WYDZIAŁ MATEMATYKI I NAUK INFORMACYJNYCH
POLITECHNIKI WARSZAWSKIEJ

Wstęp do uczenia maszynowego

projekt 2

raport

GRUPA 1
Jakub Fołtyn,
Paulina Jaszczuk

26.05.2021

1 Wstęp

Nasze dane to zapisy sesji użytkowników odwiedzających pewien sklep e-commerce. Naszym zadaniem jest odnalezienie klastrów w owym zbiorze, które wprowadziłyby pewne podgrupy użytkowników. Domyślnym, idealnym podziałem są 2 podgrupy: użytkownicy którzy dokonali jakiegoś zakupu oraz ci, którzy żadnego zakupu nie dokonali. Zbiór ten bowiem można też traktować jako zadanie klasyfikacji, wówczas powyższa informacja jest traktowana jako etykieta klas. Dane pochodzą ze strony archive.ics.uci.edu [link].

2 EDA

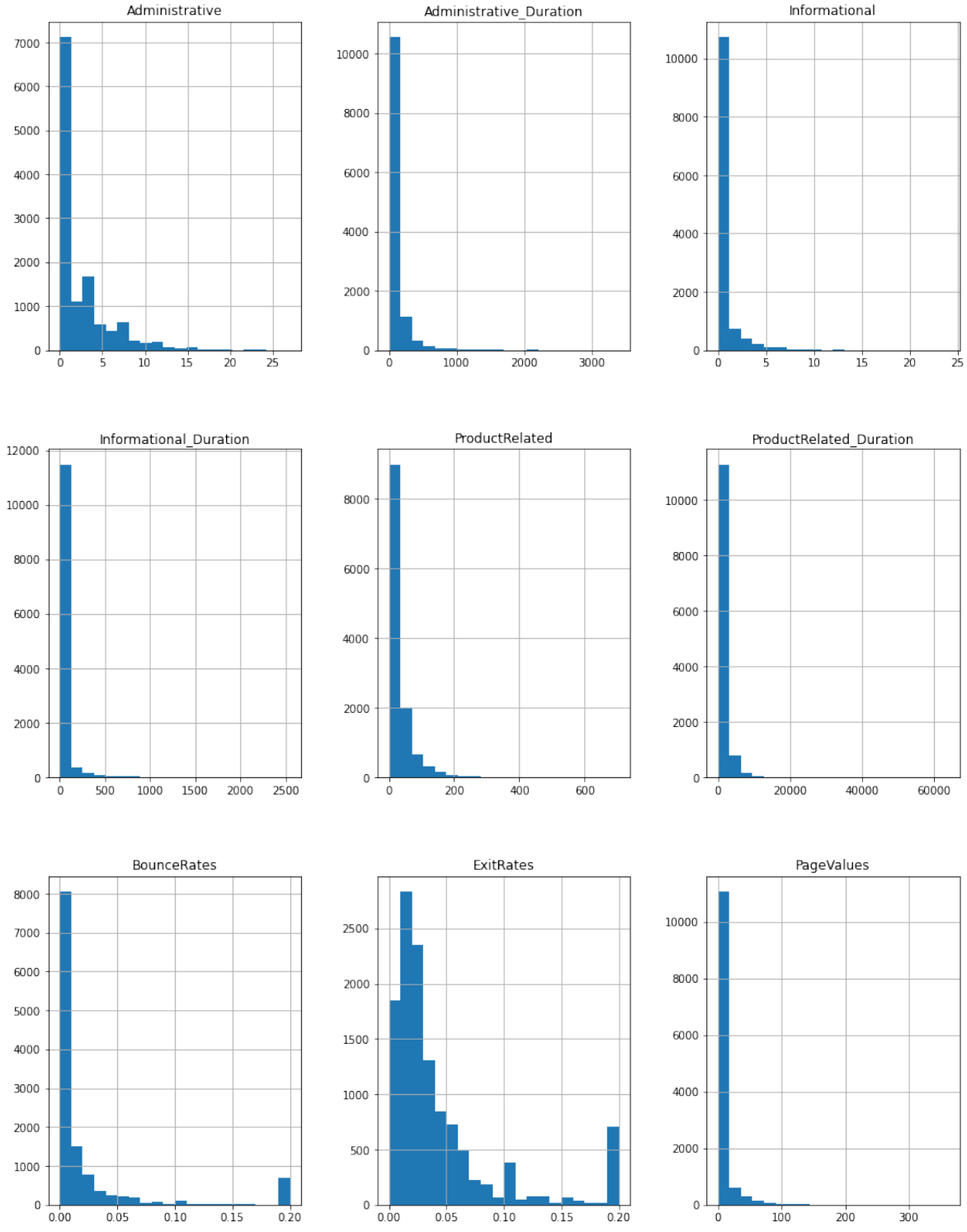
2.1 Ogólne informacje o danych

Dane dotyczą aktywności użytkowników na podstronach strony typu e-commerce. Każdy wiersz to osobna sesja osobnego użytkownika. Obserwacje zawierają dane dotyczące typu odwiedzonych podstron - zmienne typu *Page*, czasu spędzonego na nich - *Page_Duration*, pewnych współczynników charakteryzujących podstrony, np. *BounceRates*, a także informacje na temat samego użytkownika i sesji, m.in. *Operating_System* czy *SpecialDay*.

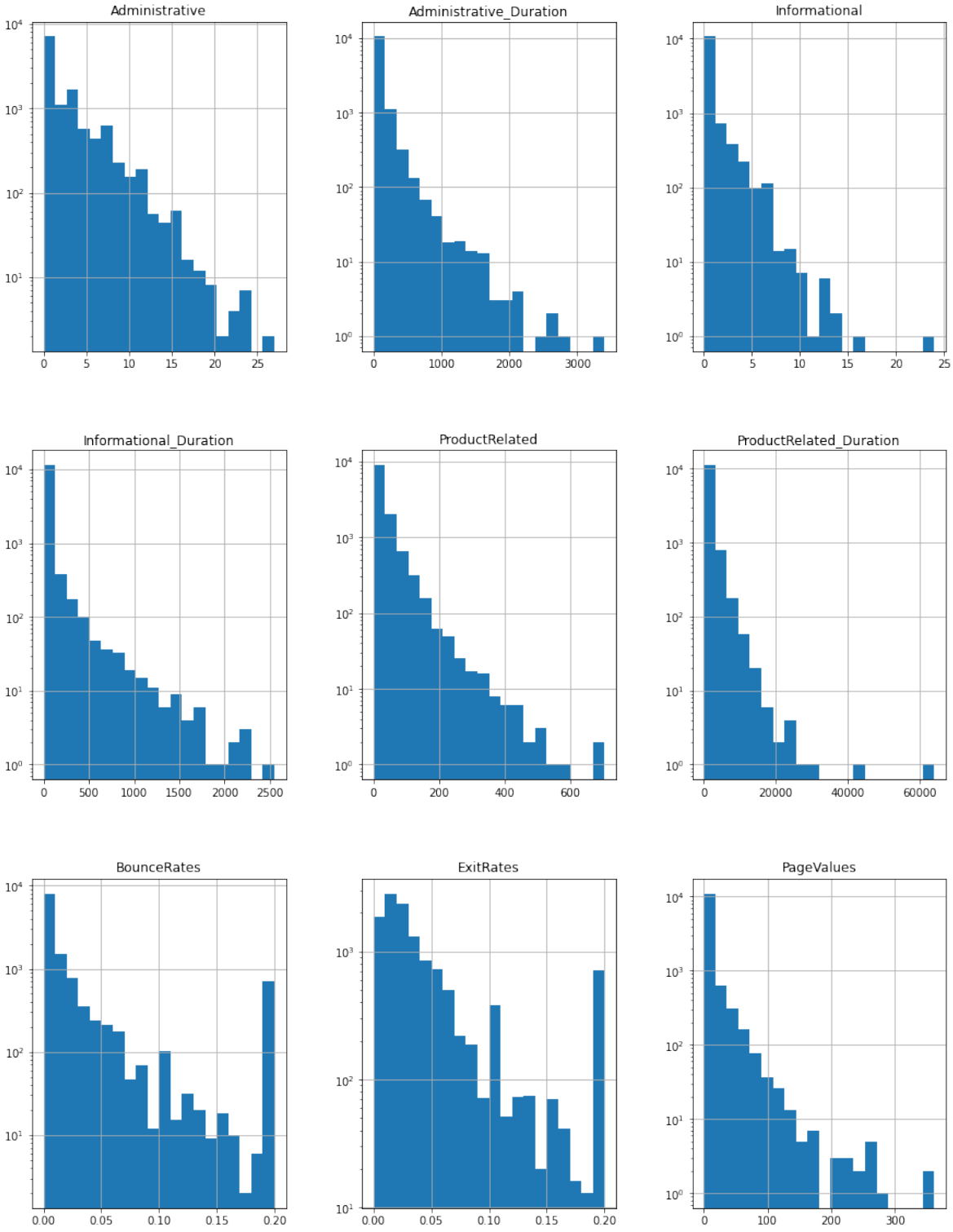
1. 12330 obserwacji (każda obserwacja to osobny użytkownik)
2. 3 cechy kategoryczne
3. 13 cech numerycznych (w tym 4 określające kategorie)
4. kolumna *Revenue* będąca targetem w zadaniu klasyfikacji, przez nas nieużywana podczas modelowania

