Projekt 2 - raport

Jakub Kozieł
Tomasz Krupiński
Jakub Lis

1 Opis problemu

Zadaniem było stworzenie modelu, który podzieli dane na klastry. Dane pochodzą ze strony archive.ics.uci.edu/ml/datasets. W zbiorze mamy bardzo dużą liczbę cech, ponieważ zawiera on 561 kolumn. Każdy rekord reprezentuje różne statystyki odczytów aktywności ludzi mierzonych przez akcelerator lub żyroskop. Dane są przeskalowane i ograniczone do przedziału [-1; 1].

```
X_train
```

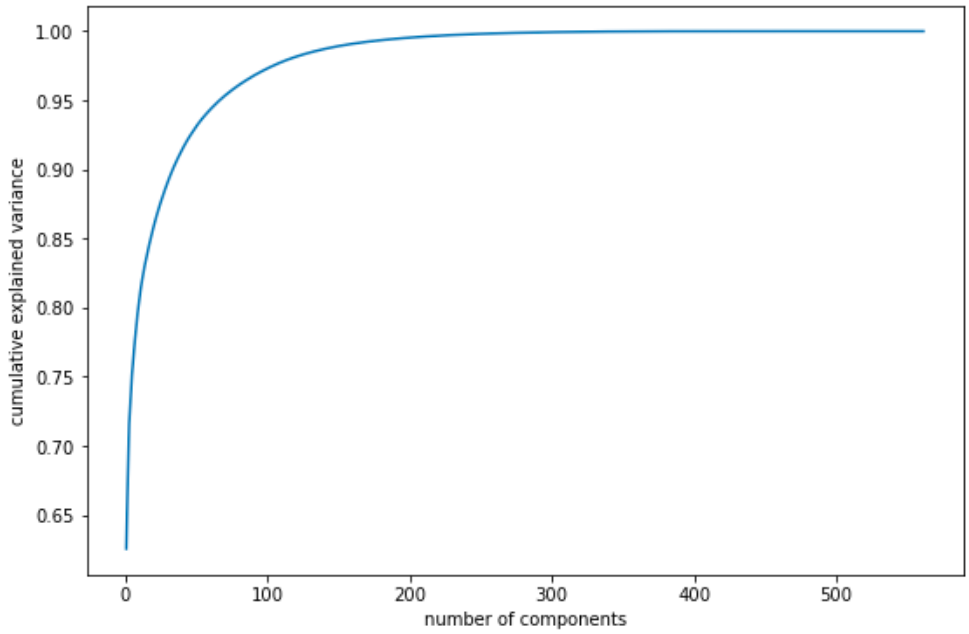
	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-meanFreq()	fBod skev
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.074323	-0.29
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	0.158075	-0.59
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	0.414503	-0.39
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	0.404573	-0.11
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	0.087753	-0.35
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7347	0.299665	-0.057193	-0.181233	-0.195387	0.039905	0.077078	-0.282301	0.043616	0.060410	0.210795	...	-0.070157	-0.58
7348	0.273853	-0.007749	-0.147468	-0.235309	0.004816	0.059280	-0.322552	-0.029456	0.080585	0.117440	...	0.165259	-0.39
7349	0.273387	-0.017011	-0.045022	-0.218218	-0.103822	0.274533	-0.304515	-0.098913	0.332584	0.043999	...	0.195034	0.025
7350	0.289654	-0.018843	-0.158281	-0.219139	-0.111412	0.268893	-0.310487	-0.068200	0.319473	0.101702	...	0.013865	0.063
7351	0.351503	-0.012423	-0.203867	-0.269270	-0.087212	0.177404	-0.377404	-0.038678	0.229430	0.269013	...	-0.058402	-0.38

7352 rows × 561 columns

Podgląd danych

2 Inżynieria cech

Przy tak dużej liczbie kolumn, pierwszym co sprawdzaliśmy były korelacje pomiędzy różnymi kolumnami - chcieliśmy trochę zredukować wymiarowość danych. Jak się okazało, korelacje w większości przypadków były bardz duże (często bliskie 1), zatem oczywistym wnioskiem było, że należy zredukować liczbę kolumn. Pomimo naszego pierwszego pomysłu, jakim było usuwanie skorelowanych kolumn, zdecydowaliśmy się na zastosowanie PCA. Taka decyzja wynikała z przebadania zachowania części modeli po redukcji wymiarów poprzez ręczne usuwanie kolumn oraz po zastosowaniu PCA. Wybraliśmy to, co dawało lepsze rezultaty (np. wyższe wartości metryki silhouette_score).



