WYDZIAŁ MATEMATYKI I NAUK INFORMACYJNYCH
POLITECHNIKI WARSZAWSKIEJ

Klasteryzacja sesji klienckich na podstawie danych z wizyt w serwisach e-commerce

---

WSTĘP DO UCZENIA MASZYNOWEGO
PROJEKT 2

Mateusz Krzyziński, Tomasz Nocoń, Paweł Fijałkowski

Grupa laboratoryjna nr 2

28.05.2021

1 Opis problemu

Praca ta jest raportem z przebiegu projektu 2 z przedmiotu Wstęp do Uczenia Maszynowego w semestrze letnim 2020/2021. Tematem zadania projektowego było przygotowanie realizującego zadane potrzeby biznesowe, narzędzia (algorytmu) klasteryzacji. Przebieg pracy nad projektem został podzielony na trzy etapy:

- eksplorację danych – EDA,
- inżynierię cech i wstępne modelowanie,
- przygotowanie finalnych modeli, wnioskowanie.

W naszym przypadku, celem było wskazanie podziału sesji klienckich na rozróżnialne w rozumieniu zbioru danych grupy. Zależało nam na ekstrahowaniu 8-12 grup klienckich, charakteryzujących się danymi, interpretowalnymi cechami. Aby nie wprowadzać subiektywizmów w pracę naszego modelu, nie nakładaliśmy dodatkowych założeń na pracę modelu (nie wybieraliśmy cech charakterystycznych grup).

Zadanie zostało wykonane w języku `Python` z użyciem biblioteki do uczenia maszynowego `Scikit-learn`. Poszczególne etapy projektu można znaleźć w repozytorium przedmiotu.

2 Opis zbioru danych

Zbiór danych - *Online shoppers intention* - zawiera informacje dotyczące statystyk związanych z sesją klienta przeglądarki na stronach e-commerce. Pojedynczy wiersz reprezentuję jedno *cookie*.

Zbiór danych składa się z:

- 12 330 wierszy,
- 18 kolumn,

Braki danych nie występują.

2.1 Informacje zawarte w zbiorze danych

Opiszemy poniżej zmienne (kolumny) znajdujące się w opisanym wyżej zbiorze danych.

2.1.1 Zmienne wyznaczane na podstawie URI hosta

- `Administrative` - ilość odwiedzonych stron o charakterze administracyjnym w trakcie trwania sesji.
- `AdministrativeDuration` - czas przebywania na stronach o charakterze administracyjnym.
- `Informational` - ilość odwiedzonych stron o charakterze informacyjnym w trakcie trwania sesji.

- `InformationalDuration` - czas przebywania na stronach o charakterze informacyjnym.
- `ProductRelated` - ilość odwiedzonych stron związanych z produktem w trakcie trwania sesji.
- `ProductRelatedDuration` - czas przebywania na stronach związanych z produktem.

2.1.2 Zmienne pochodzące ze statystyk Google Analytics

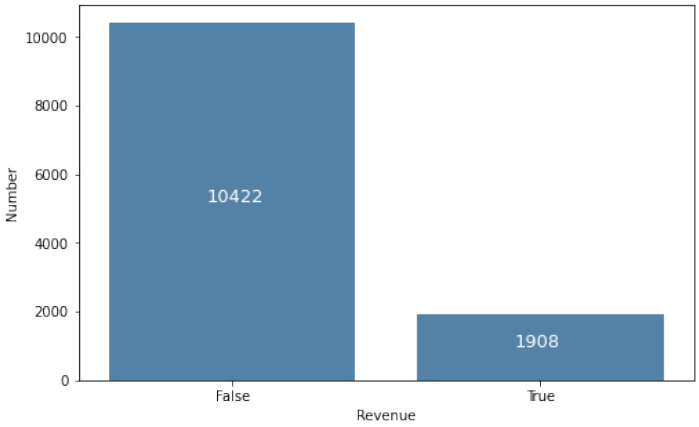
- `BounceRates` – średnia po BounceRate kazdej odwiedzanej w trakcie sesji strony. BounceRate dla danej strony to stosunek uzytkownikow pasywnych do wszystkich
- `ExitRates` – średnia po ExitRate każdej odwiedzanej strony. Exit rate to stosunek wyświetleń strony jako ostatniej sesji, do wszystkich wyświetleń
- `PageValues` – średnia po PageValues każdej odwiedzane strony. Ilość odwiedzeń danej strony przed dokonaniem transakcji