2.2 Rozkłady zmiennych

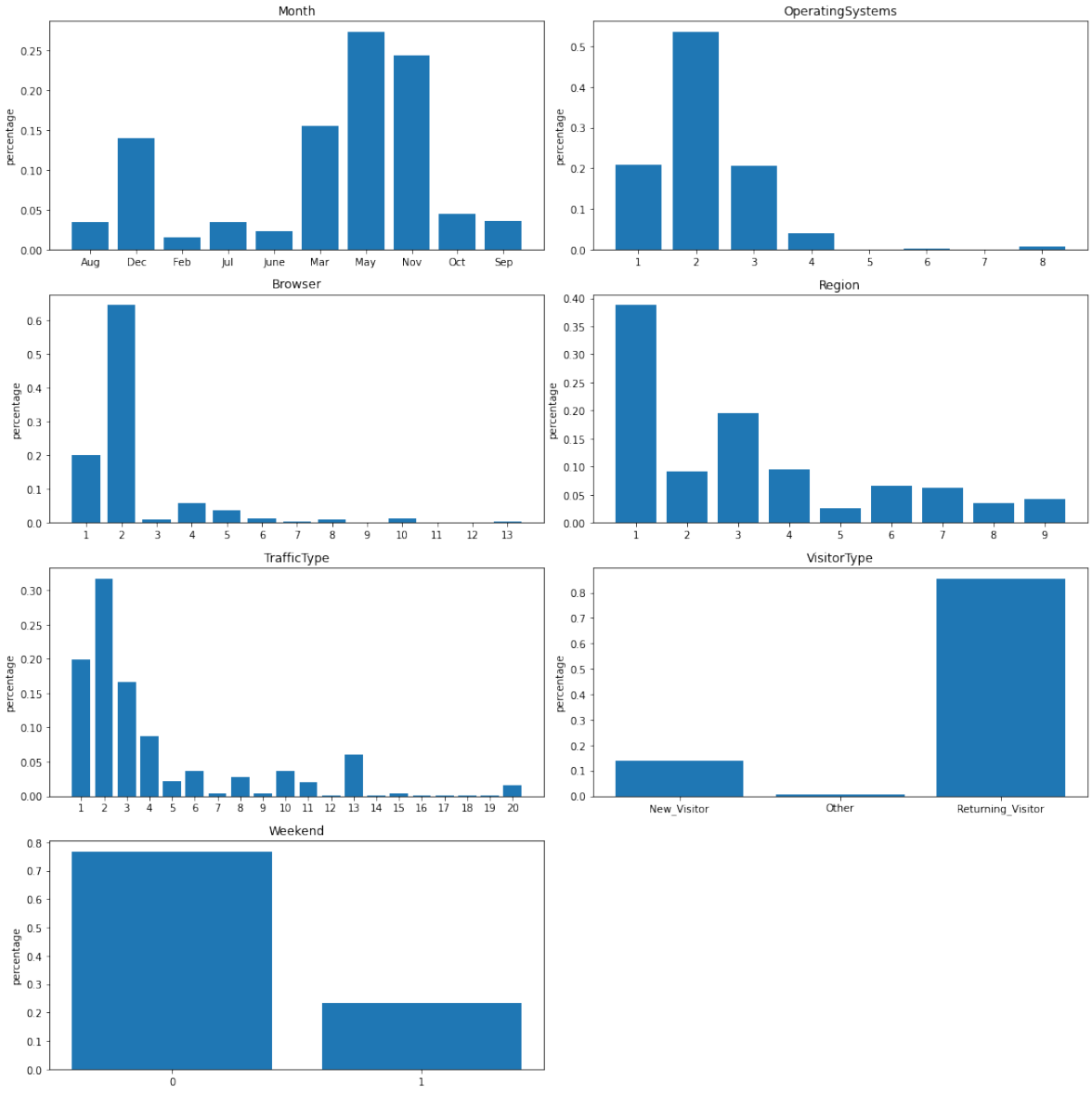
Rozkłady wszystkich zmiennych numerycznych były bardzo nierównomierne - wartość 0 występowała kilkanaście razy więcej niż inne (oprócz zmiennej *ExitRates*). Z tego powodu przygotowaliśmy również rozkłady danych zlogarytmowanych. Po tej operacji nasze dane miały rozkłady zbliżone do skośnych. W zmiennych *BounceRates* i *ExitRates* można zauważyć również pewne skupienie w wartości 0.2.
W przypadku większości zmiennych kategorycznych można zauważyć dominację jednej z wartości, jednak brak bardziej szczegółowego opisu danych uniemożliwia głębszą interpretację takich cech jak *Operating_System*, *Region* czy *Browser*. Można jednak stwierdzić, że większość sesji miała miejsce w maju i listopadzie, w dni nie weekendowe oraz były one dokonywane głównie przez użytkowników powracających.



Rysunek 1: Rozkłady zmiennych numerycznych.



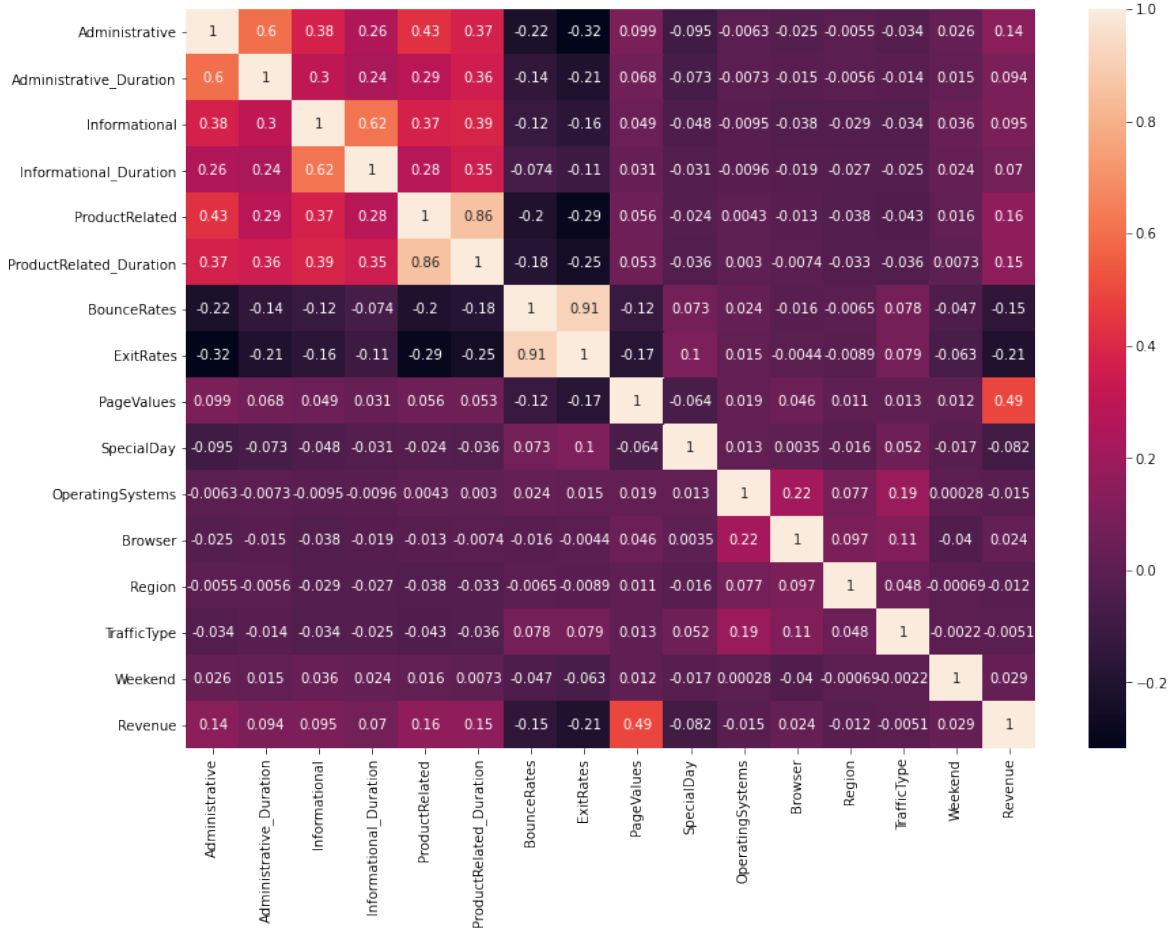
Rysunek 2: Rozkłady zlogarytmowanych zmiennych numerycznych.



