona została populacja CD4+CD8-, kwadrant prawy-górny zawiera komórki CD4+CD8+. Komórki charakteryzujące się obecnością CD4-CD8+ znajdują się w prawym dolnym kwadrancie. Lewy-dolny kwadrant zawiera komórki CD4-CD8- czyli nie posiadające tych antygenów.

Analiza manualna jest dość subiektywną techniką co powoduje, że podejmowane w niej decyzje są podatne na błędy analityka. Taka analiza wymaga dużego doświadczenia i wiedzy specjalistycznej. Kolejnym wyzwaniem jest jej czasochłonność, szczególnie w przypadku dużych zestawów danych gdy oznaczanych jest kilkanaście parametrów [12].

1.6 Techniki grupowania a analiza danych cytometrycznych

Techniki grupowania, które należą do technik analizy danych mogą pomóc przy automatyzacji analizy plików cytometrycznych, zmniejszając w znacznym stopniu jej czasochłonność i zwiększając odporność na błędy.
Sam termin grupowania, odnosi się do procesu dzielenia zbioru danych na podstawie obranych kryteriów. Metody grupowania obejmują zarówno algorytmy klasyfikacji, jak i klasteryzacji. W technikach klasyfikacji, obiekty są przypisywane do znanych klas, natomiast techniki klasteryzacji identyfikują naturalne grupy w zestawie danych bez wcześniejszych etykiet. Są to dwa zupełnie różne podejścia w analizie danych [13].

Techniki klasteryzacji należą do metod uczenia maszynowego nienadzorowanego. Algorytmy te starają się znaleźć naturalne struktury i wzorce w zestawie danych, aby identyfikować grupy obiektów (rekordów), które są bardziej podobne do siebie wzajemnie niż do obiektów w innych grupach [14].
Do metod klasteryzacji - uczenia nienadzorowanego, zaliczamy:

- metody hierarchiczne
- metody k-Means
- estymacja gęstości (Gaussian Mixture Model, HDBSCAN)

Techniki klasyfikacji należą do metod uczenia maszynowego nadzorowanego. Algorytmy przypisują dane do określonych grup na podstawie wcześniej ustalonych etykiet. Wykorzystują one wcześniej wyznaczone dane treningowe, aby nauczyć się modelu klasyfikacyjnego i móc przewidzieć przynależność nowych danych do wyznaczonych klas [15].
Do metod klasyfikacji - uczenia nadzorowanego, należą:

- maszyny wektorów nośnych (SVM)
- drzewa decyzyjne
- sieci neuronowe
- naiwny klasyfikator Bayesa

Wymienione przykłady technik grupowania, coraz częściej próbuje się adaptować do zastosowań w analizie danych uzyskanych z pomiarów przy użyciu cytometrii przepływowej. Najczęściej dotyczy to metod klasteryzacji, takich jak k-Means oraz BDSCAN. Dostępny on-line program Floreada.io (https://floreada.io/analysis) [16] oferuje możliwość zastosowania zamiast manualnego oznaczania grup populacji komórek, tych dwóch algorytmów uczenia nienadzorowanego. Program ten jest bezpłatny. Niektóre z modyfikacji metod grupowania są udostępniane tylko w programach komercyjnych jak np. FlowJo [17].

Są też podejmowane próby stosowania innych technik, na przykład takich jak redukcja wymiarowości. Największym powodzeniem cieszą się metody PCA oraz tSNE.

2. Założenia i cele pracy

Znaczenie cytometrii przepływowej we współczesnej medycynie ciągle wzrasta. Jest ona wykorzystywana zarówno w diagnostyce jak i w monitorowaniu skuteczności terapii oraz tzw. podstawowych badaniach biomedycznych.
Jednak trudności w prawidłowej analizie uzyskanych pomiarów, opracowywanie nowych metod opartych na cytometrii przepływowej i jej powszechność, stwarza konieczność z jednej strony ujednolicenia i stworzenia standardów analizy, a z drugiej zautomatyzowania tego procesu.

Celem pracy jest przyjrzenie się podstawowym algorytmom grupowania zaliczanym do metod uczenia nienadzorowanego, porównania ich skuteczności i możliwości zastosowania do automatyzacji analizy plików cytometrycznych.

3. Materiał i metody

3.1 Pochodzenie użytych plików cytometrycznych

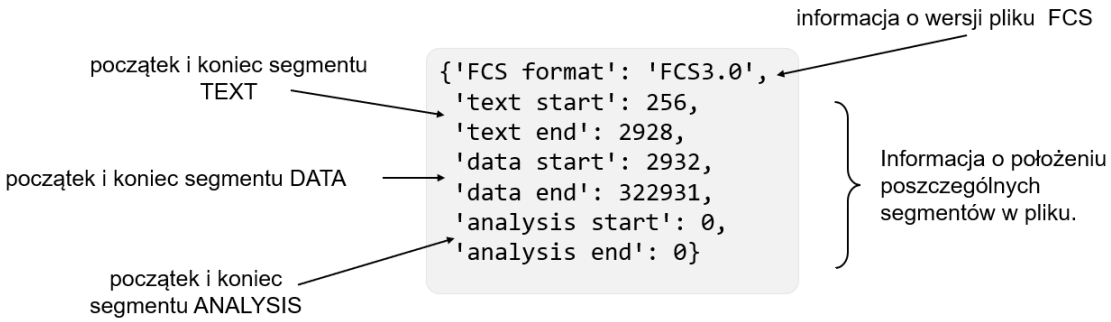
Użyte pliki cytometryczne powstały przy realizacji grantu MINIATURA 2 (NCN) pt. "Wzajemne relacje apoptozy i autofagii limfocytów T krwi obwodowej w zaburzeniach depresyjnych" (kierownik projektu: dr n. med. Krzysztof Pietruczuk). Pomiary były wykonane w pracowni Sieci Obrazowania Patologii Strukturalnej i Czynnościowej Komórek, Gdańskiego Uniwersytetu Medycznego działającej przy Katedrze i Zakładzie Fizjopatologii pod kierownictwem prof. dr hab. n. med. Jacka Witkowskiego. W celu ochrony danych osobowych i zapewnienia prywatności uczestników badania, pliki zostały zanonimizowane.
Pomiary były wykonane na cytometrze firmy Becton Dickinson FACSVerse, a uzyskane pliki to pliki cytometryczne FCS standardu 3.0.

3.2 Opis flow cytometry standard 3.0

Pliki FCS to specjalny znormalizowany format pliku zawierający dane tekstowe, po których następują dane binarne. Został stworzony po raz pierwszy w 1984 roku (FCS 1.0) przez ISAC i od tego czasu jest stale ulepszany [11].
Specyfikacja najnowszej wersji FCS 3.2 została opublikowana w 2020 roku. Plik z danymi pomiarów cytometrycznych podzielony jest na części zawierające poszczególne partie danych, które nazywane są segmentami. Główne segmenty danych wyróżnione w pliku to: NAGŁÓWEK, TEKST, DATA, ANALIZA oraz inne zdefiniowane przez osobę dokonującą pomiaru [18].

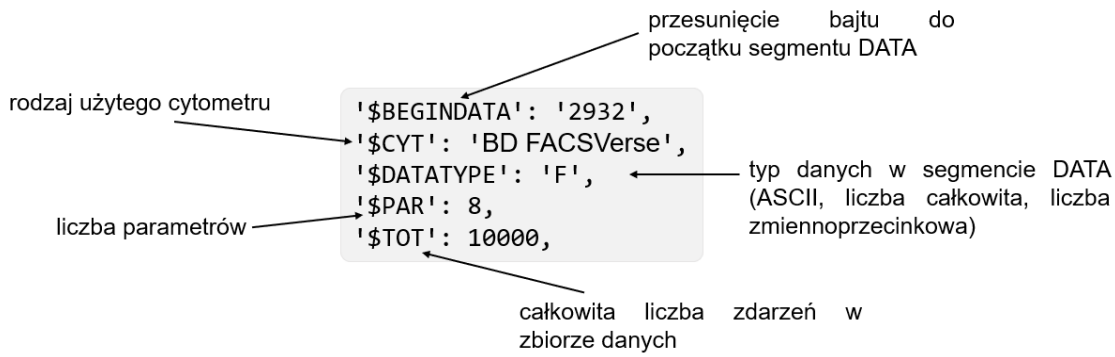
HEADER	-	obowiązkowy segment na początku pliku, zakodowany w formacie ASCI
TEXT	-	warunki eksperymentu
DATA	-	surowe dane pomiarów
ANALYSIS	-	ewentualny zapis analizy

Segment **HEADER** ma stały rozmiar - 58 bajtów i opisuje położenie innych segmentów w zbiorze danych [18].



Rys.3 Przykładowa zawartość sekcji HEADER z pliku FCS.

Segment **TEXT** jest podzielony na podsegmenty podstawowe i uzupełniające, zawierające serię par: słowo kluczowe + wartość, które opisują różne aspekty danych i warunki, w jakich przeprowadzono eksperyment. Segment ten musi zawierać wszystkie wymagane słowa kluczowe, które opisują format i kodowanie segmentu DATA [18].



Rys.4 Przykładowa zawartość sekcji TEXT z pliku FCS.

Segment **DATA** zawiera surowe dane, składające się z szeregu pomiarów natężenia światła dla każdej przechodzącej komórki. Surowe dane są przechowywane w jednym z czterech dozwolonych formatów (liczba całkowita, liczba zmiennoprzecinkowa, zmiennoprzecinkowa podwójnej precyzji lub ASCII). Dane zapisywane są w postaci tablicy, z kanałami fluorescencji i rozproszenia reprezentowanymi w kolumnach, a poszczególne zdarzenia (mierzone komórki) tworzą rzędy. Liczba zdarzeń uzyskanych z każdej próbki zwykle waha się od kilku tysięcy do kilku milionów [18].

	FSC-A	SSC-A	FITC-A	PE-A	PerCP-A	APC-A	V450-A	Time
0	262143.000000	124636.554688	2712.122803	1202.236206	881.494446	16.462471	1277.836792	1.0
1	196034.203125	91812.882812	313.105072	408.018463	137.460770	65.849884	93.827179	5.0
2	111004.945312	32835.667969	328.378510	79.639969	96.004349	1.936761	80.423294	6.0
3	120466.593750	22813.033203	233.465118	261.830048	3083.048584	-5.810284	92.933586	20.0
4	189443.281250	84884.203125	2082.639648	799.672546	973.134949	30.988182	230.546783	26.0
...	...	...	...	...	...	...	...	...
9995	135218.390625	36276.550781	282.558258	160.370895	225.828400	18.399233	54.509125	40231.0
9996	262143.000000	202469.890625	4306.013184	2265.920654	1183.689941	40.671989	29322.333984	40236.0
9997	123851.968750	22182.458984	146.188431	126.551186	3041.592285	86.185883	99.188728	40238.0
9998	188674.140625	95128.304688	2384.835205	754.943237	1023.319031	40.671989	587.090088	40246.0
9999	143112.328125	49977.898438	252.011414	279.285370	170.189529	-22.272757	630.876099	40247.0

10000 rows × 8 columns

Rys.5 Przykładowa zawartość sekcji DATA z pliku FCS w postaci ramki danych.