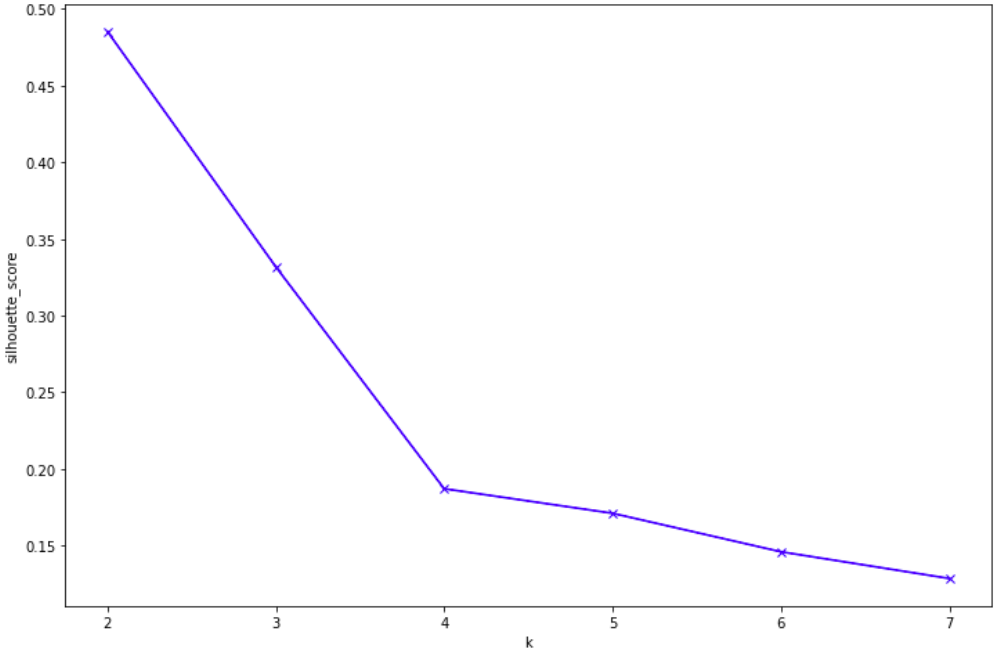
Wykres wyjaśnionej wariancji (w procentach) w zależności od liczby komponentów

Widzymy, że procent wyjaśnionej wariancji rośnie dosyć szybko, zatem nie potrzebowaliśmy ogromnej liczby wymiarów. Sprawdziliśmy, że dla 30 komponentów procent wyjaśnionej wariancji jest na poziomie 0.89 i uznaliśmy tę wartość za wstarczającą. Ograniczyliśmy nasz zbiór do 30-wymiarowego PCA.

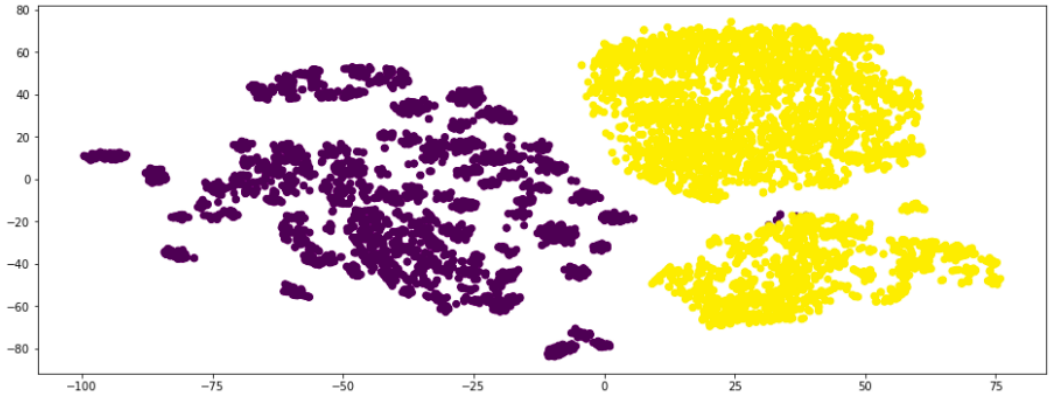
3 Modelowanie

Note: zachęcamy do korzystania z ipnyba, gdzie będzie możliwa interakcja z grafikami 3D, jeśli zajdzie taka potrzeba lub z htmla, gdzie nie jest potrzebne uruchomienie kodu od początku, aby wykresy były załadowane

Nasze wstępne modelowanie opierało się na metodzie K-Means. Różnymi metrykami, takimi jak silhouette_scorev, calinski_harabasz_score, davies_bouldin_score sprawdzaliśmy jaka będzie odpowiednia liczba klastrów. Wszystkie jednoznacznie wskazywały, że nasz zbiór jest najlepiej podzielić na 2 klastry, więc tak też zrobiliśmy.



Miara silhouette w zależnośći od dobranej liczby klastrów



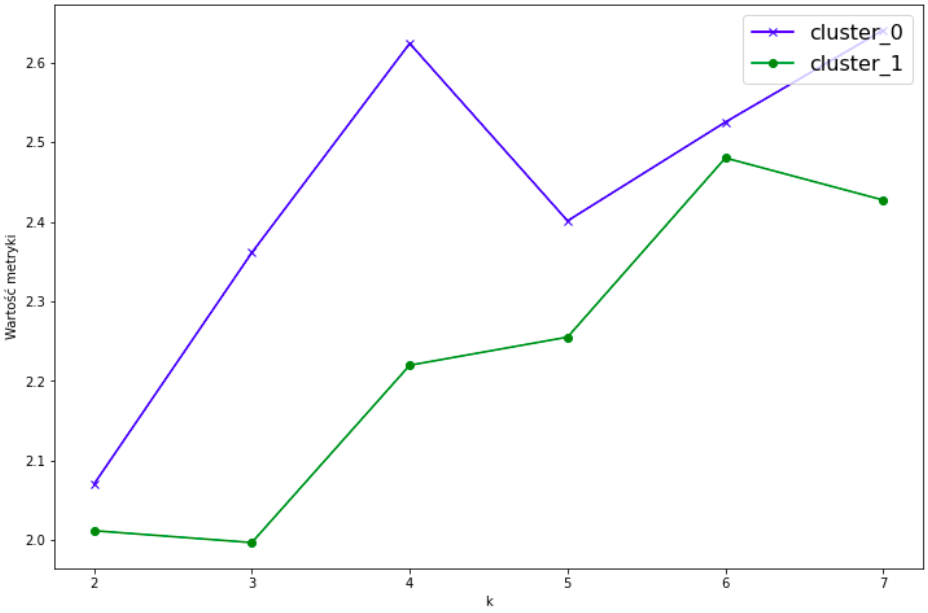
Uzyskane grupowanie na 2 klastry metodą K-Means zwizualizowane za pomocą T-SNE