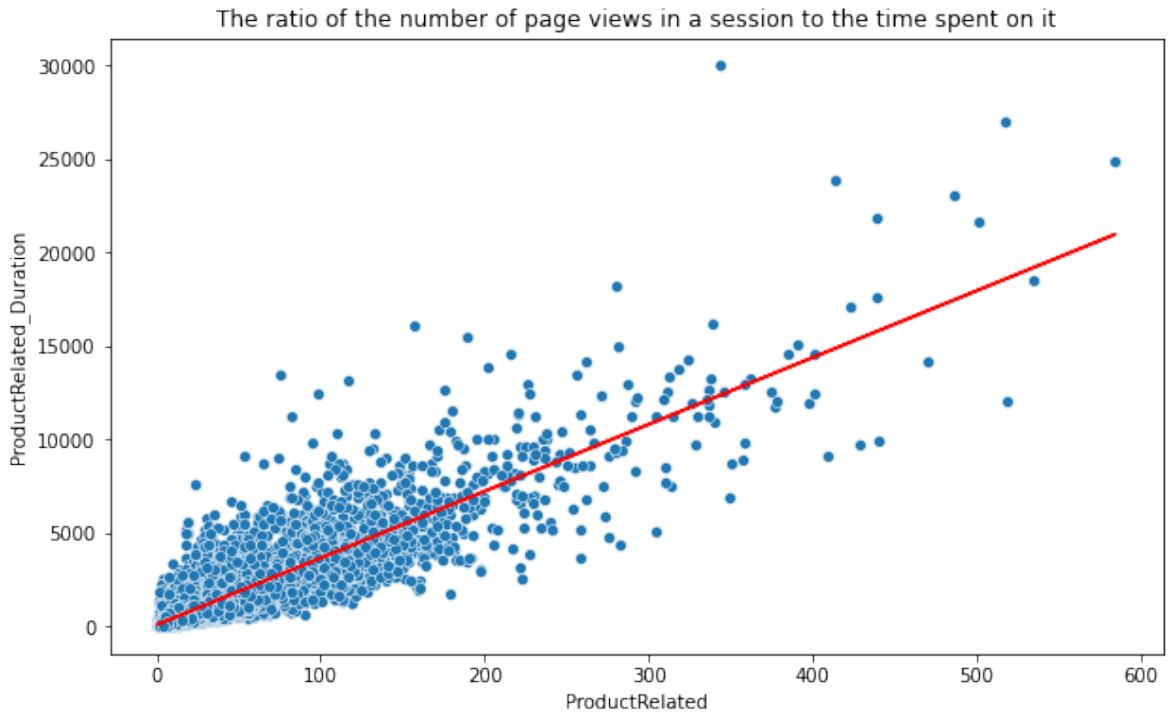
Rysunek 3: Rozkłady zmiennych kategorycznych.

2.3 Korelacja cech

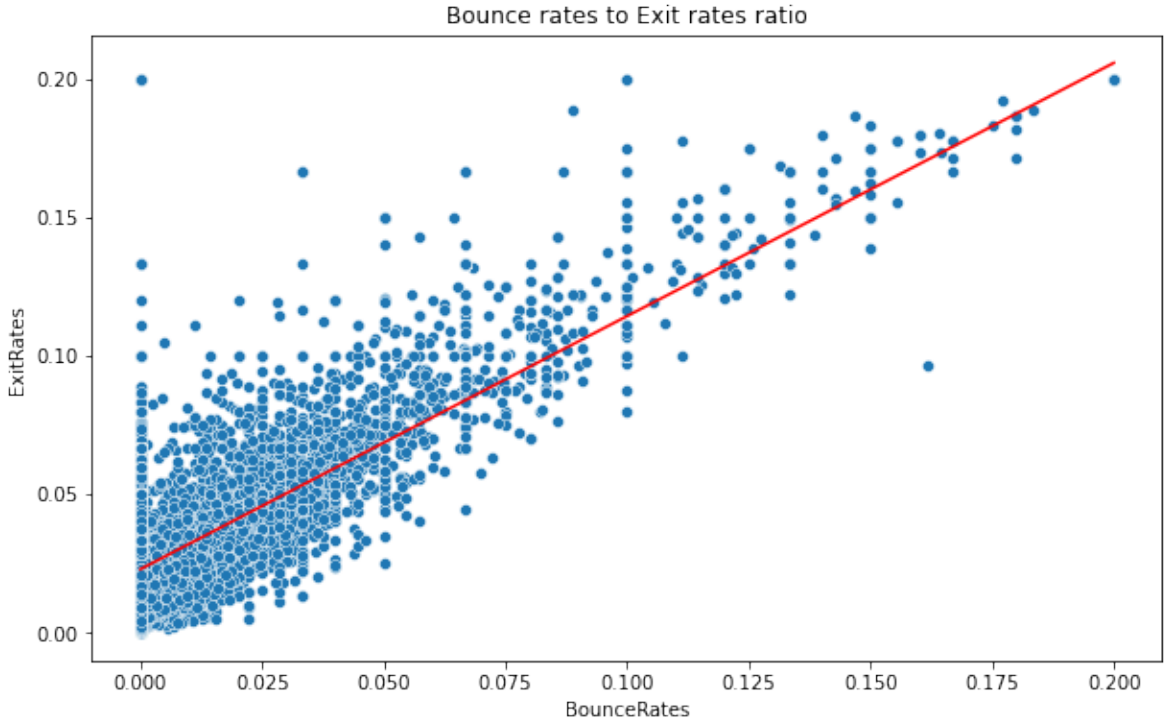
Analiza korelacji cech wykazała wysoką zależność między zmiennymi typu *Page* i *Page_Duration* oraz *ExitRate* i *BounceRates*. Pierwsza z korelacji jest dość intuicyjna i potwoerdza nasze przypuszczenia - im więcej odwiedzonych stron danego rodzaju, tym dłuższy czas na nich spędzony, zaś druga również jest logiczna po głębszym zgłębieniu danych - *BounceRates* jest specjalnym przypadkiem *ExitRates* - przy każdym wzroście pierwszej cechy rośnie też druga.



Rysunek 4: Macierz korelacji.