2.2 Zmienne dodatkowe

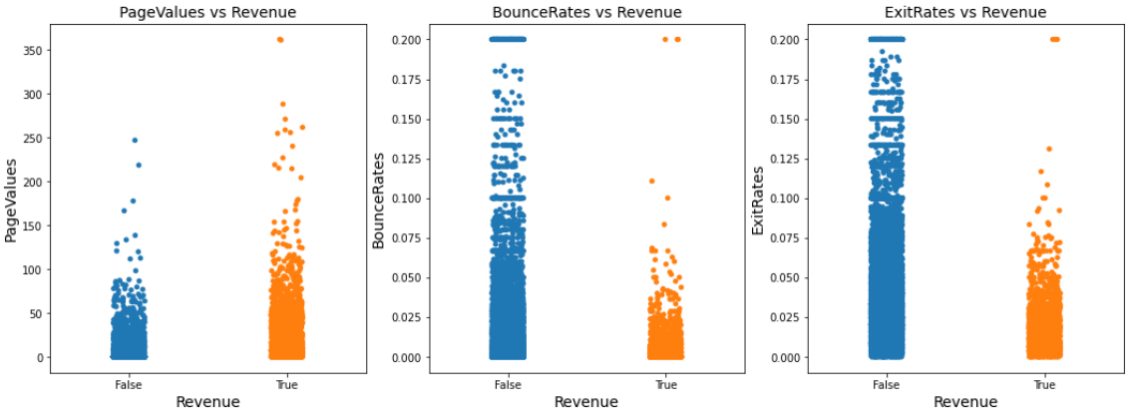
- `SpecialDay` – "bliskość" dnia realizacji sesji do święta (np. dzień matki czy walentynki)
- `Month` – miesiąc w którym nastąpiła dana sesja
- `OperatingSystems` – system operacyjny klienta
- `Browser` – przeglądarka klienta
- `Region` – region z którego nawiązano połączenie
- `TrafficType` – rodzaj ruchu, określony zmienną kategotyczną liczbową
- `VisitorType` – typ klienta (`ReturningVisitor`, `NewVisitor`, `Other`)
- `Weekend` – czy sesja odbyła się w weekend
- `Revenue` – czy sesja przyniosła aplikacji hostującej dochody

2.3 Wybrane wizualizacje dotyczące zbioru danych

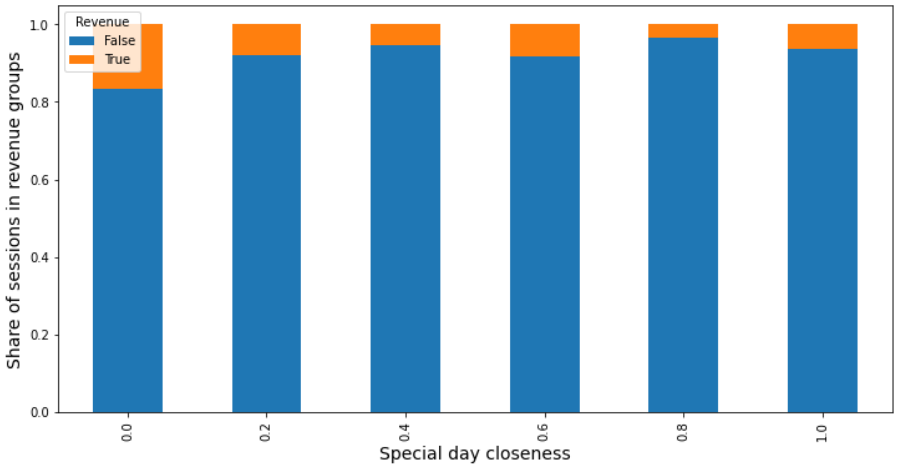
Zauważmy, że z racji na ogólny charakter naszych poszukiwań, nie mamy do dyspozycji zmiennej celu (tak jak w standardowym problemie uczenia maszynowego). Wyróżniamy jednak zmienną - `Revenue`, której rozkład w podziale na inne statystyki, może być najbardziej istotny dla naszej analizy. W związku z powyższym, prezentujemy wybrane wykresy dotyczące właśnie zmiennej `Revenue` oraz jej zależności do innych zmiennych znajdujących się w zbiorze.