Nic dziwnego, że każda stosowana metryka podpowiadała podział na 2 klastry. Jak porównaliśmy otrzymane wyniki z oryginalnymi labelami podczas drugiego kamienia milowego, to okazało się, że udało się podzielić aktywnośći na chodzenie i takie, które nie wymagają ruchu. Wiedząc jednak, że aktywności te można podzielić jeszcze bardziej postanowiliśmy stosować klastrowanie dalej - na obu powstałych zbiorach niezależnie od siebie.

Jako, że algorytm K-Means świetnie poradził sobie z początkową klasteryzacją, to nie szukaliśmy innego do tego etapu. Przebadaliśmy natomiast, różne algorytmy, aby dokonać kolejnych podziałów. Dla każdego stosowaliśmy metryki sprawdzające jaką liczbę klastrów wybrać, a następnie dzieliliśmy zbiór na klastry i wizualizowaliśmy otrzymane wyniki. Pokazywaliśmy także porównania z oryginalnymi labelami, pomimo że nie chcemy się nimi sugerować przy ostatecznym wyborze modelu - bo w realnym zadaniu nie mielibyśmy dostępu do takich etykiet.

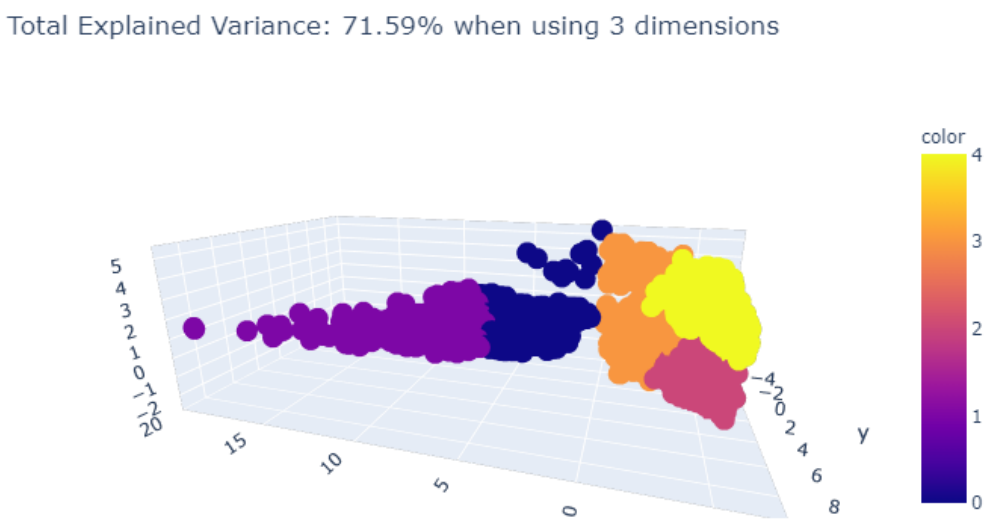
3.1 K-Means

Ponownie zaczęliśmy od algorytmu K-Means, dla którego po zastosowaniu metryki davies_bouldin_score zdecydowaliśmy się na podział klastra "zerowego" na 2 nowe klastry, a klastra "pierwszego" na 3 nowe.

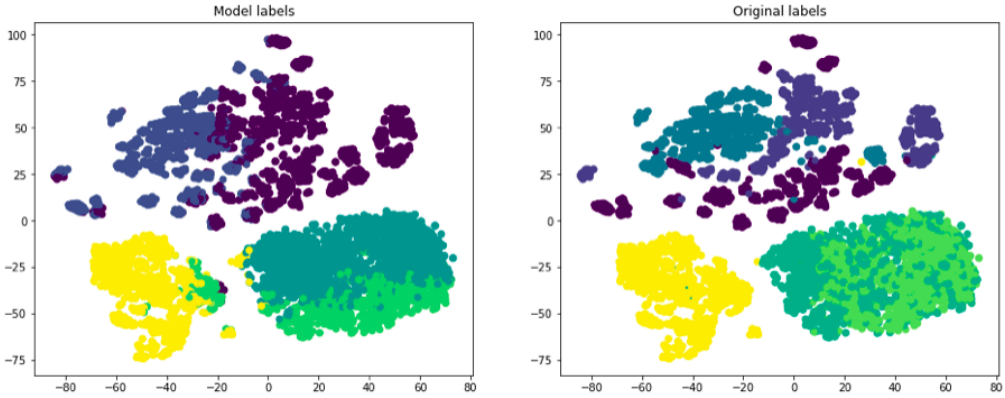


Wartość metryki davies_bouldin_score w zależności od wybranej liczby klastrów, z podziałem na klaster zerowy i pierwszy

Dla każdego podziału tworzyliśmy interaktywny wykres 3D dobrze obrazujący nasz podział oraz wykresy 2D, gdzie porównywaliśmy otrzymane podziały z oryginalnymi etykietami.