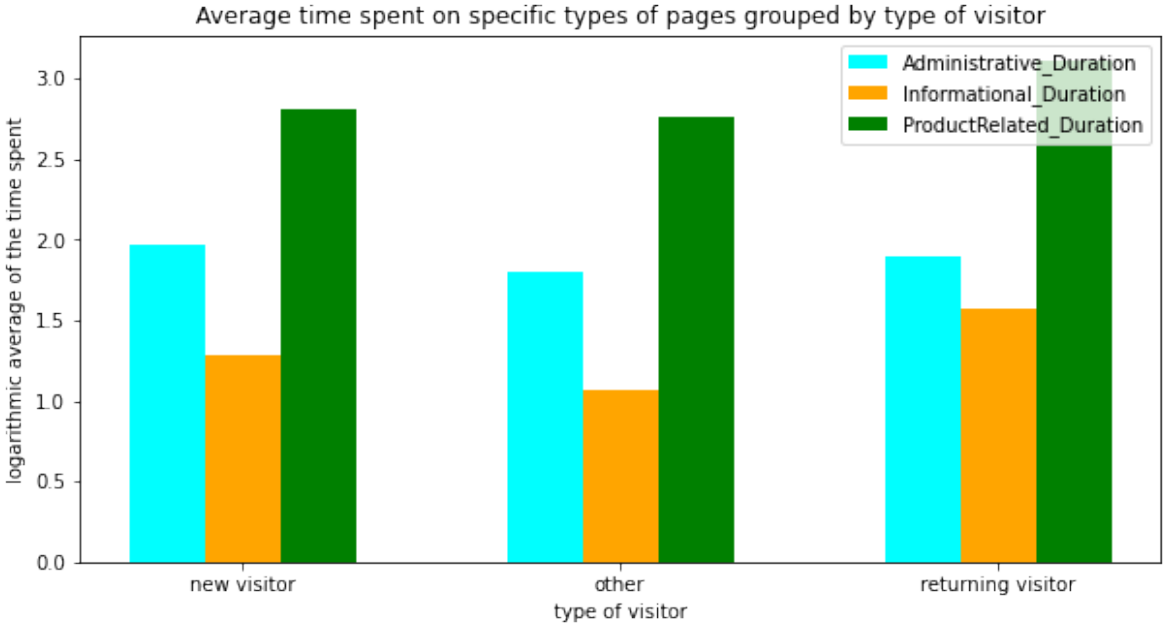
Rysunek 5: Wykres zależności zmiennych *ProductRelated* i *ProductRelated_Duration*.

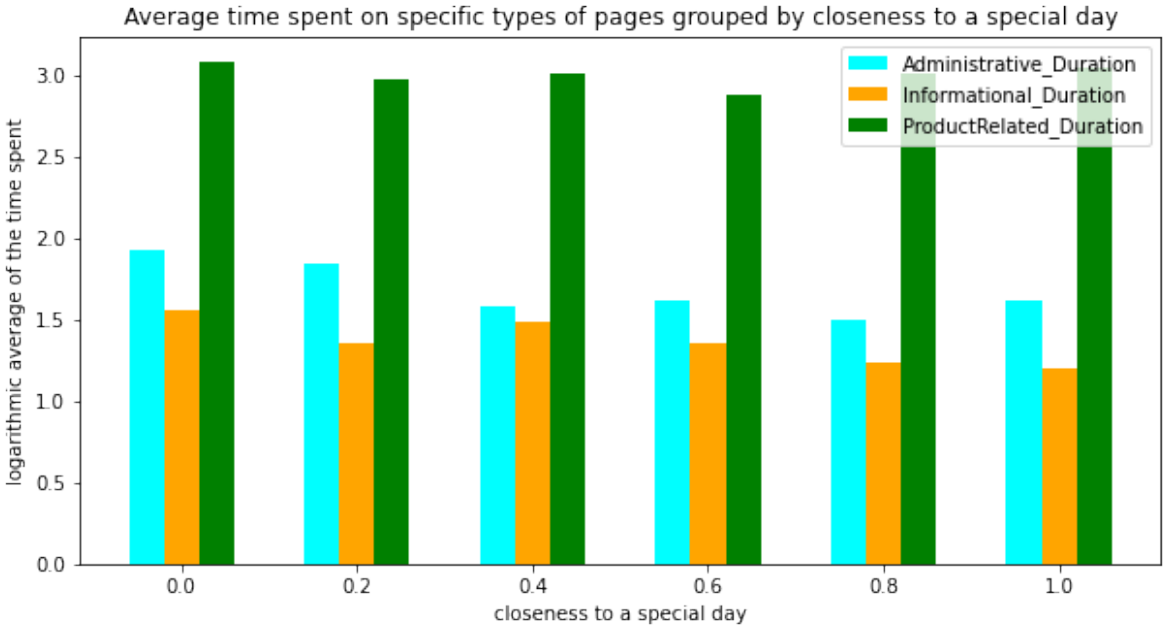


Rysunek 6: Wykres zależności zmiennych *BounceRates* i *ExitRates*.

2.4 Szczegółowa analiza

Rozważając czas spędzony na różnego rodzaju stronach w odniesieniu do konkretnych cech użytkownika i sesji, można zauważyć, że jest on niemal zawsze taki sam - klienci najwięcej czasu spędzają na stronach dotyczących produktów. Użytkownicy powracający nieco dłużej pozostają na podstronach informacyjnych, zaś wraz ze zbliżającymi się okazjami (rosnąca wartość współczynnika *SpecialDay*) nieco wzrasta czas spędzany na stronach dot. produktów kosztem innych.





3 Inżynieria cech

3.1 Kodowanie zmiennych kategorycznych

Mimo że w naszym zbiorze danych aż 7 kategorii ma charakter kategoryczny, to 4 z nich w praktyce są już zakodowane zmiennymi numerycznymi. Pozostałe 3 cechy kategoryczne zakodowaliśmy w następujący sposób:

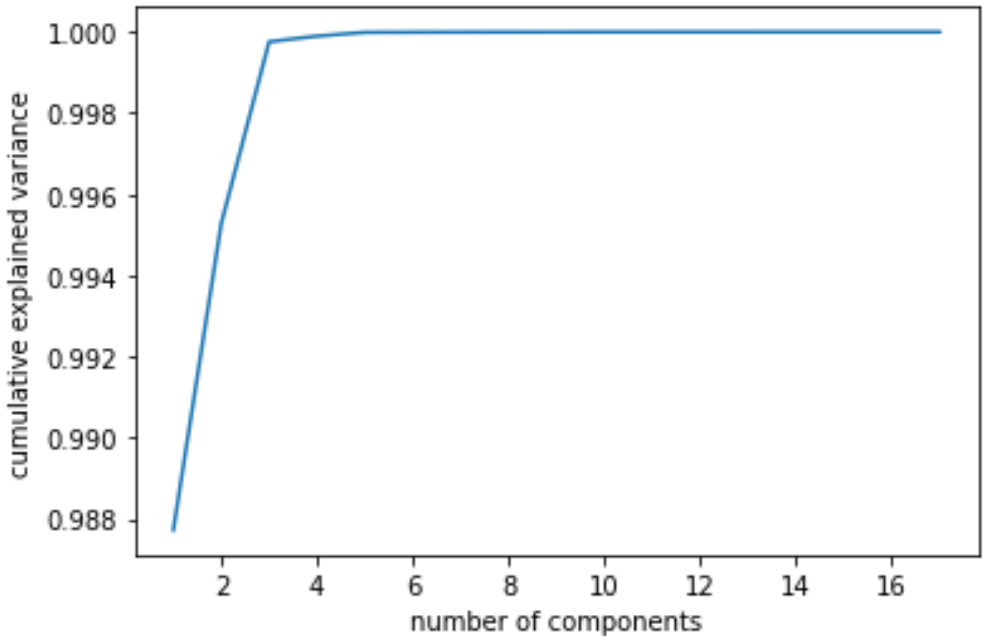
- zmienną *Weekend* zmapowaliśmy na wartości binarne (1 dla sesjii odbywającej się w weekend i 0 w drugim przypadku),
- do zakodowania *Visitor_Type* użyliśmy One Hot Encodingu, gdyż zmienna ta ma tylko 3 wartości,
- *Month*, ze względu na swój cykliczny charakter zakodowaliśmy przy użyciu enkodingu cyklicznego.