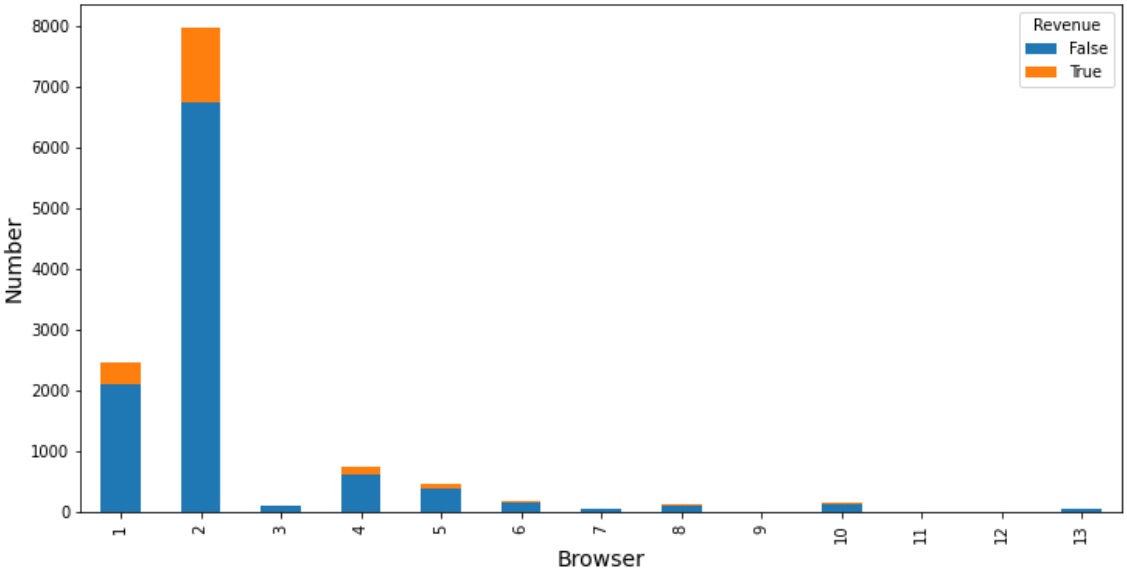
Rysunek 1: Liczności poszczególnych wartości zmiennej `Revenue`.



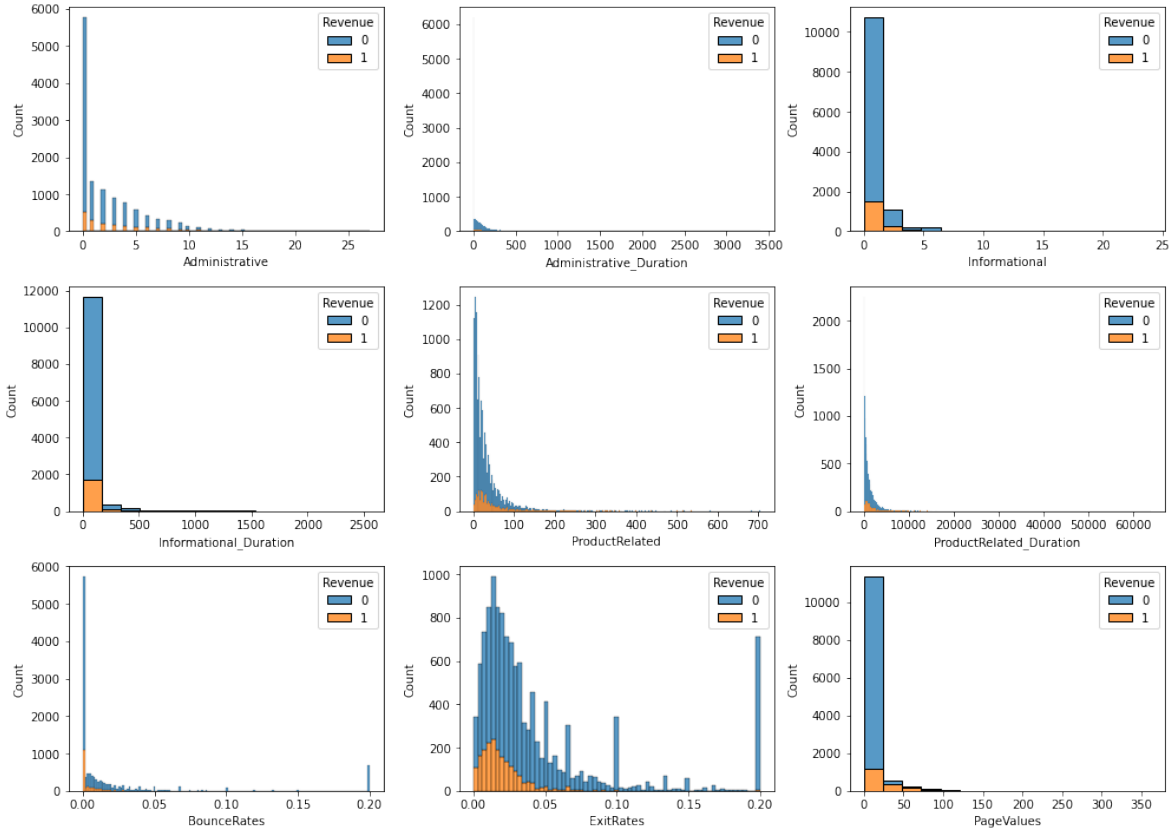
Rysunek 2: Rozkład zmiennej `Revenue` w podziale na statystyki pochodzące z usługi Google Analytics.