Wykres 3D obrazujący uzyskany podział na klastry



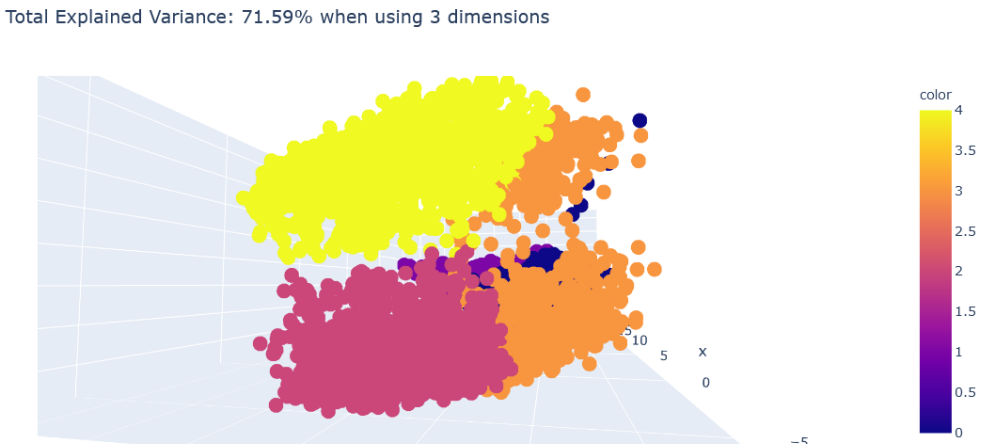
Porównanie otrzymanej klasteryzacji z oryginalnymi etykietami

A dodatkowo tworzyliśmy confusion matrix, gdzie sprawdzaliśmy, do jakich odpowiednio klastrów trafiały oryginalne etykiety.

	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	853.0	914.0	259.0	1.0	0.0	11.0
1	373.0	159.0	727.0	0.0	0.0	0.0
2	0.0	0.0	0.0	914.0	935.0	0.0
3	0.0	0.0	0.0	321.0	439.0	157.0
4	0.0	0.0	0.0	50.0	0.0	1239.0

Confusion matrix

Warto też przyjrzeć się uzyskanemu wcześniej wykresowi 3D, ale z innej perspektywy. Widzimy ciekawy podział na klastry (różowy, żółty, pomarańczowy). Ciekawy, dlatego, że inne algorytmy przeważnie dzieliły ten obszar inaczej. Szczególnym przypadkiem jest ten podział "pionowy", zaznaczony na pomarańczowo.



Wykres 3D obrazujący uzyskany podział, ale z innej perspektywy

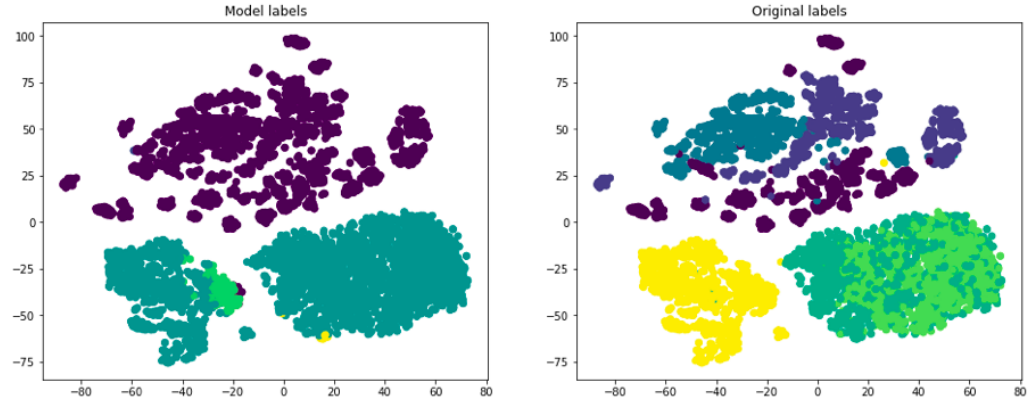
3.2 AgglomerativeClustering

Kolejnymi stosowanymi przez nas algorytmami są algorytmy, zaliczane do grupy aglomeracyjnych. Pod uwagę braliśmy połączenia typu ward, complete, average oraz single.

Dla każdego typu połączeń dobraliśmy odpowiednie, wskazane przez matryki liczby klastrów, przykładowo dla typu single wybraliśmy podział obu klastrów (zerowego i pierwszego) na 3 nowe (każdego z osobna).

3.2.1 Average

W przypadku average dzieliliśmy klaster zerowy na dwa, a klaster pierwszy na 3 - co jest dobrze widoczne w confusion matrix poniżej.



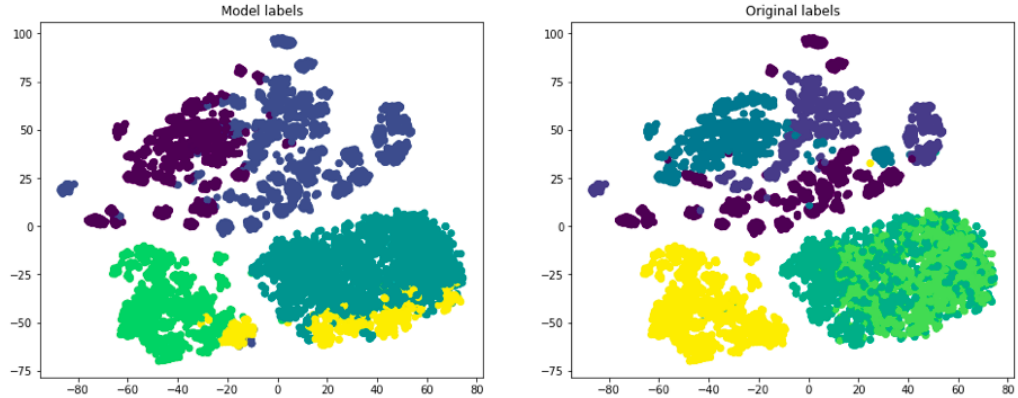
	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	1226.0	1073.0	984.0	1.0	0.0	11.0
1	0.0	0.0	2.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1266.0	1374.0	1231.0
3	0.0	0.0	0.0	0.0	0.0	165.0
4	0.0	0.0	0.0	19.0	0.0	0.0