3.2 Transformacja logarytmiczna i standaryzacja

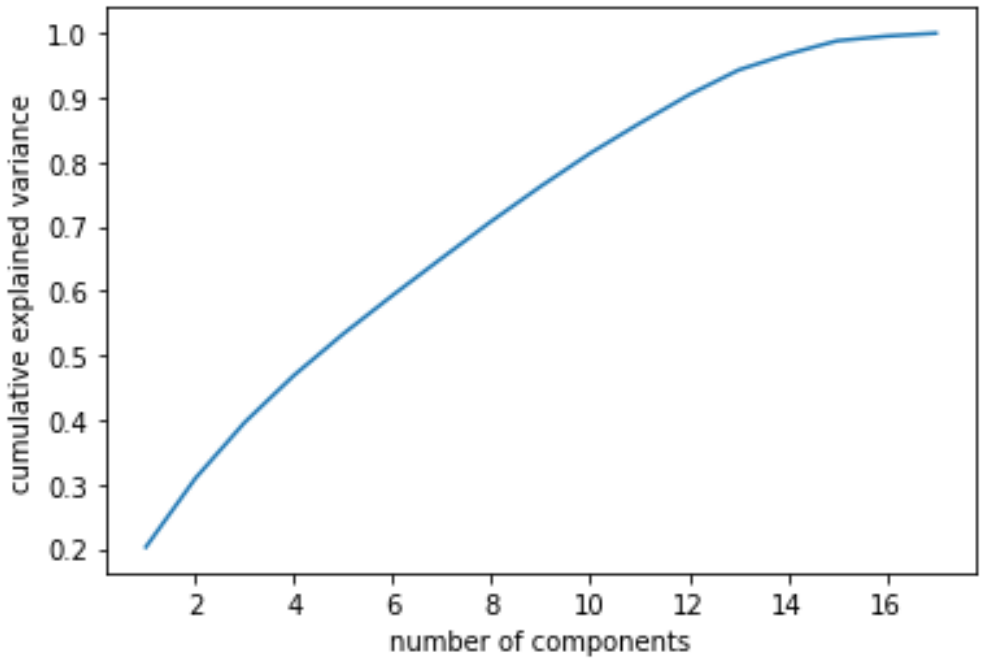
Z powodu nierównomiernego rozkładu danych utworzyliśmy w celach eksperymentalnych dodatkowe ramki danych: ze zlogarytmowanymi kolumnami dot. stron oraz ze zlogarytmowanymi kolumnami dot. stron i wskaźników stron, na których poźniej przeprowadzaliśmy wstępne modelowanie. Oprócz tego przeprowadziliśmy również standaryzację danych, co okazało się niezbędne do przeprowadzenia wiarygodnego procesu redukcji wymiarów przy pomocy PCA (opisane poniżej).

3.3 Redukcja wymiarów

W celu zredukowania wymiarowości naszych danych zastosowaliśmy algorytm PCA. Optymalną liczbę komponentów wyznaczaliśmy przy pomocy wykresu skumulowanej wyjaśnionej wariancji. Niestety, PCA przeprowadzone na danych "surowych" wskazało pewną anomalię - już jeden komponent wyjaśniał ponad 95% wariancji. Działo się tak głównie przez zmienne *ProductRelated_Duration*, *Inforamtional_Duration* i *Administrative_Duration*, które w najbardziej skrajnych przypadkach osiągały wartości nawet kilka tysięcy razy wyższe niż pozostałe kolumny, co najprawdopodobniej wpływało na algorytm PCA. Rozwiązaniem okazało się przeprowadzenie standaryzacji na danych. Jak widać na wykresach poniżej, PCA po standaryzacji daje już znacznie bardziej wiarygodne wyniki. Po analizie ów wykresu, ostatecznie zdecydowaliśmy się na ustawienie liczby komponentów na 12. (Pierwotnie nasze dane miały 17 wymiarów).



Rysunek 7: Wykres skumulowanej wyjaśnionej wariancji przed standaryzacją.

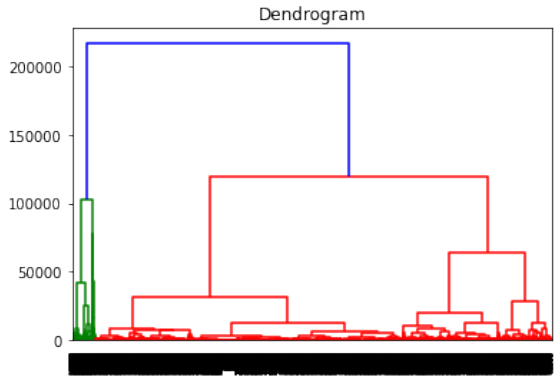


Rysunek 8: Wykres skumulowanej wyjaśnionej wariancji po standaryzacji.

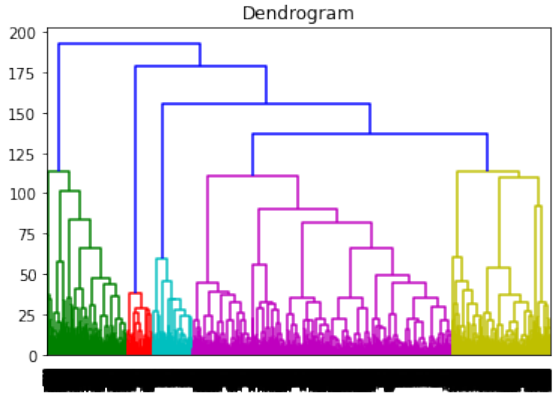
4 Modelowanie

4.1 Wybór optymalnej liczby klastrów

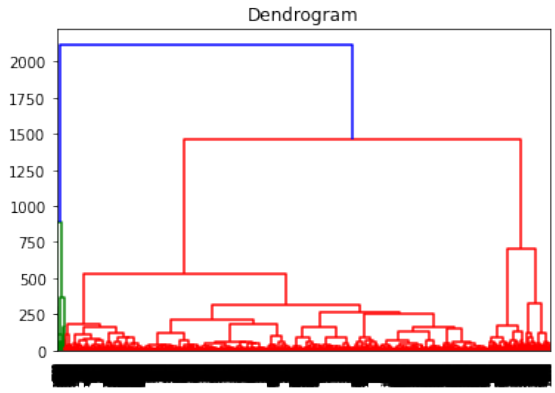
Wybór optymalnej liczby klastrów okazał się nie być trywialny. Analiza dendrogramów oraz wykresów łokciowych dawała rozbieżne wyniki. Dla danych surowych oraz zlogarytmowanych dość wyraźnie sugerowały one dwa klastry, natomiast dla danych po standaryzacji trudno było wyciągnąć jakiekolwiek wnioski. Finalnie zdecydowaliśmy się na wybór 2 klastrów, również dlatego, że w tym przypadku uzyskiwaliśmy najwyższy wynik współczynnika Silhuette score.



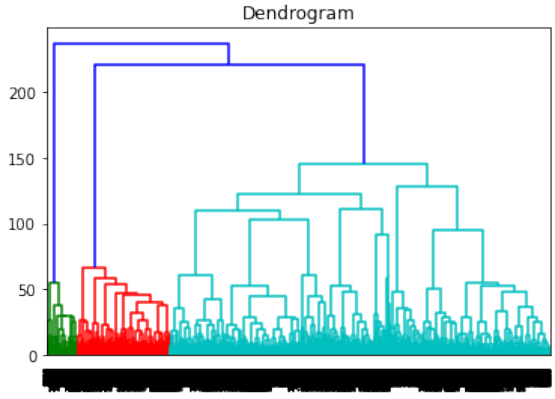
Rysunek 9: Dendrogram dla danych surowych



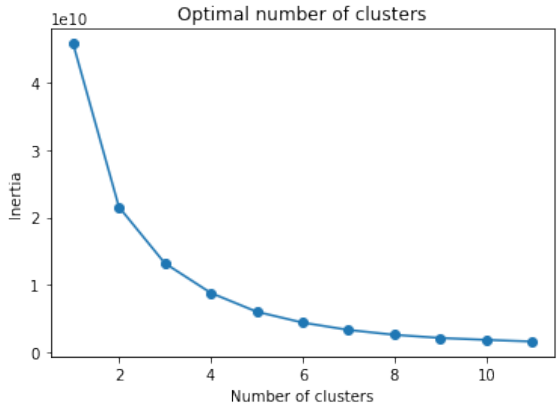
Rysunek 10: Dendrogram dla danych ustandaryzowanych