3.3 Odczyt i przygotowanie danych pomiarowych

Aby móc poddać analizie pliki cytometryczne zapisane w standardzie FCS, stosując techniki klasteryzacji oraz techniki klasyfikacji z użyciem bibliotek języka Python należy je wstępnie przygotować. Pierwszym etapem jest ich odczyt i przepisanie danych zawartych w segmencie DATA do ramki danych (DataFrame).

Odczyt pliku FCS

Do odczytu plików cytometrycznych w języku Python zostało stworzonych kilka bibliotek. W pracy została użyta biblioteka readfcs. **Readfcs** to pakiet Pythona typu open source, pozwalający na załadowanie danych do obiektów DataFrame co umożliwia ich analizę za pomocą innych narzędzi analitycznych dostarczanych przez ten język programowania [19].

```
1 import readfcs                                          # biblioteka do odczytu plików fcs
2 phenotype = readfcs.read(r'Pliki_fenotyp\Tube_001.fcs')  # wczytanie pliku z danymi pomiaru
3 phenotype.var                                           # informacje o danych
```

	n	channel	marker	$PnB	$PnE	$PnV	$PnR
FSC-A	1	FSC-A		32	0,0	209	262144
SSC-A	2	SSC-A		32	0,0	331	262144
CD7 FITC-A	3	FITC-A	CD7 FITC-A	32	0,0	431	262144
HLA-DR PE-A	4	PE-A	HLA-DR PE-A	32	0,0	415	262144
CD4 PerCP-A	5	PerCP-A	CD4 PerCP-A	32	0,0	508	262144
APC-A	6	APC-A	APC-A	32	0,0	440	262144
CD8 V450-A	7	V450-A	CD8 V450-A	32	0,0	462	262144
Time	8	Time		32	0,0		262144

Rys.6 Przykład kodu odczytu pliku FCS oraz wypisania jego zawartości. Wynik działania kodu.

W efekcie powyższego kodu otrzymujemy informację o warunkach przeprowadzonego pomiaru między innymi ilość kanałów, zastosowane fluorochromy, ustawienia aparatu.

Po zapoznaniu się z podstawowymi danymi o przeprowadzonym pomiarze dane z segmentu DATA będące odczytami detektorów zapisano w ramce danych i wyświetlono podstawowe informacje o niej.

```
meta, data = readfcs.view(r'Pliki_fenotyp\Tube_001.fcs')  # wczytanie pliku fcs
phenotype = data                                          # ramka danych z wynikami pomiarów
phenotype.info()                                          # informacja o ramce danych
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   FSC-A    10000 non-null  float32
 1   SSC-A    10000 non-null  float32
 2   FITC-A   10000 non-null  float32
 3   PE-A     10000 non-null  float32
 4   PerCP-A  10000 non-null  float32
 5   APC-A    10000 non-null  float32
 6   V450-A   10000 non-null  float32
 7   Time     10000 non-null  float32
dtypes: float32(8)
memory usage: 312.6 KB
```

Rys.7 Zapisanie do ramki danych 'phenotype' sekcji DATA i wypisanie podstawowych informacji o utworzonej ramce danych.

Przygotowanie danych

Ponieważ w pracy ma zostać zademonstrowane zastosowanie róznych metod analizy danych opartych na ich grupowaniu w celu okreslenia fenotypu komórek PBMC, zdecydowano się ograniczyć do analizy wyników otrzymanych na detektorach FSC i SSC oraz detektorach mierzących fluorescencję barwników V450 i PerCP wykorzystanych do identyfikacji populacji limfocytów CD4+ (tzw. pomocniczych) oraz CD8+ (tzw. cytotoksycznych). Z tego powodu pozostawiono w ramce danych jedynie kolumny zawierające wyniki tych pomiarów. Zmieniono także ich nazwy na krótsze.

W cytometrii przepływowej dokonuje się pomiarów rozproszenia światła oraz fluorescencji dlatego przy prawidłowym pomiarze nie będą występować wartości ujemne. Z ramki danych usunięto rekordy z wartościami poniżej 0.

```
phenotype.drop(columns={'Unnamed: 0',                    # usunięcie niepotrzebnych kolumn
                        'Time',
                        'FITC-A',
                        'PE-A',
                        'APC-A'}, inplace=True)

phenotype.rename( columns={'FSC-A': 'FSC',               # zmiana nazw reszty kolumn
                           'SSC-A': 'SSC',
                           'V450-A': 'CD8',
                           'PerCP-A': 'CD4'}, inplace=True )

phenotype.head()                                         # sprawdzenie rezultatu
```

	FSC	SSC	CD4	CD8
0	262143.000	124636.555	881.49445	1277.836800
1	196034.200	91812.880	137.46077	93.827180
2	111004.945	32835.668	96.00435	80.423294
3	120466.590	22813.033	3083.04860	92.933586
4	189443.280	84884.200	973.13495	230.546780

Rys.8 Usunięcie niepotrzebnych kolumn oraz nadanie nowych nazw pozostałym.

```
phenotype = phenotype[~(phenotype <= 0).any(axis=1)]     # usunięcie wierszy zawierających
                                                         # wartości <=0 z ramki danych
phenotype.describe().applymap('{:,.2f}'.format)          # wypisanie podstawowych statystyk
```

	FSC	SSC	CD4	CD8
count	9,359.00	9,359.00	9,359.00	9,359.00
mean	141,173.21	52,818.70	1,347.20	3,523.99
std	41,518.78	46,020.79	1,468.74	9,214.48
min	63,269.17	10,783.03	1.09	0.89
25%	117,769.49	28,346.37	136.37	121.53
50%	127,719.20	34,072.82	642.57	228.76
75%	150,531.77	53,766.80	2,576.30	1,029.87
max	262,143.00	262,143.00	25,356.06	252,292.34

Rys.9 Usunięcie wartości ujemnych oraz wypisanie podstawowych statystyk pomiarów.

Na koniec usunięto wartości odstające ustalone na podstawie percentyli. Jako próg dla wartości odstających wybrano 1 percentyl.

```python
columns = phenotype.columns.tolist()                    # lista nazw kolumn z 'phenotype'

outliers = pd.DataFrame(columns=['Below',               # ramka danych 'outliers'
                                 'Above',
                                 '% Below',
                                 '% Above',
                                 'Lower_limit',
                                 'Upper_limit'], index=columns)

total_records = len(phenotype)                          # całkowita ilości rekordów

for col in columns:
    upper_limit = phenotype[col].quantile(.99)          # wartości danych odstających
    lower_limit = phenotype[col].quantile(.01)

    below_count = (phenotype[col] < lower_limit).sum()  # ilość wartości odstających
    above_count = (phenotype[col] > upper_limit).sum()

    below_percent = (below_count / total_records) * 100 # procent wartości odstających
    above_percent = (above_count / total_records) * 100

    outliers.at[col, 'Below'] = below_count             # zapisywanie wyników
    outliers.at[col, 'Above'] = above_count
    outliers.at[col, '% Below'] = below_percent
    outliers.at[col, '% Above'] = above_percent
    outliers.at[col, 'Lower_limit'] = lower_limit
    outliers.at[col, 'Upper_limit'] = upper_limit

outliers.style.format('{:,.2f}')                        # wypisanie danych z 'outliers'
```

	Below	Above	% Below	% Above	Lower_limit	Upper_limit
FSC	93.00	0.00	0.99	0.00	73,921.86	262,143.00
SSC	94.00	0.00	1.00	0.00	18,362.27	262,143.00
CD4	86.00	94.00	0.92	1.00	8.73	4,798.54
CD8	93.00	94.00	0.99	1.00	11.62	36,294.09

Rys.10 Ustalenie wartości, ilości i procentu występowania danych odstających charakterystycznych dla poszczególnych kolumn ramki danych 'phenotype' i wypisanie tych informacji w postaci ramki danych 'outliers'.

Na podstawie zebranych obliczeń w ramce danych 'outliers' podjęto decyzję o usunięciu niektórych rekordów których wartości nie znajdowały się w określonym przedziale.

```python
for col in columns:                 # pobieranie wartości odstających dla danej kolumny
    lower_limit = outliers.at[col, 'Lower_limit']
    upper_limit = outliers.at[col, 'Upper_limit']
                                    # usuwanie rekordów z wartościami odstającymi
    phenotype.drop(phenotype[phenotype[col] < lower_limit].index, inplace=True)
    phenotype.drop(phenotype[phenotype[col] > upper_limit].index, inplace=True)
                                    # dodatkowe usunięcie górnych wartości odstających
phenotype.drop(phenotype.loc[phenotype['FSC'] > 260000].index, inplace=True)
phenotype.drop(phenotype.loc[phenotype['SSC'] > 260000].index, inplace=True)
                                    # podstawowe statystyki po usunięciu wartości odstających
phenotype.describe().applymap('{:,.2f}'.format)
```

	FSC	SSC	CD4	CD8
count	8,414.00	8,414.00	8,414.00	8,414.00
mean	135,953.85	45,720.83	1,305.38	2,629.41
std	30,939.87	29,842.56	1,379.29	6,655.95
min	73,921.86	18,364.10	8.73	11.62
25%	117,927.55	28,286.64	132.01	121.53
50%	127,099.92	33,578.61	589.66	216.25
75%	142,865.75	46,105.54	2,592.94	765.81
max	258,555.70	248,063.23	4,792.58	36,291.46

Rys.11 Usunięcie wartości odstających i wypisanie podstawowych statystyk ostatecznie przygotowanej ramki danych do analizy.

3.4 k-Means

Jednym z pierwszych algorytmów klasteryzacji, który próbowano i nadal próbuje się stosować do analizy danych FCM jest grupowanie k-Means [20]. Jest to algorytm iteracyjny, polegający na wyszukiwaniu punktów o cechach wspólnych z punktem znajdującym się w tzw. centrum. Punkty te określa się mianem centroidów. Dane leżące najbliżej centroidów algorytm grupuje tworząc klastry danych. W algorytmie tym istotne jest pojęcie metryki czyli odległości, którą można różnie definiować. Najczęściej rozumie się ją, jako najmniejszą sumę odległości pomiedzy centroidami a daną obserwacją (rekordem) [21].

Popularną metryką jest metryka euklidesowa, mająca zastosowanie w przestrzeni dwuwymiarowej, będąca przybliżeniem spostrzegania przestrzeni przez człowieka [22].

$$d_e(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Kolejne kroki algorytmu k-Means [20, 21]:

1. Wybór klastrów.
2. Obliczenie odległości punktów od centroidów.
3. Przypisanie każdemu z centroidów odpowiednich punktów.
4. Przeliczenie na nowo położenia centroidów.

Algorytm zakończy swoje działanie w momencie kiedy współrzędne położenia centroidów nie będą zmieniać się w stosunku do ich wcześniejszych współrzędnych lub zostanie spełniony warunek stopu zdefiniowany przez użytkownika.