Otrzymane rezultaty dla połączenia typu average

Otrzymany podział, patrząc jedynie na wykres po lewej (czyli bez korzystania z oryginalnych etykiet) wydaje się być bardzo słaby. Szczególnie można zwrócić uwagę na klaster żółty, który ma 2 elementy, które w żaden sposób nie są wyodrębnione od pozostałych (tak naprawdę mamy na turkusowym dwa żółte punkty, które intuicyjnie w takim przypadku też powinny być turkusowe).

3.2.2 Ward

Dla połączenia "ward" liczba dobranych klastrów dla podziału klastra zerowego i pierwszego jest taka sama jak dla average (tzn. odpowiednio podział na 2 i na 3 nowe klastry).



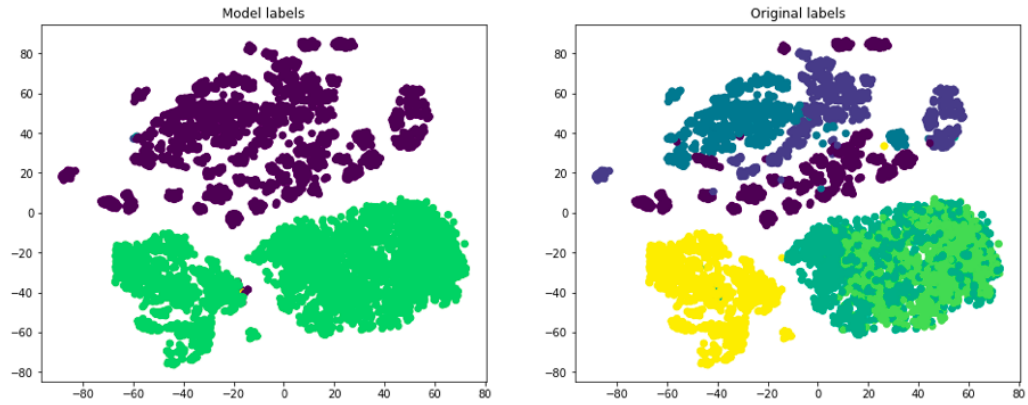
	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	350.0	114.0	818.0	0.0	0.0	0.0
1	876.0	959.0	168.0	1.0	0.0	11.0
2	0.0	0.0	0.0	1107.0	1140.0	1.0
3	0.0	0.0	0.0	8.0	0.0	1260.0
4	0.0	0.0	0.0	170.0	234.0	135.0

Otrzymane rezultaty dla połączenia typu ward

W naszym odczuciu podział nie jest zły, a zdecydowanie jest najlepszy z zastosowanych metod aglomeracyjnych, jak się zaraz okaże.

3.2.3 Single

W przypadku połączeń single zgodnie z zastosowanymi metrykami podzieliiśmy klaster zerowy na 3 nowe klastry, a klaster pierwszy na 2.



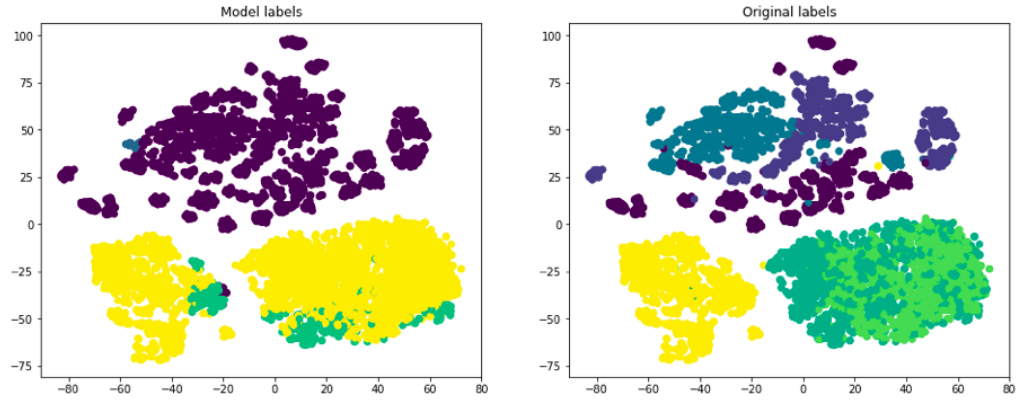
	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	1226.0	1073.0	983.0	1.0	0.0	11.0
1	0.0	0.0	2.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1285.0	1374.0	1394.0
4	0.0	0.0	0.0	0.0	0.0	2.0

Otrzymane rezultaty dla połączenia typu single

Otrzymana klasteryzacja była bardzo słaba, szczególnie przez otrzymanie klastrów posiadających 1 lub 2 punkty.