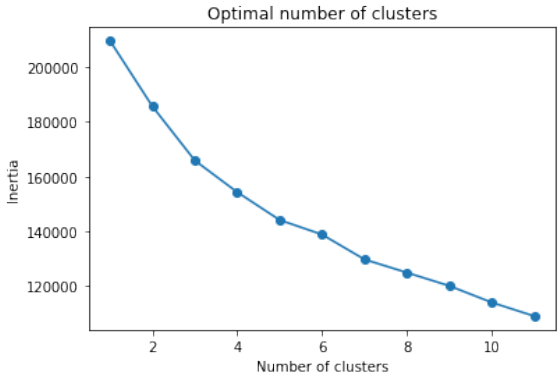
Rysunek 11: Dendrogram dla danych zlogarytmowanych



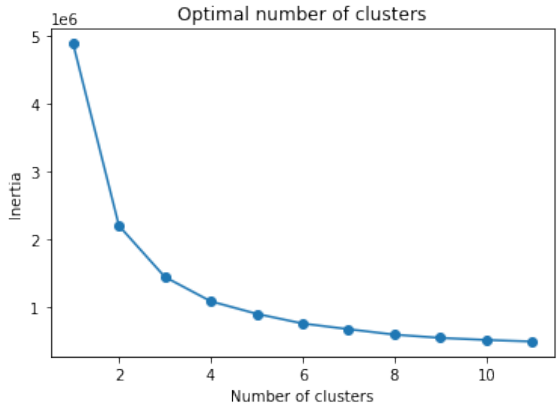
Rysunek 12: Dendrogram dla danych zlogarytmowanych ustandaryzowanych



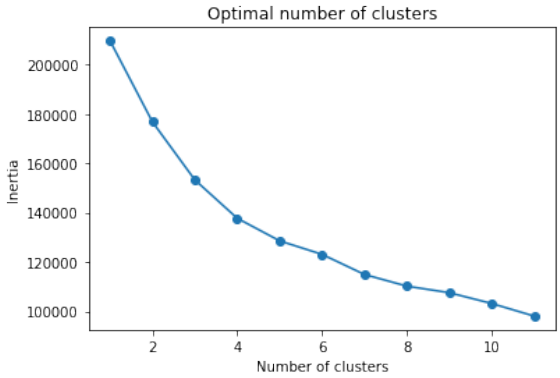
Rysunek 13: Wykres łokciowy dla danych surowych



Rysunek 14: Wykres łokciowy dla danych ustandaryzowanych



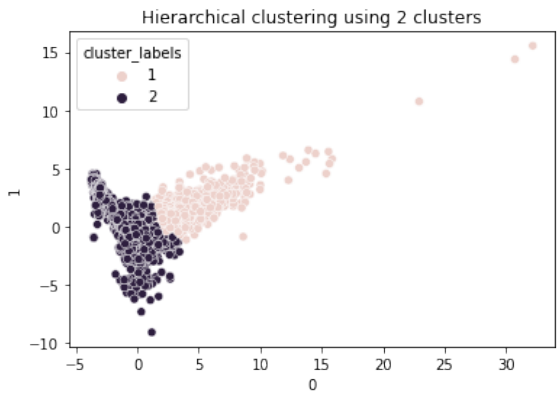
Rysunek 15: Wykres łokciowy dla danych zlogarytmowanych



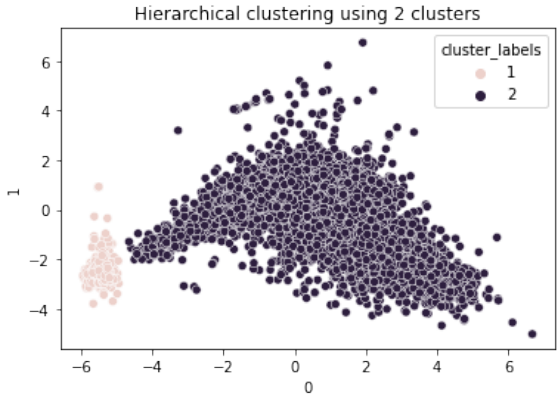
Rysunek 16: Wykres łokciowy dla danych zlogarytmowanych ustandaryzowanych

4.2 Wstępne modelowanie

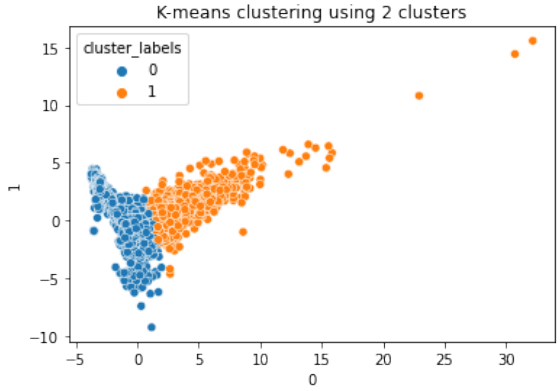
Wstępne modelowanie przeprowadziliśmy najprostszymi metodami: **Hierarchiczną** oraz **K-means**. Jako miarę jakości naszych klastrów wybraliśmy **Silhuette score**, który bierze pod uwagę zarówno średnie odległości wewnątrz klastra jak i średnie odległości pomiędzy klastrami. W ramach testów porównywaliśmy jakości klastrowania dla wszystkich ramek danych (tj. danych "surowych", wystandaryzowanych oraz zlogarytmowanych), a także dla różnych liczb komponentów w algorytmie PCA. Dla danych zredukowanych poprzed PCA do 2 wymiarów sporządziliśmy również wizualizację clusteringu. Najlepsze wyniki osiągało klastrowanie hierarchiczne na zlogarytmowanym zbiorze danych.



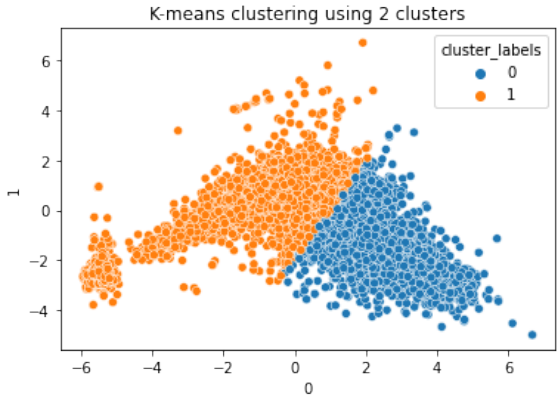
Rysunek 17: Clustering danych ustandaryzowanych



Rysunek 18: Clustering danych zlogarytmowanych ustandaryzowanych



Rysunek 19: Clustering danych ustandaryzowanych



Rysunek 20: Clustering danych zlogarytmowanych ustandaryzowanych

4.3 Końcowe modelowanie

4.3.1 Dyskusja

W końcowym modelowaniu postanowiliśmy wykorzystać zarówno znane nam, jak i nowe algorytmy. Najwiekszym naszym problemem pozostaje wymiarowość naszych danych - nie jesteśmy w stanie zmniejszyć jej do liczby wymiarów "przyjaznej" człowiekowi (2 lub 3) nie tracąc przy tym większości wariancji, przez co wszelkie wizualizacje efektów naszego klastrowania obarczone są błędem wynikającym z zastosowania algorytmów zmniejszających wymiarowość (błąd ten przekracza niestety granice akceptowalności), a co za tym idzie - nie jesteśmy w stanie stwierdzić, czy zastosowany przez nas algorytm zwraca "dobre" klastrowanie na podstawie jakże intuicyjnych wykresów - przez co jesteśmy zmuszeni polegać praktycznie jedynie na metrykach- w naszym przypadku są to *Silhuette score* czy *Calinski Harabasz score*.