Rysunek 3: Rozkład zmiennej `Revenue` w podziale bliskość dni świątecznych. Rozkład wydaje się mało intuicyjny, udział pomarańczowej części słupka nie zwiększa się przy zwiększaniu wartości bliskości.



Rysunek 4: Rozkład zmiennej `Revenue` w podziale typ przeglądarki klienta. Widzimy istotną przewagę pewnej grupy przeglądarek (domyślamy się - Chrome, Firefox). Na dalszym etapie pracy dokonamy mapowań grup o mniejszych licznościach w jedną "inne".



Rysunek 5: Rozkłady zmiennych liczbowych w podziale na wartość `Revenue`. Na dalszym etapie pracy wykonamy transformację logarytmiczną tych zmiennych.

3 Inżynieria cech

Bazując na rzeczywistych danych, często nie lada problemem jest dobrać zestaw przekształceń danych w ten sposób, aby odseparować obserwacje od siebie. Mierząc się z tym wyzwaniem, przyjęliśmy strategię eksperymentalną, zakładającą różne możliwe przekształcenia zmiennych.

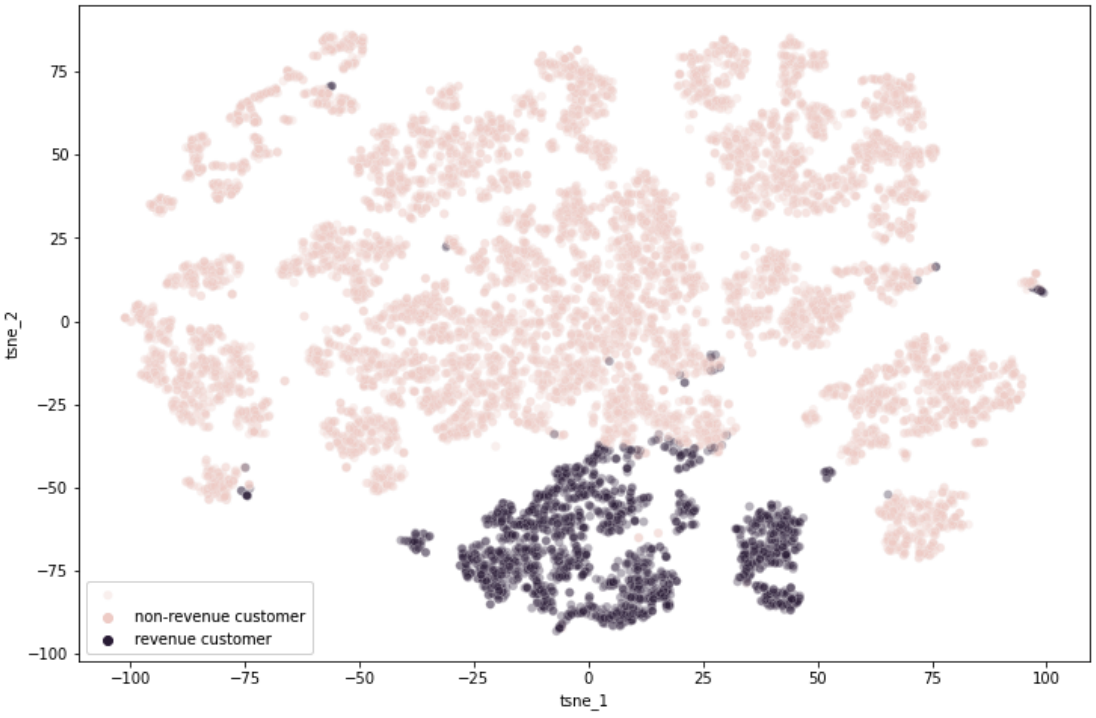
Dane zostały poddane różnorakim kombinacjom przekształceń z podanego poniżej zbioru:

- Zakodowanie zmiennych kategorycznych jako numerycznych

  Z uwagi na charakterystykę algorytmów klasteryzacyjnych (potrzebują zdefiniowanych metryk porównywania odległości), wszystkie zmienne kategoryczne (`Month`, `VisitorType`, `Weekend` i `Revenue`) zostały zakodowane jako liczbowe. `Month` dodatkowo została zakodowana przy użyciu współrzędnych biegunowych (polarnych), celem zachowania naturalnej cykliczości i "dystansu".