3.2.4 Complete

Dla połączeń complete podzieliliśmy każdy z dwóch klastrów na 2 nowe.



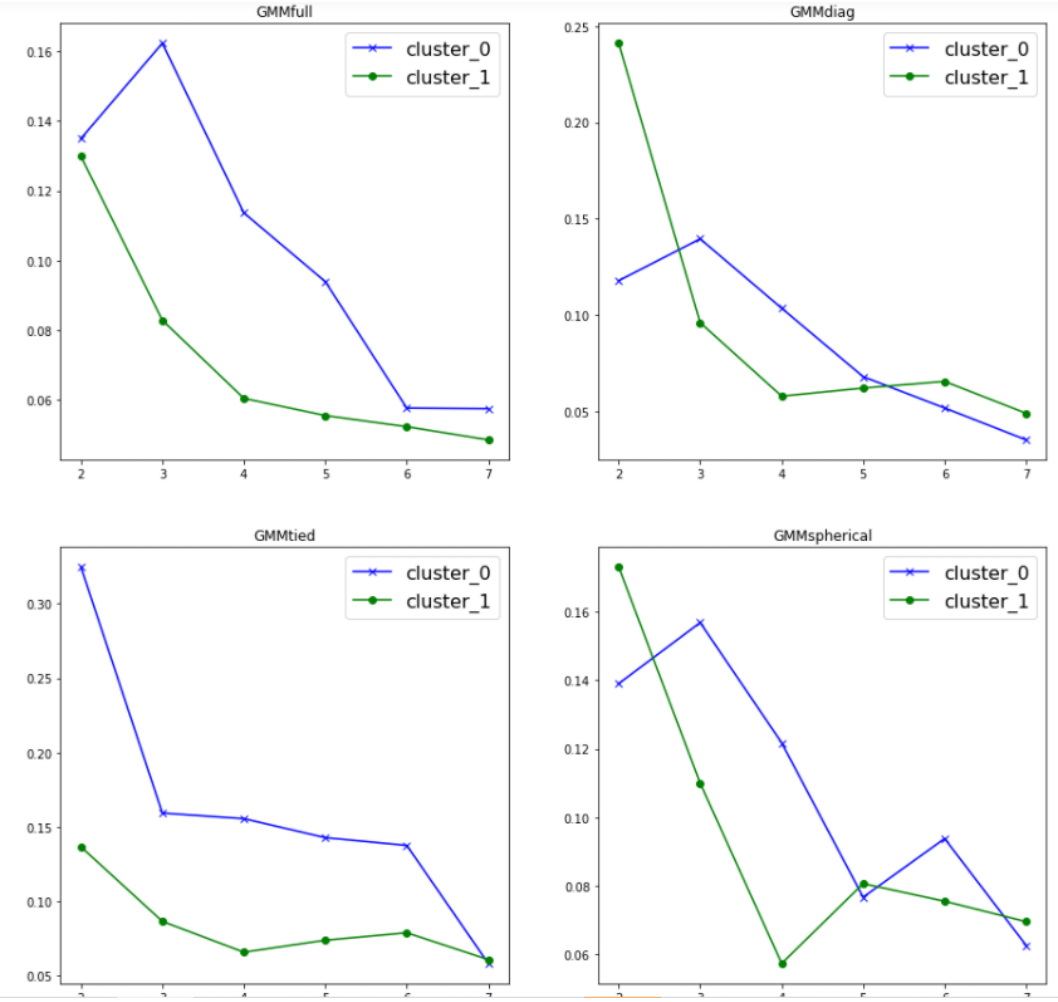
	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	1225.0	1073.0	964.0	1.0	0.0	11.0
1	1.0	0.0	22.0	0.0	0.0	0.0
2	0.0	0.0	0.0	159.0	88.0	155.0
3	0.0	0.0	0.0	1126.0	1286.0	1241.0

Otrzymane rezultaty dla połączenia typu complete

Znów powstał małoliczny klaster, zatem gdybyśmy wybierali algorytm aglomeracyjny, to pozostalibysmy przy wardzie.

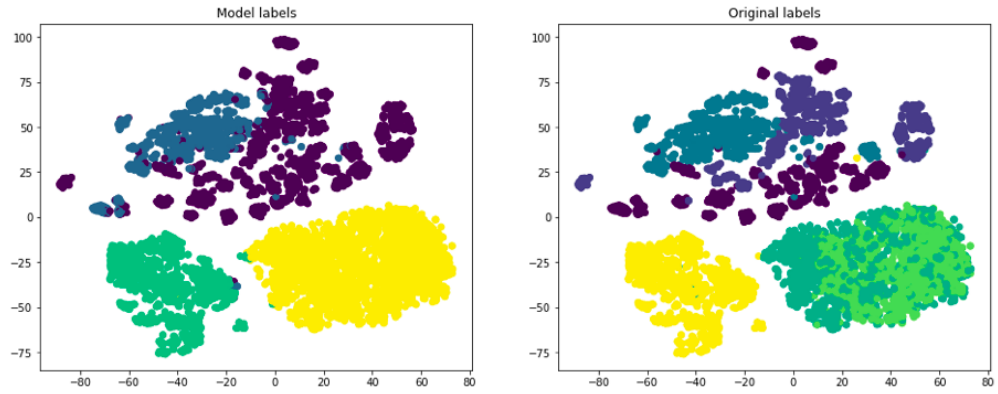
4 GMM

Następnym wybranym algorytmem był GMM, ponownie zaczęliśmy od metryk takich jaki silhouette_score, które widać poniżej.