W pracy do implementacji algorytmu k-Means zastosowano bibliotekę scikit-learn. Oferuje ona ulepszoną wersję tego algorytmu tzw. k-Means++, który osiąga zbieżność szybciej i dokładniej. Różnica polega na inicjalizacji kolejnych centroidów. W algorytmie k-Means++ są one wybierane z wykorzystaniem prawdopodobieństwa proporcjonalnym do kwadratu odległości od najbliższego już wybranego centroidu [23]. W przypadku podstawowej wersji tej metody wykorzystywana jest średnia.
Przy analizie wyników z detektorów FSC i SSC, liczbę klastrów zdefiniowano na 3, zgodnie z eksperymentalnymi danymi naukowymi wskazujacymi na istnienie trzech podstawowych populacji komórek PBMC wyróżnianych pod kątem wielkości i struktury (ziarnistości) charakterystycznych dla tych grup. Użyto w tym przypadku następującego etykietowania: debris, lymphocytes oraz monocytes.
Zostały również zdefiniowane początkowe współrzędne centroidów dla trzech klastrów w celu przyspieszenia i stabilizacji działania algorytmu.

```python
from sklearn.cluster import KMeans                          # import KMeans z biblioteki sklearn
from matplotlib.colors import ListedColormap                # import funkcji określającej kolory

initial_centroids = [[70000, 20000],                        # pozycje początkowe centroidów
                     [125000, 40000],
                     [200000,  100000]]

kmeans = KMeans(n_clusters = 3,                             # podstawowe parametry KMeans
                init = initial_centroids,
                n_init = 1,
                max_iter = 300,
                random_state = 42)

kmeans.fit(phenotype[['FSC','SSC']])

labels = kmeans.predict(phenotype[['FSC','SSC']])           # przewidywanie przynależności

centroids = kmeans.cluster_centers_

centroid = pd.DataFrame(centroids, columns = ['FSC','SSC'])
label = pd.DataFrame(labels, columns = ['label'])


colors = ['#08d8ff', '#0a5da3','#102b52']                   # lista kolorów klastrów
cmap = ListedColormap(colors)
                                                            # wyświetlenie wykresu FSC/SSC
plt.figure(figsize=(7, 5))
plt.scatter(phenotype['FSC'], phenotype['SSC'], alpha=0.3, s=area, c=labels, cmap=cmap)
plt.scatter(centroid['FSC'],centroid['SSC'], marker='o', color='red', s=50)

plt.gca().set_facecolor('#F5F9FD')
plt.xticks([50000,100000,150000,200000,250000])             # ustawienie etykiet dla skali osi
plt.yticks([0,50000,100000,150000,200000,250000])
plt.title('K-Means Clustering (PBMC)', fontsize=14, fontweight='bold')
plt.xlabel('FSC - cell size', fontsize=13)
plt.ylabel('SSC - cell granularity', fontsize=13)
legend_labels = ['Debris', 'Lymphocyte', 'Monocytes']       # legenda wykresu
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=color, markersize=10)
    for color in colors]
plt.legend(legend_elements, legend_labels, loc='upper right')
plt.show()
```

Rys.12 Kod w Python z zastosowaniem algorytmu k-Means z użyciem biblioteki scikit-learn. Zdefiniowanie położenia początkowych centroidów, nadanie klastrom kolorów w celu ich rozróżnienia na wykresie. Wykres utworzony z wykorzystaniem biblioteki matplotlib.

3.5 HDBSCAN

Algorytm HDBSCAN (ang. Hierarchical Density-Based Spatial Clastering of Applications with Noise) jest algorytmem klasteryzacji opartym na gęstości, czyli grupuje punkty (dane), które są ściśle związane ze sobą w przestrzeni, a oddziela te znajdujące się od siebie daleko. Identyfikuje również wartości odstające oraz szumy nadając im etykietę punktów nieprzypisanych. W przeciwieństwie do algorytmu k-Means nie definiuje się w nim klastrów. HDBSCAN dostosowuje się do różnych gęstości identyfikując klastry o róznych kształtach [24].

Działanie algorytmu HDBSCAN [25]:

1. Przekształcenie przestrzeni danych zgodnie z gęstością.
2. Budowa minimalnego drzewa rozpinającego punktów danych z wykorzystaniem wag i sortowanie jego krawędzi od najdłuższej do najkrótszej.
3. Kostruowanie hierarchicznego drzewa klastrów łacząc punkty leżące blisko siebie. W powstałym dendrogramie wysokość to gęstość pomiędzy punktami.
4. Konstruowanie klastrów w oparciu o minimalny rozmiar klastra poprzez likwidację najdłuższych krawędzi drzewa, reprezentujących przejścia między klastrami.
5. Wyodrębnianie stabilnych klastrów z drzewa.

W algorytmie HDBSCAN należy określić minimalną liczbę punktów potrzebną do utworzenia klastra.

W pracy liczbę tę ustawiono na 260 oraz w przypadku analizy parametrów FSC i SSC podano etykiety: unclassified, Monocytes, Lymphocytes. Do implementacji algorytmu HDBSCAN zastosowano bibliotekę scikit-learn.

```python
import hdbscan                                                  # import funkcji HDBSCAN
from matplotlib.colors import ListedColormap

clusterer = hdbscan.HDBSCAN(min_cluster_size=260,               # definiowanie algorytmu
                            cluster_selection_epsilon=0.0)

                                                                # dopasowanie modelu
cluster_labels = clusterer.fit_predict(phenotype[['FSC', 'SSC']])

                                                                # ramka danych z wynikami
phenotype_HDBSCAN = phenotype[['FSC', 'SSC']].copy()
phenotype_HDBSCAN['cluster'] = cluster_labels

                                                                # mapowanie etykiet klastrów
cluster_names = {0: 'unclassified', 1: 'Monocytes', 2: 'Lymphocytes'}
phenotype_HDBSCAN['cluster_name'] = phenotype_HDBSCAN['cluster'].map(cluster_names)

cmap = ListedColormap(['#605C5C','#57ABFF','#0064C8'])          # lista kolorów klastrów

                                                                # wykres wyników klasteryzacji
plt.figure(figsize=(7, 5))
scatter = plt.scatter(phenotype_HDBSCAN['FSC'], phenotype_HDBSCAN['SSC'],
                      c=phenotype_HDBSCAN['cluster'],
                      s=5, alpha=0.5, cmap=cmap)
plt.gca().set_facecolor('#F5F9FD')
plt.xticks([50000,100000,150000,200000,250000])
plt.yticks([0,50000,100000,150000,200000,250000])
plt.xlabel('FSC - cell size', fontsize=13)
plt.ylabel('SSC - cell granularity', fontsize=13)
plt.title('HDBSCAN clustering (PBMC)', fontsize=14, fontweight='bold')

plt.legend(handles=scatter.legend_elements()[0],                # legenda wykresu
           labels=list(cluster_names.values()), l
           oc="upper right",
           title='Cell populations')

plt.show()

```

Rys.13 Kod w Python z zastosowaniem algorytmu HDBSCAN. Zdefiniowanie minimalnej ilości zdarzeń dla klastrów na 260. Nadanie nazw dla poszczególnych grup jako unclassified, Monocytes, Lymphocytes i odpowiadających im kolorów w celu rozróżnienia na wykresie. Wykres utworzony z wykorzystaniem biblioteki matplotlib.

3.6 Gaussian Mixture Model

Gaussian Mixture Model (GMM) jest modelem probabilistycznym w którym zakłada się że dane stanowią skończoną ilość rozkładów Gaussa o nieznanych parametrach [26]. Stano on powszechną metodę uczenia nienadzorowanego wykorzystywaną do grupowania.
Model ten dzieli dane na subpopulacje o rozkładzie normalnym. Miarą tego jak silnie punkty (dane) są ze sobą powiązane jest macierz kowariancji [27]. W metodzie GMM dokonuje się szacunku parametrów rozkładu gaussowskiego takich jak średnia, wariancja oraz waga klastra na podstawie których obliczane jest prawdopodobieństwo przynależności punktu do danej populacji [28].

Etapy działania algorytmu GMM [28]:

1. Zainicjowanie średnich μ, kowariancji cov oraz wag w
2. Iteracja do osiągnięcia zbieżności

   Krok E (krok oczekiwań)

   Obliczenie wartości oczekiwanych dla punktu danych przy przyjętych parametrach modelu μ, cov, w.

   Krok M (krok maksymalizacji)

   Aktualizacja wartości μ, cov, w.

   Zbieżność

   Ocena prawdopodobieństwa, sprawdzenie zbieżności, podjęcie decyzji o ewentualnym zakończeniu iteracji.

Do implementacji w Pythonie modelu GMM zastosowano bibliotekę scikit-learn. W postaci listy zostały zdefiniowane początkowe punkty centroidów w celu usprawnienia działania algorytmu i określono liczbę klastrów. Poszczególnym klastrom nadano etykiety: Debris, Lymphocytes, Monocytes. Ustalono progi prawdopodobieństwa dla każdej z grup uzyskując w ten sposób dodatkową subpopulację punktów niesklasyfikowanych, uzyskując kontrolę nad stopniem rozmycia klastrów.

```python
from sklearn.mixture import GaussianMixture                    # import algorytmu GMM
from matplotlib.colors import ListedColormap

initial_centroids = [[70000, 20000],                           # początkowe centroidy
                     [125000, 40000],
                     [200000,  100000]]

gmm = GaussianMixture(n_components=3,                          # parametry użycia GMM
                      means_init=initial_centroids,
                      covariance_type='full',
                      random_state=0)

gmm.fit(phenotype[['FSC','SSC']])                              # dopasowanie modelu

probs = gmm.predict_proba(phenotype[['FSC','SSC']])            # prawdopodobieństwo
                                                               # przynależności punktów

thresholds = [0.9, 0.35, 0.8]                                  # progi prawdopodobieństwa

labels = np.full(probs.shape[0], 3)                            # przypisanie punktów
for i, threshold in enumerate(thresholds):
    mask = (probs[:, i] >= threshold) & (labels == 3)
    labels[mask] = i

plt.figure(figsize=(7, 5))

cmap = ListedColormap(['#003D7A', '#57ABFF', '#0064C8', '#605C5C'])
plt.scatter(phenotype.FSC, phenotype.SSC, c=labels, s=5, alpha=0.5, cmap=cmap)

clusters = ['Debris','Lymphocytes', 'Monocytes']                # etykiety klastrów

for i in range(gmm.n_components):
    mean = gmm.means_[i]
    plt.text(mean[0], mean[1], clusters[i], color='#000000', ha='center',
             fontsize=10, fontweight='bold',
             bbox=dict(facecolor='#F5F9FD', alpha=0.5, edgecolor='none'))

plt.gca().set_facecolor('#F5F9FD')
plt.xticks([50000,100000,150000,200000,250000])
plt.yticks([0,50000,100000,150000,200000,250000])
plt.xlabel('FSC - cell size', fontsize=13)
plt.ylabel('SSC - cell granularity', fontsize=13)
plt.title('Gaussian Mixture Model (PBMC)', fontsize=14, fontweight='bold')
plt.show()
```

Rys.14 Kod w Python z zastosowaniem algorytmu GMM z biblioteki scikit-learn. Określenie współrzędnych początkowych centroidów. Nadanie nazw dla poszczególnych grup jako Debris, Monocytes, Lymphocytes i grupie niesklasyfikowanej odpowiadających im kolorów w celu rozróżnienia na wykresie. Wykres utworzony z wykorzystaniem biblioteki matplotlib.