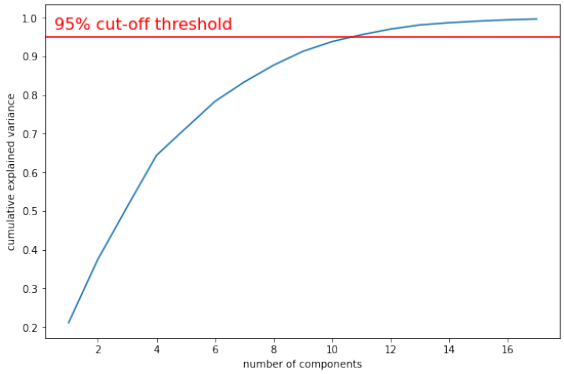
4.3.2 Dane

Podobnie jak w przypadku wstępnego modelowania, wybrane algorytmy stosujemy do różnych tabel danych. Biorąc pod uwagę wyniki ze wstępnego modelowania, postanowiliśmy skupić się na danych, w których 6 pierwszych kolumn zostało poddanych logarytmizacji - wyniki dla nich były bowiem najwyższe. Wciąż pamiętamy jednak o potrzebie standaryzacji danych, dlatego tym razem postanowiliśmy poddać naszą wybraną tabelę aż 3 rodzajom tejże, do czego wykorzystaliśmy:

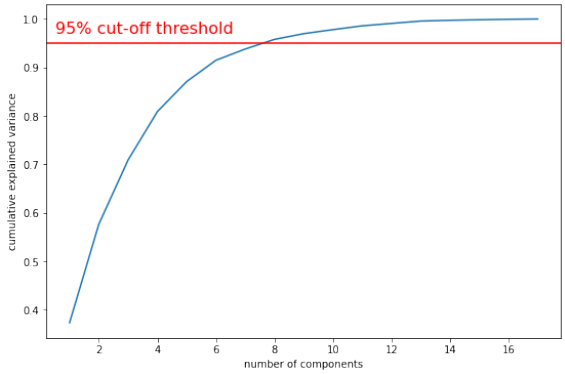
- *StandardScaler*
- *MinMaxScaler*
- *Normalizer*

Pozwoliło nam to w miarę bezpiecznie zredukować wymiarowość owych 3 tabel: do, odpowiednio, 14, 11 oraz 8 wymiarów (za pomocą algorytmu PCA, początkowo nasze dane miały 17 wymiarów). Tym sposobem w kazdej z tych tabel wyjaśnione jest ok. 95% wariancji. Podsumowując, mamy więc 8 tabel na których testujemy różne algorytmy:

- *log*, w której 6 pierwszych kolumn poddaliśmy logarytmizacji.
- *log_standard*, dodatkowo zostały przetransformowane *StandardScaler'em*.
- *log_minmax*, które dodatkowo zostały przetransformowane *MinMaxScaler'em*.
- *log_normal*, które dodatkowo zostały przetransformowane *Normalizer'em*.
- *log_standard_pca14*, które dodatkowo zostały przetransformowane *StandardScaler'em* i zredukowane do 14 wymiarów.
- *log_minmax_pca11*, które dodatkowo zostały przetransformowane *MinMaxScaler'em* i zredukowane do 11 wymiarów.
- *log_normal_pca8*, które dodatkowo zostały przetransformowane *Normalizer'em* i zredukowane do 8 wymiarów.
- *org*, czyli dane oryginalne (poddane tylko encodingowi)- jako pewien punkt wyjścia, sprawdzający, czy nasze operacje na danych przynoszą zamierzone efekty



Rysunek 21: Wykres skumulowanej wyjaśnionej wariancji dla danych przetransformowanych przy użyciu MinMaxScalera



Rysunek 22: Wykres skumulowanej wyjaśnionej wariancji dla danych przetransformowanych przy użyciu Normalizera

4.3.3 Algorytmy

W naszym klastrowaniu korzystamy z nastepujących algorytmów:

- *HDBScaner*, w którym stroiliśmy parametr *min_cluster_size*
- *Hierarchy clustering*
- *Kmeans*
- *DBScan*
- *GMM*

Być może warto zatrzymać się nad algorytmem *HDBScaner*, który nie był omawiany na zajęciach. Jak sugeruje nazwa, jest on ściśle związany ze znanym nam *DBScan'em*. W dużym uproszczeniu - jest to *DBScan* z zelementami hierarchicznymi - algorytm łączy "wyspy" punktów w coraz bardziej "gęste" drzewa. Minimalna wielkość takiej "wyspy" określana jest przez parametr *min_cluster_size*, dlatego też strojenie go jest tak istotne w tym algorytmie.

4.3.4 Wyniki

Po wypróbowaniu wielu różnych algorytmów i przekształceń, ku naszemu zdziwieniu, okazało się, że najlepiej sprawdzają się 2 najprostsze algorytmy: *kmeans* oraz *hierarchy clustering*, zastosowane na danych ze zlogarytmizowanymi pierwszymi 6 kolumnami.

	silhuette	calinski_harabasz
org	0.834663	11187.208055
log	0.759619	14535.028942
log_std	0.301841	1600.661676
log_minmax	0.296635	2783.141908
log_normal	0.409325	5700.764995
log_normal_pca8	0.390145	6146.579736
log_minmax_pca11	0.307439	2943.188531
log_std_pca14	0.310649	1652.166015

Rysunek 23: Wartości miar jakości klasteringu hierarchicznego dla różnych postaci danych.

	silhuette	calinski_harabasz
org	0.718966	12014.241283
log	0.784008	15717.850131
log_std	0.138576	1942.923615
log_minmax	0.304165	2850.523800
log_normal	0.403050	6218.947277
log_normal_pca8	0.419109	6636.408046
log_minmax_pca11	0.314170	3015.950189
log_std_pca14	0.145664	2034.453599

Rysunek 24: Wartości miar jakości klasteringu metodą kmeans dla różnych postaci danych.

	silhuette	calinski_harabasz
org	-0.360560	2.912319
log	-0.413713	2.490451
log_std	0.196189	29.638740
log_minmax	-0.411708	29.474116
log_normal	-0.247189	26.671063
log_normal_pca8	-0.148652	34.057205
log_minmax_pca11	-0.274757	67.636045
log_std_pca14	-0.370275	14.951245

Rysunek 25: Wartości miar jakości klasteringu metodą DBSCAN (eps=0.2) dla różnych postaci danych.

	silhuette	calinski_harabasz
org	0.245719	2681.672502
log	-0.099050	311.734271
log_std	0.129037	1729.161446
log_minmax	0.205116	2068.004450
log_normal	0.213799	2962.656163
log_normal_pca8	0.235992	3292.983768
log_minmax_pca11	0.162638	1631.284643
log_std_pca14	0.135004	1799.268020

Rysunek 26: Wartości miar jakości klasteringu metodą GMM dla różnych postaci danych.

	silhuette	calinski_harabasz
org	-0.583551	27.103903
log	-0.532322	34.829208
log_std	-0.271612	63.546249
log_minmax	-0.139957	157.974129
log_normal	-0.381067	55.502787
log_normal_pca8	-0.310026	45.888981
log_minmax_pca11	-0.215696	63.712303
log_std_pca14	-0.257373	47.695656

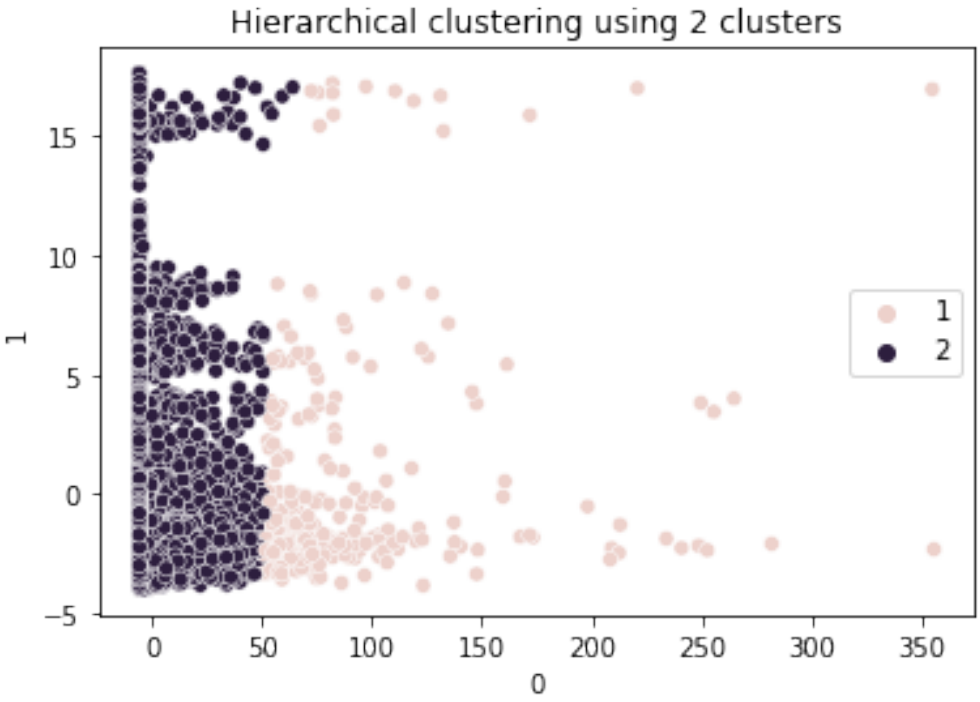
Rysunek 27: Wartości miar jakości klasteringu metodą HDBSCAN (min_cluster_size5) dla różnych postaci danych.