Metryka silhouette score dla a modelu GMM

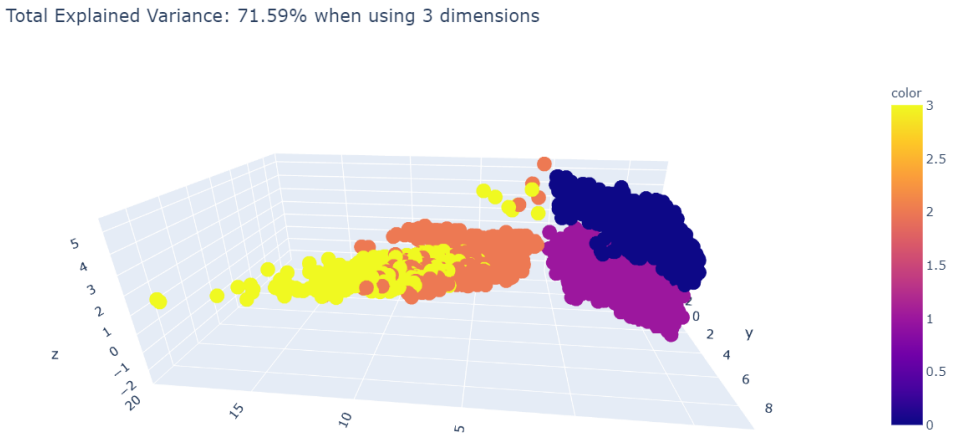
Zastosowaliśmy jeszcze metrykę davies_bouldin_score, ale jej wyniki były bardzo podobne. Stąd decyzja na podziały takie, gdzie widać najwyższą wartość silhouette_score. Tym razem pokażemy jedynie jedną wartość covariance_type, a nie jak w przypadku metod aglomeracyjnych. Uznaliśmy, że najlepiej zaklasteryzował nasze dane model z covariance_type typu tied.



	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	1137.0	1056.0	104.0	1.0	0.0	4.0
1	89.0	17.0	882.0	0.0	0.0	7.0
2	0.0	0.0	0.0	17.0	0.0	1396.0
3	0.0	0.0	0.0	1268.0	1374.0	0.0

Klasteryzacja dla modelu GMM z covariance_type "tied"

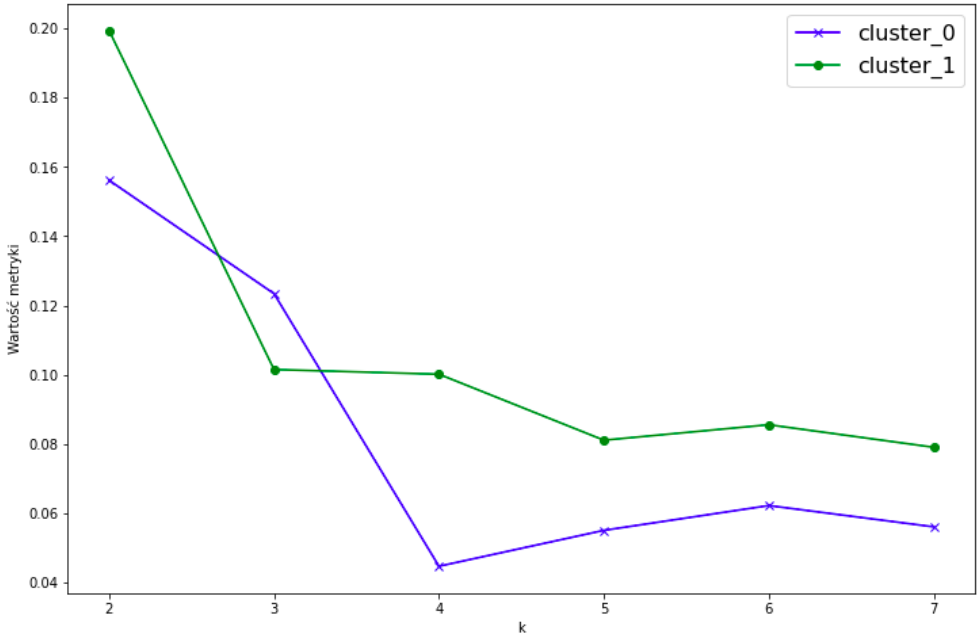
Uznaliśmy, że ta klasteryzacja jest całkiem dobra ze względu na dobrze rozłożone liczności klastrów i widoczne rozdzielenie (chociaż trochę) pomiędzy klastrami.



Wykres 3D obrazujący uzyskaną klasteryzacją za pomocą GMM

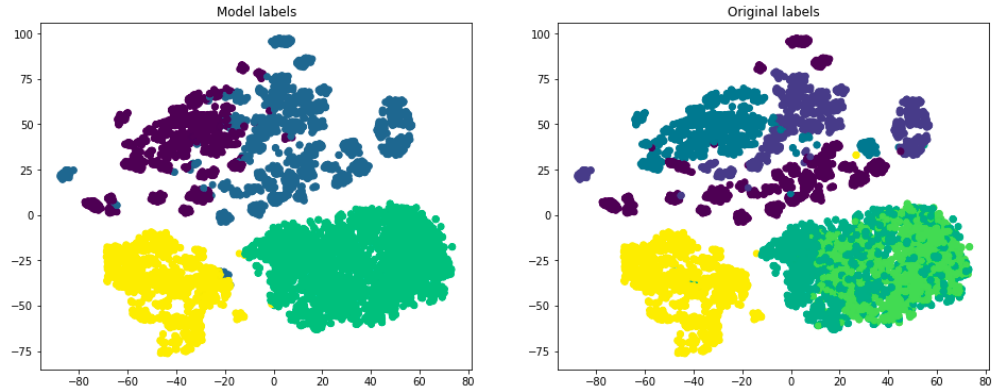
5 Birch

Na końcu rozpatrywaliśmy algorytm Birch, dla którego także zaczęliśmy od wyznaczenia odpowiednich liczb klastrów za pomocą metod silhouette_score oraz davies_bouldin_score.