3.7 Przygotowanie ramki danych z docelowym wynikiem klasteryzacji

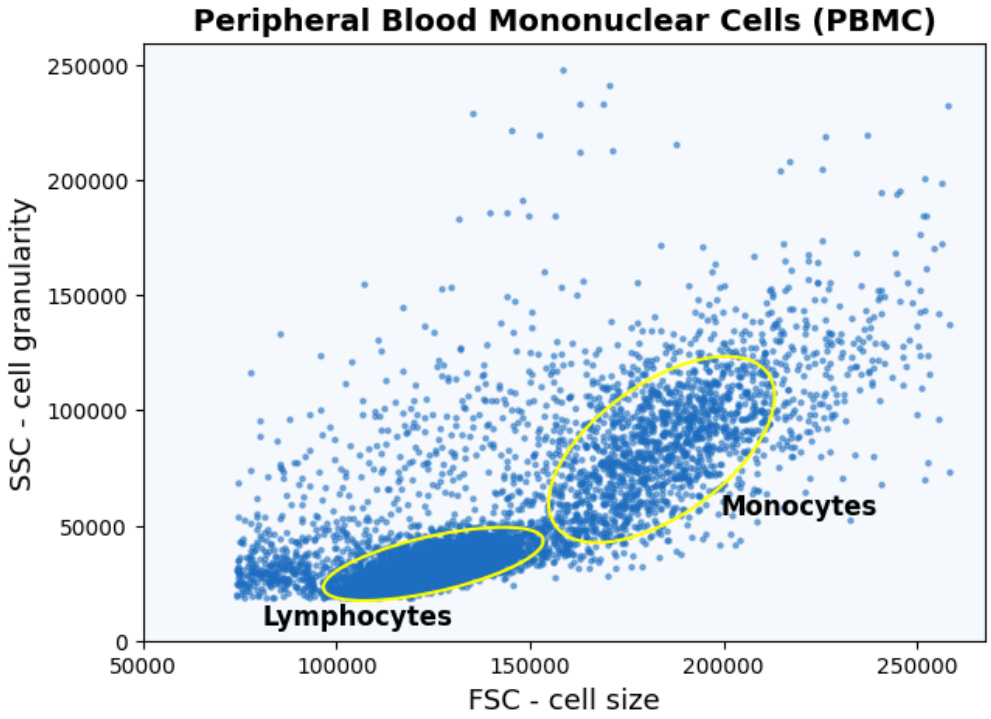
W celu umożliwienia przeprowadzenia późniejszej oceny niektórych parametrów skuteczności i stopnia poprawności użytych w pracy algorytmów klasteryzacji stworzono na podstawie bramkowania ręcznego ramkę danych z najbardziej porządanym wynikiem klasteryzacji. Użyto do tego jednego z plików cytometrycznych zawierającego 10000 pomiarów, który poddano również automatycznemu ustaleniu populacji komórek przez algorytmy uczenia nienadzorowanego.
Po odczytaniu i przygotowaniu pliku tak jak zostało to opisane w punkcie 3.3, został wygenerowany wykres zalezności pomiędzy parametrami FSC i SSC. Następnie naniesiono na wykres dwie elipsy. Jedna zaznaczająca miejsce gdzie doswiadczalnie została określona przestrzeń w której znajdują się limfocyty - wchodzące w skład komórek PBMC. Druga elipsa oznaczała miejsce występowania monocytów. Każdemu punktowi, który znalazł się wewnątrz elips przypisano odpowiednią etykietę. Ponieważ każdy punkt na wykresie reprezentuje badaną komórkę, która stanowi równocześnie jeden rekord w pliku, uzyskany wynik zapisano w ramce danych, której nadano nazwę 'truth_data'.

Zawarto w niej trzy kolumny:

'FSC' - wyniki pomiarów na detektorze FSC
'SSC' - wyniki pomiarów na detektorze SSC
'cluster' - przynależność danego rekordu (komórki) do jednej z trzech grup

Dany rekord mógł zostać zaliczony do grupy należącej do elips i otrzymać etykietę 'Lymphocytes' lub 'Monocytes', albo w przypadku jeżeli nie znalazł się w żadnej z tych grup otrzymywał etykietę 'unclassified'.

```python
from matplotlib.patches import Ellipse                    # import funkcji rysowania elipsy

      # ---- kod tworzenia wykresu zależności FSC/SSC --------------------------------

lymphocytes = Ellipse((125000, 33300), 61000, 23000,      # rysowanie elips limfocytów
                      angle=23, fill=False, lw=1.5,
                      edgecolor='yellow')
monocytes = Ellipse((184000, 83000), 90000, 43000,        # rysowanie elips monocytów
                      angle=60, fill=False, lw=1.5,
                      edgecolor='yellow')

truth_data = pd.DataFrame(data={'FSC': data['FSC'],       # tworzenie ramki truth_data
                                'SSC': data['SSC']})
truth_data['cluster'] = 'unclassified'                    # dodanie kolumny 'cluster'
                                                          # z wartościami 'unclassified'

for idx, row in truth_data.iterrows():                    # etykiety punktów w elipsach
    point = np.array([row['FSC'], row['SSC']])
    if lymphocytes.contains_point(point):
        truth_data.at[idx, 'cluster'] = 'Lymphocytes'
    elif monocytes.contains_point(point):
        truth_data.at[idx, 'cluster'] = 'Monocytes'

plt.gca().add_patch(lymphocytes)                          # dodanie elips do wykresu
plt.gca().add_patch(monocytes)
```



Rys.15 Kod w Python zastosowany do ręcznego bramkowania populacji limfocytów i monocytów. Wykres FSC/SSC z zaznaczonymi bramkami.

	FSC	SSC	cluster
1	196034.203125	91812.882812	Monocytes
2	111004.945312	32835.667969	Lymphocytes
3	120466.593750	22813.033203	Lymphocytes
4	189443.281250	84884.203125	Monocytes
5	128579.968750	40327.281250	Lymphocytes
6	124728.570312	34395.738281	Lymphocytes
7	109908.914062	48135.269531	unclassified
8	87501.835938	26779.757812	unclassified
9	150783.437500	80319.632812	unclassified
10	117842.445312	37765.710938	Lymphocytes

Rys.16 Pierwszych 10 rekordów utworzonej ramki danych 'truth_data'.

3.8 Metody walidacji zatosowanych algorytmów

Dla analizy skupień zostały opracowane ogólne techniki walidacji, które mozna wykorzystać niezaleznie od stosowanej metody grupowania. Metody walidacji dzielimy na miary wewnętrzne, zewnętrzne i oparte na stabilności.
Do miar wewnętrznych zaliczane są: Silhouette Score oraz Davies-Bouldin Index. Zastosowanie ich wymaga jedynie dostępu do danych, które podlegają grupowaniu i opierają się one na pojęciu odległości między poszczególnymi obserwacjami.
Miary zewnętrzne do których należą Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), Fowlkes-Mallows Index (FMI) oraz współczynnik Jaccarda wymagają istnienia pewnego wzorca grupowania do którego maja dążyć algorytmy wykorzystane w procesie klasteryzacji. Oceniany jest w ten sposób stopień dokładności metody [29].

Ocenie poddano zastosowane w pracy trzy metody klasteryzacji: k-Means++, HDBSCAN oraz GMM. Wyniki działania tych algorytmów na przykładowym pliku cytometrycznym, zostały odpowiednio zapisane w ramkach danych:

'kMeans_data', 'HDBSCAN_data', 'GMM_data'

W pracy oceniano zgodność wyniku z prawdziwym podziałem (ARI, NMI, FMI, Jaccard coefficient), jakość klasteryzacji (Silhouette score, Davies-Bouldin index) oraz czas obliczeń poszczególnych algorytmów.
Ocenę algorytmów przeprowadzono na komputerze z procesorem intel Core i7-10700 2.9GHz 8 rdzeni z pamięcią RAM 64GB, wyposażonym w system operacyjny Windows 10 Pro.

Czas obliczeń

Ponieważ czas w którym algorytm dokonuje podziału danych na grupy może mieć duży wpływ na jego przydatność w przypadku klasteryzacji dużych zestawów danych, postanowiono zmierzyć jak długo każda z użytych metod dokonuje klasteryzacji tych samych danych. Wynik stanowi średnia z 500 pomiarów czasu działania.

```
import time                                          # import biblioteki time

execution_times = []                                 # listy na wyniki pomiaru czasu

for i in range(500):                                 # pętla wykonująca się 500 razy
    start_time = time.time()                         # rozpoczęcie pomiaru czasu

    # -------  kod algorytmu klasteryzacji --------------------------

    end_time = time.time()                           # koniec pomiaru czasu
    execution_time = end_time - start_time           # obliczenie czasu działania algorytmu
    execution_times.append(execution_time)           # dodanie czasu wykonania do listy

avg = sum(execution_times) / len(execution_times)    # obliczenie średniego czasu działania

```

Rys.17 Kod Pythona pomiaru czasu działania algorytmu jako średnia z 500 jego uruchomień.

Miary zewnętrzne (ARI, NMI, FMI, Jaccard)

W celu użycia miar zewnętrzych do oceny algorytmów klasteryzacji najpierw zakodowano użyte w kolumnie 'claster' nazwy : Lymphocyte, Monocyte i unclassified do formatu numerycznego.

```python
from sklearn.preprocessing import LabelEncoder
                                # kodowanie etykiet 'truth_data' do formatu numerycznego
encoder = LabelEncoder()
truth_labels = encoder.fit_transform(truth_data['cluster'])

                                # kodowanie etykiet klas w 'kMeans_data',
                                # 'HDBSCAN_data' i 'GMM_data' do formatu numerycznego
kMeans_labels = encoder.transform(kMeans_data['cluster'])
hdbscan_labels = encoder.transform(HDBSCAN_data['cluster'])
gmm_labels = encoder.transform(GMM_data['cluster'])
```

Rys.18 Kod Python kodowania etykiet do wartości numerycznych.

Adjusted Rand Index (ARI)

ARI używany często w walidacji klastrów jest miarą zgodności między parami elementów pochodzących ze zbioru z określonymi etykietami grup oraz zbioru danych z etykietami nadanymi przez algorytm [30].