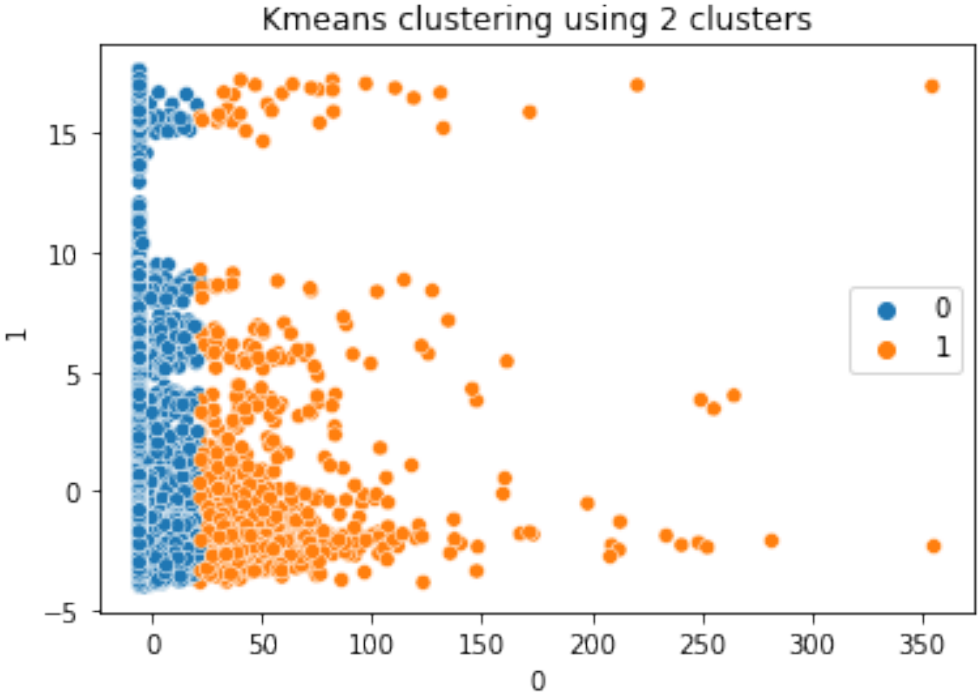
	silhuette	calinski_harabasz
org	-0.640194	39.606864
log	0.432437	3267.039793
log_std	0.580922	339.866014
log_minmax	0.206832	1104.272094
log_normal	0.043220	930.058314
log_normal_pca8	-0.118889	480.165841
log_minmax_pca11	0.102193	739.665281
log_std_pca14	0.589478	354.918082

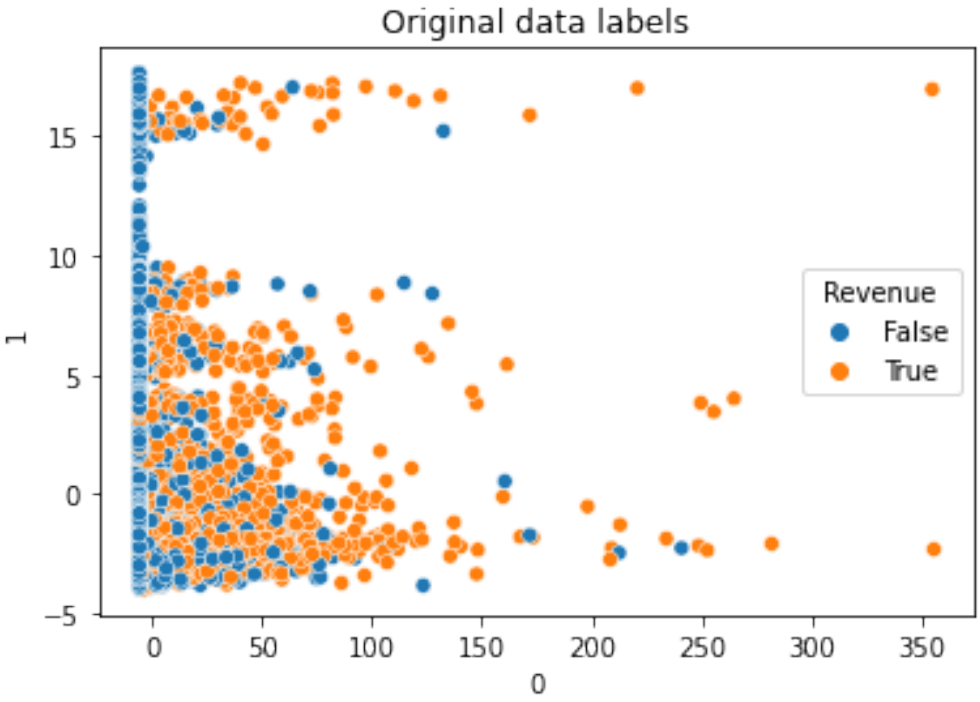
Rysunek 28: Wartości miar jakości klasteringu metodą HDBSCAN (min_cluster_size20) dla różnych postaci danych.

4.4 Wizualizacje

Próba wizualizacji zklastrowanych danych nie spełniła naszych oczekiwań. Scatterploty wyraźnie różniły się od siebie. Taka sytuacja mogła mieć dwie przyczyny - albo nasza klasteryzaca nie była dość dobra, albo tak drastyczna redukcja wymiarowości (do 2 lub 3 wymiarów) pociągnęła za sobą wiele błędów widocznych na wizualizacji.


Hierarchical clustering using 2 clusters
15
10
5
0
−5
1
0
50
100
150
200
250
300
350
0
1
2


Kmeans clustering using 2 clusters
15
10
5
0
−5
1
0
50
100
150
200
250
300
350
0
0
1



4.5 Porównanie z oryginalnymi etykietami

Miary takie jak *Silhuette Score* czy *Calinski Harabasz Score* oraz wizualizacje pomogły nam wybrać model i formę danych, natomiast nie odpowiedziały nam one w pełni na pytanie o jakość klasteryzacji. W tym celu porównaliśmy finalne etykiety nadane danym w procesie clusteringu z tymi oryginalnymi. Okazało się, że klastering hierarchiczny przypisał do poprawnej grupy 89% obserwacji, zaś kmeans - 88%.

5 Podsumowanie

Zadanie klasteryzacji użytkowników sklepów e-commerce jest prawdopodobnie jednym z najpopularniejszych, a zarazem najprzydatniejszych zadań Uczenia Maszynowego. Poprawnie skonstruowany algorytm będzie w stanie wskazać, w jaki sposób strona może zostać zmodyfikowana, by zachęcić jak największy odsetek odwiedzających do zakończenia swojej wizyty zakupem. W naszym projekcie skupiliśmy się na rozdzieleniu użytkowników kupujących od użytkowników odwiedzających na podstawie samych danych dotyczących ich sesji. Zadanie to okazało się nie być łatwym- dane nie chciały się "zredukować" do niskich wymiarów, co bardzo utrudniało wizualizację, a co za tym idzie- intuicyjne "rozeznanie" w danych. Z tego też powodu zastosowaliśmy wiele mniej lub bardziej skomplikowanych algorytmów- i, ku naszemu zdziwieniu, najlepsze wyniki dały algorytmy najprostsze (*Kmeans* i *Hierarchy clustering*), ponadto zastosowane na

praktycznie nieprzetworzonych danych (poddanych tylko logarytmizacji). Jest to- naszym zdaniem- najlepszy przykład na to, że rozwiązania najprostsze (które mogą być nawet pomijane ze względu na swoją prostotę) czasem okazują się tymi najlepszymi.