Metryka silhouette score dla a modelu GMM

Druga metryka dała identyczne podpowiedzi, co do wyboru liczby klastrów, tzn. aby dobrać po 2 klastry dla obu naszych zbiorów. Wyniki zastosowanego algorytmu prezentujemy poniżej.



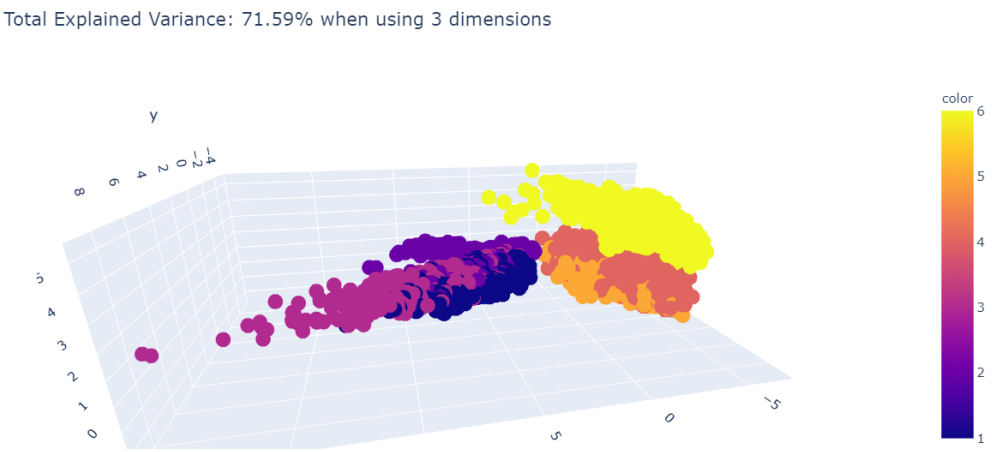
	WALKING	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS	SITTING	STANDING	LAYING
0	350.0	114.0	818.0	0.0	0.0	0.0
1	876.0	959.0	168.0	1.0	0.0	11.0
2	0.0	0.0	0.0	1277.0	1374.0	1.0
3	0.0	0.0	0.0	8.0	0.0	1395.0

Otrzymane rezultaty za pomocą metody Birch

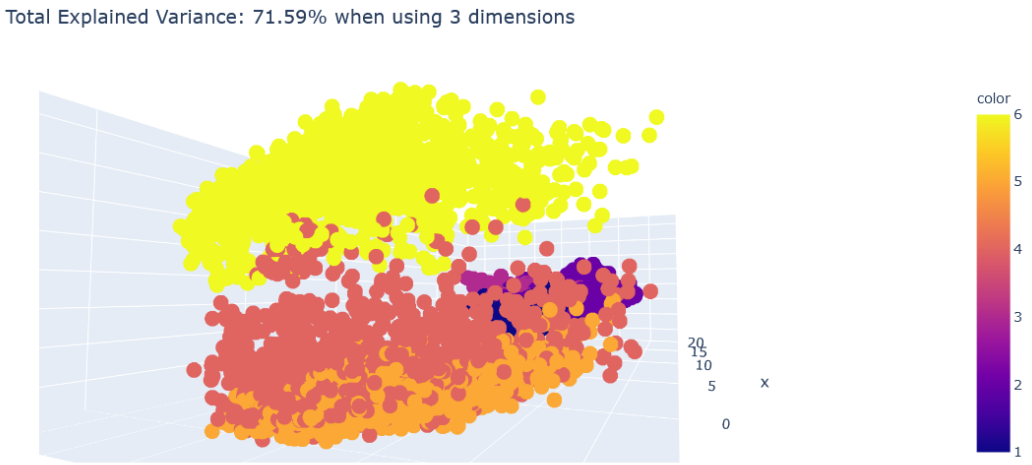
Właściwie dostaliśmy podziały bardzo podobne jak w przypadku algorytmu GMM z covariance_type "tied", oba algorytmy zadziałały podobnie i w obu wypadkach uznajemy to działanie za całkiem dobre.

6 Podsumowanie

Końcowo możemy przyjrzeć się jak wygląda oryginalny zbiór, aby pokazać jak wyglądałaby "idealna" klasteryzacja.



Wykres 3D obrazujący oryginalne labele



Wykres 3D obrazujący oryginalne labele, ale z innej perspektywy

Z tej perspektywy widzimy, że poprawnym zaetykietowaniem jest podzielenie klastra na trzy "poziome". Każdy z zastosowanych algorytmów nie wyodrębnił tych klastrów w pożądany sposób.

Jako finalny model bez dostępu do etykiet i możliwości sprawdzenia wyników zdecydowalibyśmy się na **KMeans**. Dostarczał on do tego dobrych przesłanek optycznych i zachowanie się metryk też go popierało.