Miara Adjusted Rand Index w grupowaniu trzech klastrów: Tablica kontyngencji i matematyczne podstawy ARI:

		Algorytm		
		Lymphocytes	Monocytes	unclassified
Podział oczekiwany	Lymphocytes	$t_{11} = a$ (true)	$t_{12} = b$ (false)	$t_{13} = c$ (false)
	Monocytes	$t_{21} = d$ (false)	$t_{22} = e$ (true)	$t_{23} = f$ (false)
	unclassified	$t_{31} = g$ (false)	$t_{32} = h$ (false)	$t_{33} = i$ (true)

$$ARI = \frac{\binom{n}{2}(a+e+i) - [(a+b)(a+c)+(d+e)(d+f)+(g+h)(g+i)]}{\frac{1}{2}\left[\binom{a+b}{2}+\binom{c+d}{2}+\binom{e+f}{2}+\binom{g+h}{2}+\binom{i}{2}\right] - [(a+b)(a+c)+(d+e)(d+f)+(g+h)(g+i)]}$$

Określając wartość ARI dla poszczególnych algorytmów klasteryzacji posłużono się funkcją adjusted_rand_score z biblioteki Python scikit-learn. Wyniki zapisano w ramce danych 'score'.

```
from sklearn.metrics import adjusted_rand_score     # import funkcji ARI

                                                    # ARI dla k-Means, HDBSCAN i GMM
ari_kMeans = adjusted_rand_score(truth_labels, kMeans_labels)
ari_hdbscan = adjusted_rand_score(truth_labels, hdbscan_labels)
ari_gmm = adjusted_rand_score(truth_labels, gmm_labels)

score.loc['k-Means', 'ARI'] = ari_kMeans            # dopisanie indeksu ARI do 'score'
score.loc['HDBSCAN', 'ARI'] = ari_hdbscan
score.loc['GMM', 'ARI'] = ari_gmm
```

Rys.19 Kod Python wyliczający wartość ARI dla stosowanych w pracy trzech metod klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

Metryka ARI może przybierać wartości od -1 do 1. Wartość 1 oznacza idealne dopasowanie natomiast 0 mówi o przypisaniu losowym. Jeżeli współczynnik ARI mieści się pomiędzy zakresami 0-1 mówimy o częściowej zgodności. Im wyższa wartość ARI tym lepsze dopasowanie pomiędzy uzyskanym podziałem a oczekiwanym wynikiem klasteryzacji. Wyniki poniżej zera mówią o dopasowaniu gorszym niż dopasowanie losowe.

Normalized Mutual Information (NMI)

Normalized Mutual Information jest miarą opartą na teorii informacji używaną do oceny klasteryzacji, w celu określenia poprawności przypisania danych do klastrów w porównaniu z rzeczywistymi etykietami.
Wartości NMI przybierają zakres między 0 a 1. Osiągnięcie wartości 1 oznacza doskonałą zgodność podziału zbiorów, natomiast 0 brak jakiegokolwiek podobieństwa [31, 32]. Jest to równoznaczne ze stwierdzeniem, iż im wyższa wartość NMI tym lepiej algorytm dokonuje klasteryzacji w porównaniu do wartości rzeczywistych.

Etapy wyliczania metryki NMI [26, 31, 32]:

1. Konstrukcja tablicy kontyngencji (tak jak w przypadku metryki ARI).
2. Obliczenie wzajemnej informacji między dwoma zbiorami danych (MI - mutual information), czyli miary podobieństwa między klastrami zbiorów.
3. Wyliczenie entropi zbiorów.
4. Podstawienie obliczonych wartości do wzoru i wyliczenie wskaźnika NMI.

Ogólny wzór na miarę NMI [26]:

$$NMI(A,B) = \frac{MI(A,B)}{avg(H(A),H(B))}$$

- $MI(A,B)$ - wzajemna informacja między dwoma podziałami, podziałem rzeczywistym (referencyjnym) a podziałem dokonanym przez algorytm klasteryzacyjny
- $H(A)$ - entropia podziału A
- $H(B)$ - entropia podziału B
- $avg(H(A),H(B))$ - średnia z wyliczonych entropi obu podziałów

Ponieważ w bibliotece scikit-learn jako średnią przyjęta jest średnia arytmetyczna dlatego powyższy wzór w tym przypadku przyjmuje ostatecznie postać:

$$NMI(A,B) = \frac{2 * MI(A,B)}{H(A) + H(B)}$$

W pracy posłużono się funkcją normalized_mutual_info_score z biblioteki Python scikit-learn w celu ustalenia wartości NMI dla zastosowanych algorytmów. Wyniki zapisano w ramce danych 'score'.

```
from sklearn.metrics import normalized_mutual_info_score              # import funkcji NMI

                                          # obliczenie Normalized Mutual Information dla
                                          # wyników klasteryzacji k-Means, HDBSCAN i GMM
nmi_kMeans = normalized_mutual_info_score(truth_labels, kMeans_labels)
nmi_hdbscan = normalized_mutual_info_score(truth_labels, hdbscan_labels)
nmi_gmm = normalized_mutual_info_score(truth_labels, gmm_labels)

score.loc['k-Means', 'NMI'] = nmi_kMeans    # dopisanie indeksu FMI do ramki danych 'score'
score.loc['HDBSCAN', 'NMI'] = nmi_hdbscan
score.loc['GMM', 'NMI'] = nmi_gmm
```

Rys.20 Kod Python wyliczający wartość NMI dla stosowanych w pracy trzech metod klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

Fowlkes-Mallows Index (FMI)

Indeks FMI jest definiowany jakośrednia geometryczna dwóch miar jakości klasyfikacji jakimi są precyzja oraz czułość. Precyzja mówi jak wiele punktów przypisanych do klastra przez algorytm faktycznie do niego należy. Czułość ile rzeczywiści należących do kalstra punktów zostało poprawnie zidentyfikowanych [34].

Wzór na obliczenie indeksu FMI [26]:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

- TP - liczba punktów prawdziwie pozytywnych
- FP - liczba punktów fałszywie pozytywnych
- FN - liczba punktów fałszywie negatywnych

Uzyskanie wyniku 0 świadczy o całkowitym braku zgodności pomiędzy klastrami referencyjnymi a ustalonymi przez algorytm, wartość 1 mówi o całkowitej zgodności pomiędzy podziałami.
W pracy posłużono się funkcją z biblioteki Python scikit-learn w celu obliczenia wartości NMI dla zastosowanych algorytmów. Wyniki zapisano w ramce danych 'score'.

```python
from sklearn.metrics import fowlkes_mallows_score                    # import funkcji FMI

                                          # obliczenie Fowlkes-Mallows Index dla
                                          # wyników klasteryzacji k-Means, HDBSCAN i GMM
fmi_kMeans = fowlkes_mallows_score(truth_labels, kMeans_labels)
fmi_hdbscan = fowlkes_mallows_score(truth_labels, hdbscan_labels)
fmi_gmm = fowlkes_mallows_score(truth_labels, gmm_labels)

score.loc['k-Means', 'FMI'] = fmi_kMeans     # dopisanie indeksu FMI w ramce danych 'score'
score.loc['HDBSCAN', 'FMI'] = fmi_hdbscan
score.loc['GMM', 'FMI'] = fmi_gmm
```

Rys.21 Kod Python wyliczający wartość FMI dla stosowanych w pracy trzech metod klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

Jaccard Index

Jaccard Indeks, kolejna miara zewnętrzna, to tzw. indeks poprawności. Jest on wykorzystywany do pomiaru podobieństwa między dwoma podziałami danych [35]. Jaccard Indeks jest definiowany jako iloraz mocy części wspólnej zbiorów i mocy sumy tych zbiorów [35, 36]:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard Indeks przyjmuje wartości między 0 a 1 i podobnie jak w innych miarach zewnętrznych wynik 0 oznacza brak podobieństwa natomiast 1 pełne pokrycie pomiędzy klastrami określonymi przez algorytm a rzeczywistym podziałem zbioru.

Do określenia tego indeksu dla zastosowanych algorytmów wykorzystano funkcję zawartą w bibliotece scikit-learn Pythona jaccard_score [26]. Wyniki obliczeń zapisano w ramce danych 'score'.

```python
from sklearn.metrics import jaccard_score           # import funkcji jaccard indeks

truth_labels = truth_data['cluster']                # przypisanie etykiet do zmiennych
kmeans_labels = kMeans_data['cluster']
hdbscan_labels = HDBSCAN_data['cluster']
gmm_labels = GMM_data['cluster']
                                                    # obliczanie indeksu Jaccard dla metod
kmeans_jaccard = jaccard_score(truth_labels, kmeans_labels, average='macro')
hdbscan_jaccard = jaccard_score(truth_labels, hdbscan_labels, average='macro')
gmm_jaccard = jaccard_score(truth_labels, gmm_labels, average='macro')

score.loc['k-Means', 'Jaccard'] = kmeans_jaccard    # zapisanie indeksu Jaccard do 'score'
score.loc['HDBSCAN', 'Jaccard'] = hdbscan_jaccard
score.loc['GMM', 'Jaccard'] = gmm_jaccard
```

Rys.22 Kod Python który oblicza Jaccard Index dla badanych w pracy trzech metod klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

Miary wewnętrzne (Silhouette Score, Davies-Bouldin Index)

Silhouette Score

Index Silhouette należy do miar wewnętrzych oceny algorytmów klasteryzacji. Jego wartość jest obliczana na podstawie struktury samej klasteryzacji i jest niezależna od informacji zewnętrznych. Do wyznaczenia Silhouette Score jest używana średnia odległość wewnątrz klastra i średnia odległość do najbliższego klastra dla każdej obserwacji. Ostatecznie wyciągana jest średnia ze wszystkich obliczeń.

Wzór na współczynnik Silhouette Score dla pojedynczej próbki:

$$SilhouetteScore = \frac{y - x}{max(x, y)}$$

- x - średnia odległość wewnątrz klastra
- y - średnia odległość do najbliższego klastra

Współczynnik Silhouette przyjmuje wartości miedzy -1 a 1, gdzie wartości bliskie jedynki wskazują na dobrą jakość klasteryzacji.
Do wyliczeń indeksu Silhouette wykorzystano funkcję z biblioteki scikit-learn Pythona silhouette_score [26]. Wyniki obliczeń zapisano w ramce danych 'score'.

```
from sklearn.metrics import silhouette_score        # import funkcji silhouette_score

                    # ------ kod algorytmu klasteryzacji -------

                                                    # obliczenie wyniku Silhouette score
silhouette_avg = silhouette_score(data[['FSC','SSC']], labels)
score.loc['GMM', 'Silhouette'] = silhouette_avg     # zapisanie w ramce danych 'score'

```

Rys.23 Kod Python wyliczający Silhouette Score dla stosowanych w pracy trzech algorytmów klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

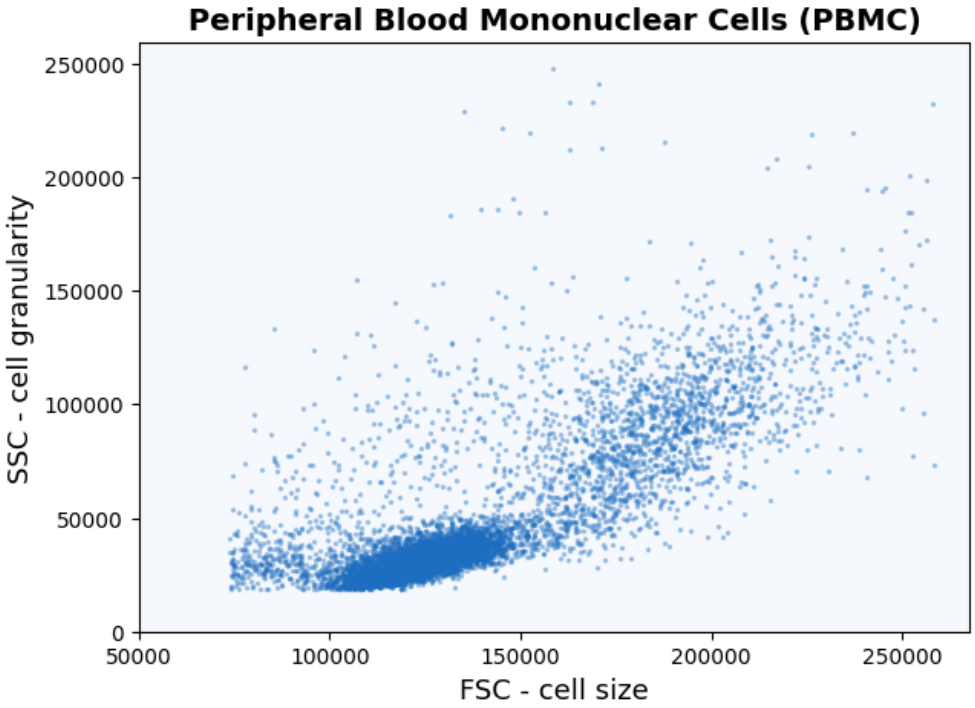
Davies-Bouldin Index

Miara ta sprawdza jak dobrze zostało wykonane grupowanie poprzez wyznaczenie sredniego prawdopodobieństwa porównując odległość między skupieniami z wielkością tych skupień. Na najlepszy podział wskazują wartości bliskie zeru.
Do wyliczeń użyto funkcję z biblioteki scikit-learn Pythona [26], a wyniki zapisano w ramce danych 'score'.

```
from sklearn.metrics import davies_bouldin_score     # import funkcji Daviesa-Bouldina

                    # ------ kod algorytmu klasteryzacji -------

                                                     # indeksu Daviesa-Bouldina
db_score = davies_bouldin_score(data[['FSC','SSC']], labels)
score.loc['GMM', 'Davies'] = db_score                # zapisanie wyniku w 'score'

```

Rys.24 Kod Python wyliczający indeks Daviesa-Bouldina dla stosowanych w pracy trzech algorytmów klasteryzacji pomiarów na detektorach FSC i SSC w przypadku komórek PBMC.

4. Wyniki badań



Rys.25 Wykres dot-plot przykładowego pliku z pomiaru cytometrycznego, uzyskany po wstępnym przygotowaniu danych pomiarowych. Rozkład populacji komórek PBMC w zależności od ich wielkości i ziarnistości. Wykres wykonano w języku Python przy użyciu biblioteki matplotlib. Widok bez stosowania metod klasteryzacji i klasyfikacji.

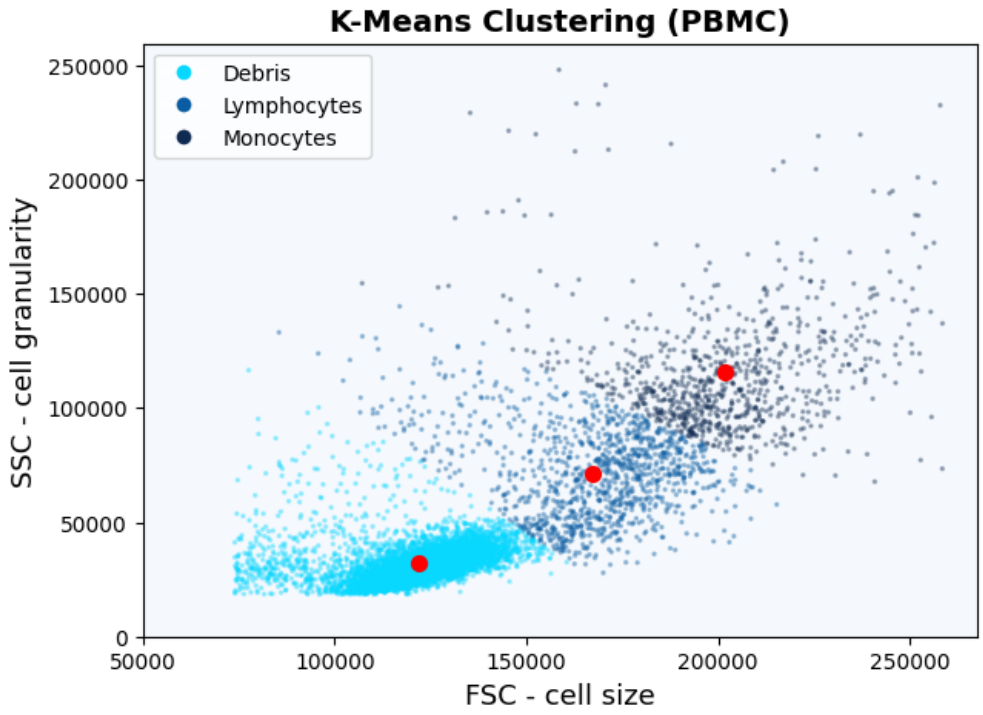
4.1 Otrzymane wyniki grupowania

k-Means

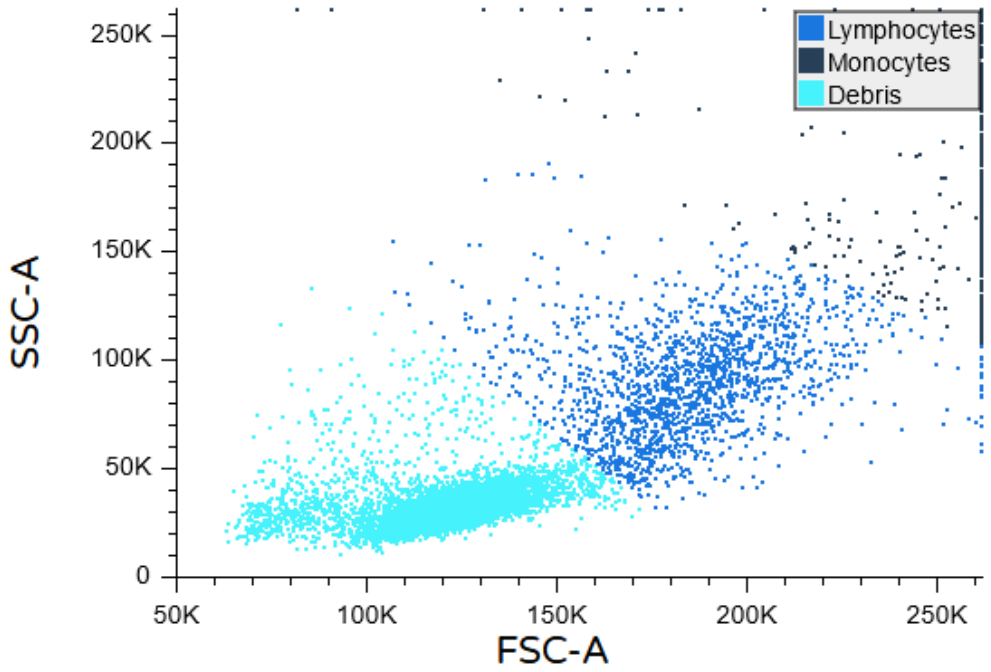
Z powodu najmniejszej złożoności oraz dużej popularności, jako pierwszą metodę klasteryzacji, sprawdzono jakość działania algorytmu k-Means w przypadku analizy danych cytometrycznych.

Możliwość analizy z wykorzystaniem k-Means oferuje również program Floreada.io dostępny bezpłatnie on-line. Podano klasteryzacji ten sam plik cytometryczny ze zdefiniowanymi takimi samymi parametrami działania algorytmu.

Wyniki porównano.



Rys.26 Wynik klasteryzacji komórek PBMC z użyciem metody k-Means, udostępnianej przez bibliotekę scikit-learn w Python. Efekt kodu zastosowanego w pracy.



Rys.27 Wynik klasteryzacji komórek PBMC metodą k-Means, programem Floreada.io dostępnym on-line (https://floreada.io/analysis).

Uzyskany wynik jak i analiza programem Floreada.io nie dają pożądanych efektów. Głównym celem podczas grupowania komórek PBMC jest przede wszystkim wyodrębnienie populacji limfocytów oraz oddzielenie ich od fragmentów komórkowych stanowiących tzw resztki (debris). To limfocyty najczęściej podlegają dalszej analizie. Drugą populacją komórek którą można również chcieć poddać badaniu są monocyty. Niestety ani zastosowana w tej pracy metoda ani udostępniony on-line program nie gwarantują uzyskania prawidłowego rozdziału na populacje.

Obydwa programy źle rozpoznają subpopulacje wchodzące w skład komórek PBMC. Jako grupa stanowiąca resztki komórkowe powstające podczas preparatyki laboratoryjnej identyfikowana jest cała populacja limfocytów wraz z debris.

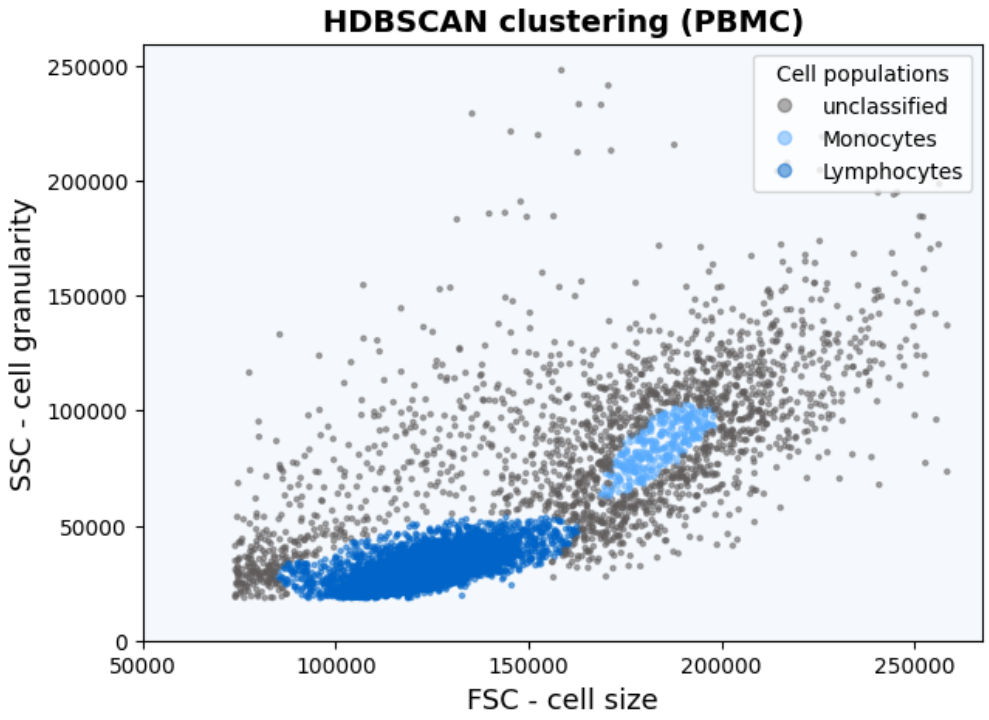
Należy zauważyć jednak, że sposób przygotowania zawartych w pliku cytometrycznym danych który został zastosowany w pracy jest lepszy aniżeli oferuje to udostępniony on-line program. Uzyskany wykres jest pozbawiony wartości odstających co wpływa jak widać na jakość klastrowania ponieważ grupa danych zaliczana do monocytów w przypadku metody zastosowanej w pracy mniej rozmija się z rzeczywistością niż w przypadku programu Floreada.io.

HDBSCAN

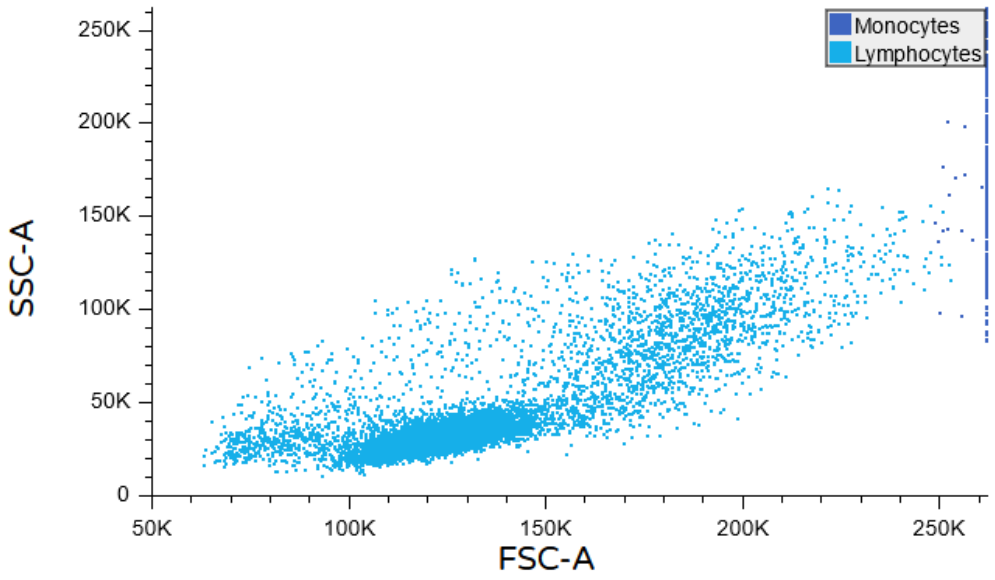
Jako drugą metodę klasteryzacji sprawdzano zastosowanie algorytmu HDBSCAN. Ponieważ program Floreada.io oferuje zastosowanie do automatycznej klasteryzacji algorytm DBSCAN, którego modyfikację stanowi użyta w pracy metoda, postanowiono porównać efekt ich zastosowania.

Metoda DBSCAN niestety nawet po wielu próbach dostosowywania parametrów działania nie radziła sobie z klasteryzacją tego typu danych. Jedynym plusem jej działania jest fakt że część danych które przez tę aplikację zostały uznane za wartości odstające nie znalazła się na wykresie.

Użyty w pracy algorytm HDBSCAN wydaje się zdecydowanie lepszym rozwiązaniem zarówno w porównaniu z programem Floreada.io jak i metodą k-Means. Dobrze rozpoznaje populację limfocytów i oddziela ją od pozostałych grup. Niestety najlepszy uzyskany eksperymentalnie wynik nie jest zadowalający jeżeli chodzi o populację monocytów - zbyt dużo punktów, które powinny należeć do tej grupy nie zostało tam zaliczonych.



Rys.28 Wynik klasteryzacji komórek PBMC z użyciem algorytmu HDBSCAN. Efekt działania kodu zastosowanego w pracy.

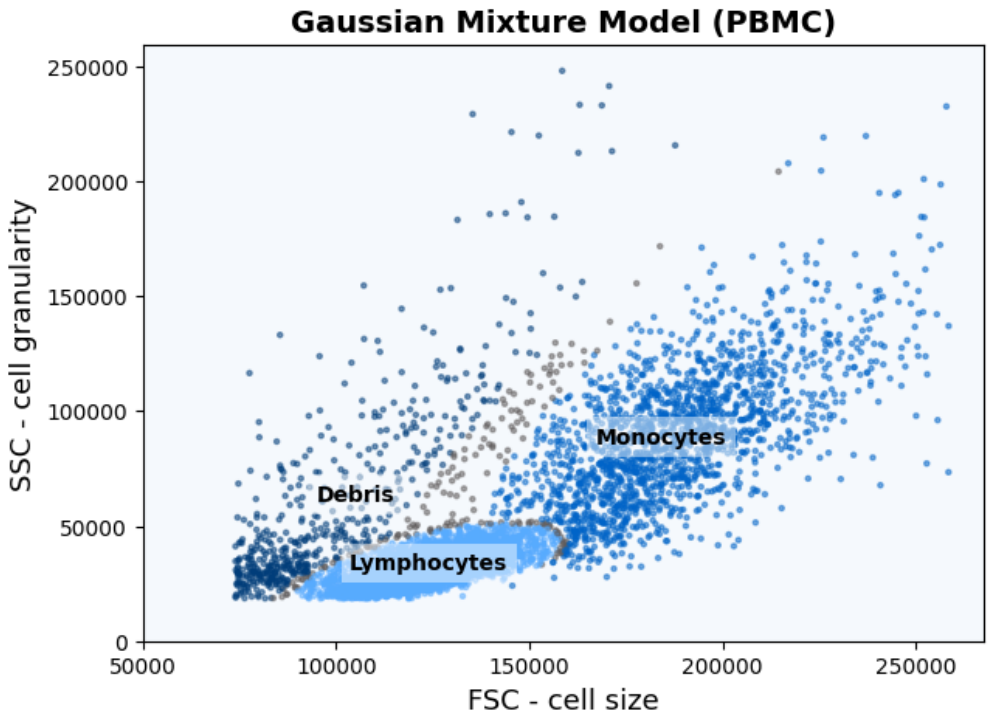


Rys.29 Wynik klasteryzacji komórek PBMC metodą DBSCAN, z użyciem programu Floreada.io dostępnym on-line (https://floreada.io/analysis).

Gaussian Mixture Model (GMM)

Jako kolejną metodę sprawdzano algorytm Gaussian Mixture Model. Niestety autorzy programu Floreada.io nie oferują już więcej metod automatycznej klasteryzacji.

Jednak działanie metody klasteryzacji GMM w przypadku danych cytometrycznych dało bardzo obiecujący wynik. Algorytm rozpoznał prawidłowo wszystkie trzy grupy danych. Określając próg prawdopodobieństwa dla każdego z klastrów ograniczono możliwość błędnego zakwalifikowania punktu. Dobrze zdefiniowana jest zarówno grupa komórek o parametrach odpowiadających populacji limfocytów jak i tych należących do populacji monocytów. Wyraźnie obrazuje to wykres zależności FSC i SSC otrzymany w wyniku klasteryzacji metodą GMM.



Rys.30 Wynik klasteryzacji komórek PBMC z użyciem algorytmu Gaussian Mixture Model udostępnianego przez bibliotekę scikit-learn w Python. Punkty koloru szarego to tzw. punkty nie zdefiniowane. Efekt działania kodu użytego w pracy.

4.2 Walidacja zastosowanych metod klasteryzacji

Wyniki mierzonych parametrów poszczególnych metod klasteryzacji określających jakość i czas wykonanego zadania zebrano w ramce danych 'score'.

Najszybszym algorytmem w przypadku rodzaju danych badanych w pracy oraz ich konkretnej ilości (10000 rekordów), okazał się k-Means. Metoda ta osiągała też lepsze wyniki w przypadku miar wewnętrznych jak Silhouette Score i Davies-Bouldin Index, które oceniają jednak bardziej strukturę samego wyodrębnionego klastra a nie zgodność podziału całego zestawu danych na oczekiwane grupy. W parametrach należących do miar zewnętrznych lepsze były HDBSCAN i GMM ze zdecydowaną przewagą tego drugiego. To te miary są w przypadku podstawowego fenotypowania komórek najistotniejsze.

	time	Jaccard	Silhouette	Davies	ARI	NMI	FMI
k-Means	0.013772	0.116638	0.625608	0.81619	0.708231	0.550843	0.875388
HDBSCAN	0.296973	0.555282	0.506207	2.653619	0.7847	0.653357	0.906653
GMM	0.045495	0.691118	0.588722	1.155289	0.858363	0.742166	0.935668

Rys.31 Wyniki miar walidacji zastosowanych w pracy metod klasteryzacji danych z pomiaru cytometrycznego dotyczących odczytów na detektorach FSC i SSC.

Przyznano po 3 punkty najlepszemu uzyskanemu wynikowi z danego parametru a po 1 punkcie najgorszemu. Rezultaty zapisano w ramce danych.

	time	Jaccard	Silhouette	Davies	ARI	NMI	FMI
k-Means	3.0	1.0	3.0	3.0	1.0	1.0	1.0
HDBSCAN	1.0	2.0	1.0	1.0	2.0	2.0	2.0
GMM	2.0	3.0	2.0	2.0	3.0	3.0	3.0

Rys.32 Zestawienie przyznanych punktów za osiągnięte wyniki w użytych miarach walidacyjnych.

Po wyliczeniu średniej ważonej oraz procentu uzyskanego wyniku ogólnego dla wszystkich zastosowanych miar przy ocenie algorytmów klasteryzacji, ostateczne rezultaty zostały zapisane w ramce danych.

	Weighted_Average	Percent_Score
k-Means	1.72	57.33
HDBSCAN	1.64	54.67
GMM	2.64	88.00

Rys.33 Ostateczne rezultaty określenia przydatności wybranych algorytmów klasteryzacji w przypadku danych cytometrycznych wykorzystywanych do podstawowych oznaczeń fenotypu komórek PBMC.

5. Podsumowanie

Wyznaczenie dwóch głównych populacji komórek PBMC jakimi są limfocyty i monocyty, stanowi pierwszy krok w analizie cytometrycznej i rzutuje na kolejne jej etapy. W pracy sprawdzano przydatność trzech podstawowych technik klasteryzacji danych używanych w analizie skupiń, jak k-Means, HDBSCAN oraz GMM. Nie każda z metod okazała się równie skuteczna. Za jedyny dobrze rokujący algorytm można uznać GMM, który w przyjętej punktacji oceny działania, w przypadku tego typu danych ulokował się na pierwszej pozycji zdobywając 2,64 punkta na 3,0 co stanowi 88% skuteczności.

Pomimo, że metoda HDBSCAN według uzyskanych punktów zajęła ostatnie trzecie miejsce, należy zauważyć, że różnica pomiędzy nią a k-Means w przyjętym sposobie oceny jest bardzo mała. Z kolei biorąc pod uwagę wykresy zależności pomiarów FSC i SSC z naniesionymi wynikami działania tych dwóch algorytmów, nalezy stwierdzić zdecydowaną przewagę techniki HDBSCAN. Mozna przypuszczać, że niski wynik uzyskany w przypadku tej metody może wynikać ze zbyt restrykcyjnego wyodrębnienia populacji monocytów podczas której algorytm 'gubił' dużą ilość komórek, które powinien jeszcze przypisać do tego klastra. Możliwe, że po ustaleniu lepszych parametrów jego działania lub zastosowaniu pewnych jego modyfikacji udałoby się poprawić tę skuteczność.

Czy jednak stosowanie uczenia nienadzorowanego daje możliwość automatyzacji tego typu analiz?

Algorytm GMM jest pod tym kątem bardzo obiecujący i warto podjąć próbę jego zastosowania w tym celu. Sprawdza się on dobrze zarówno pod względem poprawności klasteryzacji jak i biorąc pod uwagę rzeczywisty czas wykonywania podziału. Potrzebuje on co prawda więcej czasu aniżeli algorytm k-Means ale dla zarówno celów badawczych jak i klinicznych wazniejsza jest w tym przypadku poprawność podziału, a tego zdecydowanie brakuje k-Means.

Przygotowanie danych pozyskanych z plików cytometrycznych stosując metody dostępne w bibliotekach Python, sprawdzają bardzo dobrze i mogą być dokonywane bez konieczności udziału użytkownika. Charakter danych sprawia, że przebieg procesu

przygotowawczego nie jest skomplikowany, a utrata rekordów w powstającej w ten sposób strukturze ramki danych to około 1,5% na kolumnę. Trzeba jedynie pamiętać, że liczba kolumn będzie rosła wraz ze wzrostem mierzonych parametrów na komórkach. W przypadku dużej ich ilości należałoby wprowadzić modyfikację samej metody pomiaru cytometrycznego zwiększając liczbę odczytów (czyli ilość mierzonych komórek), co zwiększy nam w efekcie zawartość rekordów w pliku. Liczba dokonywanych pomiarów w plikach cytometrycznych wykorzystanych w pracy to 10000. Jest to minimum potrzebne przy oznaczeniach 6-10 parametrach (oprócz FSC i SSC, przy których odczytach nie wykorzystuje się barwników flurescencyjnych).
Kontrolując ilość odrzucanych danych, czyli zbierając informacji na temat utworzonej ramki danych w wyniku przygotowania pliku cytometrycznego i przedstawienie ich do akceptacji użytkownika przed przystąpieniem do docelowej analizy na pewno ustrzeże przed ewentualnymi błędami.
Automatyzacja tego etapu jest możliwa i pożądana.

Dalsza automatyzacja analizy pliku, czyli wyodrębnienie z uzyskanej ramki danych największych populacji komórek na podstawie ich wielkości i struktury wewnętrznej (ziarnistości) pozostaje sprawą dyskusyjną, ponieważ całkowite zaufanie algorytmowi GMM raczej nie będzie mozliwe. W pracy poddano sprawdzeniu ta metodą w sumie 350 plików wygenerowanych podczas projektu MINIATURA 2, kazdy zawierający wyniki pomiarów 10000 komórek. Z tego nadawało się do przeprowadzenia zautomatyzowanej analizy metodami klasteryzacji 224 plików. Wynikało to z faktu nie uzyskania klasycznego rozkładu danych cytometrycznych w pozostałych 126 plikach. Na odmienność rozkładu pomiarów mogą wpływać drobne błędy preparatyki próbki. Pliki te nie nadawały się do tego typu analizy automatycznej i wymagały metody manualnej z wykorzystaniem wiedzy eksperckiej.
Z tego powodu Użycie algorytmu GMM może być zastosowane jedynie pod warunkiem przedstawienia wyniku swojego działania nie tylko w ostatecznie interesującej badacza, diagnostę lub lekarza formie procentowej zawartości populacji w mieszaninie komórek, ale musi przedstawiać również podstawowe informacje dotyczące procesu klasteryzacji i wykresy dot-plot obrazujące efekty pracy algorytmu.

Jednak możliwość analizy 224 plików z 350 stanowi już bardzo duże usprawnienie analizy tego typu danych. Jest duża oszczędność czasu, a zaangażowanie do analizy pomiaru cytometrycznego osoby specjalizującej się konkretnie w tej technice laboratoryjnej może skupić się jedynie na pozostałych, problematycznych 126 plikach.

W dążeniu do automatyzacji tego typu zadań nie można zapominać o istnieniu bardziej zaawansowanych grupach metod uczenia maszynowego, które mogą dać lepsze wyniki niz metody klasteryzacji czyli algorytmach klasyfikacyjnych uczenia nadzorowanego.
Należałoby sprawdzić, któe z nich najlepiej radzą sobie w przypadku ustalania np. fenotypu komórek i porównać ich skuteczność i czas działania z algorytmami klasteryzacji danych.

Obecnie istnieje potrzeba tego typu rozwiązań, szczególnie dostępnych on-line. pomogłoby to nie tylko usprawnić analizę i skrócić jej czas ale także ujednolicić metodykę oceny uzyskanych wyników podczas pomiaru i ustrzec przed typowymi błędami początkujących adeptów cytometrii przepływowej.

Literatura:

[1] Rieseberg M, Kasper C, Reardon KF, Scheper T. Flow cytometry in biotechnology. Appl Microbiol Biotechnol. 2001 Aug;56(3-4):350-60. doi: 10.1007/s002530100673.

[2] El-Hajjar L, Ali Ahmad F, Nasr R. A Guide to Flow Cytometry: Components, Basic Principles, Experimental Design, and Cancer Research Applications. Curr Protoc. 2023 Mar;3(3):e721. doi: 10.1002/cpz1.721.

[3] Sędek Ł, Sonsala A, Szczepański T, Mazur B. Techniczne aspekty cytometrii przepływowej. Diagn Lab. 2010; 46 (4): 415-420.

[4] Skotny A, Pucińska J. Współczesna cytometria przepływowa. (Modern flow cytometry). Acta Bio-Optica et Informatica Medica Inżynieria Biomedyczna, vol. 19, nr 1, 2013.

[5] Sędek Ł, Mazur B. Przeciwciała monoklonalne i poliklonalne i ich zastosowanie w cytometrii przepływowej. (The application of monoclonal and polyclonal antibodies in flow cytometry). Postępy biologii komórki, 2008 t. 24 nr Suplement 24, s. 17-34.

[6] Adan A, Alizada G, Kiraz Y, Baran Y, Nalbant A. Flow cytometry: basic principles and applications. Crit Rev Biotechnol. 2017 Mar;37(2):163-176. doi: 10.3109/07388551.2015.1128876.

[7] Kaczmarek A, Mackiewicz A, Leporowska E, Osawa T. The role of flow cytometry in clinical diagnosis. Contemporary Oncology/Współczesna Onkologia. 2002;6(6):366-373.

[8] Gunasekera TS, Attfield PV, Veal DA. A flow cytometry method for rapid detection and enumeration of total bacteria in milk. Appl Environ Microbiol. 2000 Mar;66(3):1228-32. doi: 10.1128/AEM.66.3.1228-1232.2000.

[9] Robinson JP. Flow cytometry: past and future. Biotechniques. 2022 Apr;72(4):159-169. doi: 10.2144/btn-2022-0005.

[10] https://www.learnhaem.com/courses/flow-cytometry/lessons/limitations/

[11] https://isac-net.org/

[12] Bashashati A, Brinkman RR. A survey of flow cytometry data analysis methods. Adv Bioinformatics. 2009;2009:584603. doi: 10.1155/2009/584603.

[13] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput. 1999 Surv. 31, 3, 264–323. doi: 10.1145/331499.331504.

[14] https://en.wikipedia.org/wiki/Cluster_analysis

[15] https://en.wikipedia.org/wiki/Supervised_learning

[16] https://floreada.io/analysis

[17] https://www.flowjo.com/

[18] Spidlen J, Moore W, Parks D, Goldberg M, Blenman K, Cavenaugh JS; ISAC Data Standards Task Force; Brinkman R. Data File Standard for Flow Cytometry, Version FCS 3.2. Cytometry A. 2021 Jan;99(1):100-102. doi: 10.1002/cyto.a.24225.

[19] Sun S, Wolf A (2022). readfcs: Read FCS files. Lamin Blog. https://doi.org/10.56528/rfcs

[20] Aghaeepour N, Nikolic R, Hoos HH, Brinkman RR. Rapid cell population identification in flow cytometry data. Cytometry 2011, 79A: 6-13. https://doi.org/10.1002/cyto.a.21007.

[21] Hearty J. Zaawansowanie uczenie maszynowe z językiem Python. Gliwice, Helion, 2017.

[22] Albrzykowski L. Uczenie maszynowe. Elementy matematyki w analizie danych. Gliwice, Helion, 2023.

[23] Lattanzi S, Sohler C. A Better k-means++ Algorithm via Local Search. Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research (2019) 97:3662-3671.

[24] Stewart G, Al-Khassaweneh M. An Implementation of the HDBSCAN* Clustering Algorithm. Applied Sciences. 2022; 12(5):2405. https://doi.org/10.3390/app12052405.

[25] https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

[26] https://scikit-learn.sourceforge.net/dev/modules/mixture.html

[27] https://www.mygreatlearning.com/blog/gaussian-mixture-model/

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

[29] Frühwirth-Schnatter S, Celeux G, Robert Ch. Handbook of Mixture Analysis. Taylor & Francis, 2019.

[30] Santos JM, Embrechts M, "On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification", Lecture Notes in Computer Science, Springer, Vol. 5769, pp. 175-184, 2009.

[31] Danon L, Díaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment. 2005 Sept 1;(9):219-228. doi: 10.1088/1742-5468/2005/09/P09008.

[32] Taya, F, de Souza, J, Thakor, NV. Comparison method for community detection on brain networks from neuroimaging data. Appl Netw Sci 1, 8 (2016). https://doi.org/10.1007/s41109-016-0007-y.

[33] https://en.wikipedia.org/wiki/Fowlkes%E2%80%93Mallows_index

[34] https://en.wikipedia.org/wiki/Precision_and_recall

[35] Fletcher S, Islam MZ. Comparing sets of patterns with the Jaccard index. AJIS. 2018.7;220. doi: 10.3127/ajis.v22i0.1538.

[36] https://en.wikipedia.org/wiki/Jaccard_index