- Standaryzacja, przekształcająca rozkład zmiennej liczbowej na taki o wartości oczekiwanej 0 i odchyleniu standardowym 1

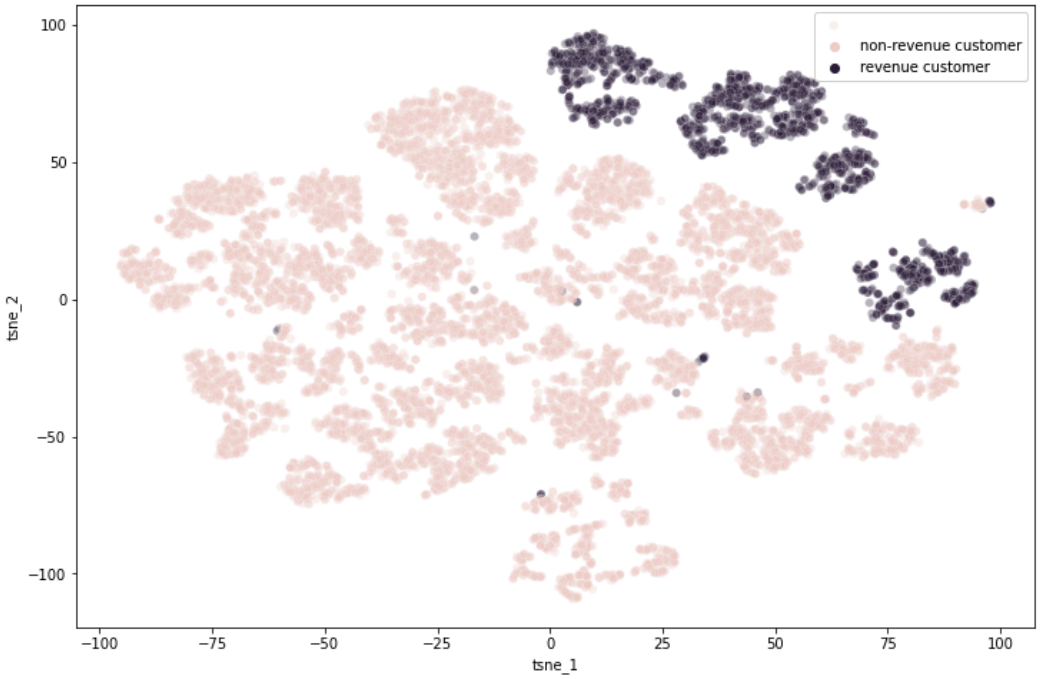
  Była to trasformacja stanowiąca układ odniesienia do reszty operacji. Nie spodziewaliśmy się większego efektu, jednak po zastosowaniu metody redukcji wielowymiarowości `t-SNE`, okazało się, że sesje klienckie zostały skutecznie rozdzielone na te, które przyniosły dochód i te które tego dochodu nie przyniosły (zmienną `Revenue`).



Rysunek 6: Wizualizacja `t-SNE` po wykorzystaniu skalowania standardowego.

- Transformacja logarytmiczna zmiennych o skośnych rozkładach

Wykonaliśmy transformację ze względu na to, że w wyniku eksploracji danych zauważyliśmy, że rozkłady zmiennych numerycznych są skośne (patrz Rysunek 5). Po zastosowaniu powyższego przekształcenia algorytm `t-SNE` poprawił precyzję w podziale na zmienną `Revenue`.



Rysunek 7: Wizualizacja `t-SNE` po wykorzystaniu transformacji logarytmicznej.

- One-hot-encoding

  Innym sposobem zakodowania informacji związanej ze zmienną kategoryczną, którego użyliśmy był `OHE`. Pomimo starań, kodowanie to wniosło (z uwagi na ogromną ilość dodatkowych wymiarów) jedynie szum do algorytmów klastrowania i nie poprawiło wyniku (badanego dalej przy użyciu `t-SNE`).

- "Mulitple hot encoding"

  Zaproponowaliśmy użycie bardziej wyrafinowanej, uwzględniającej dystanse (tak ważne dla algorytmów klastrowania) wersję `OHE`. Zamiast wypełniać kolumny odpowiadające danym poziomom cechy jedynkami, ustawiliśmy w nich wartości równe $\frac{1}{2m}$, gdzie m oznacza liczbę unikatowych wartości danej, kodowanej zmiennej.

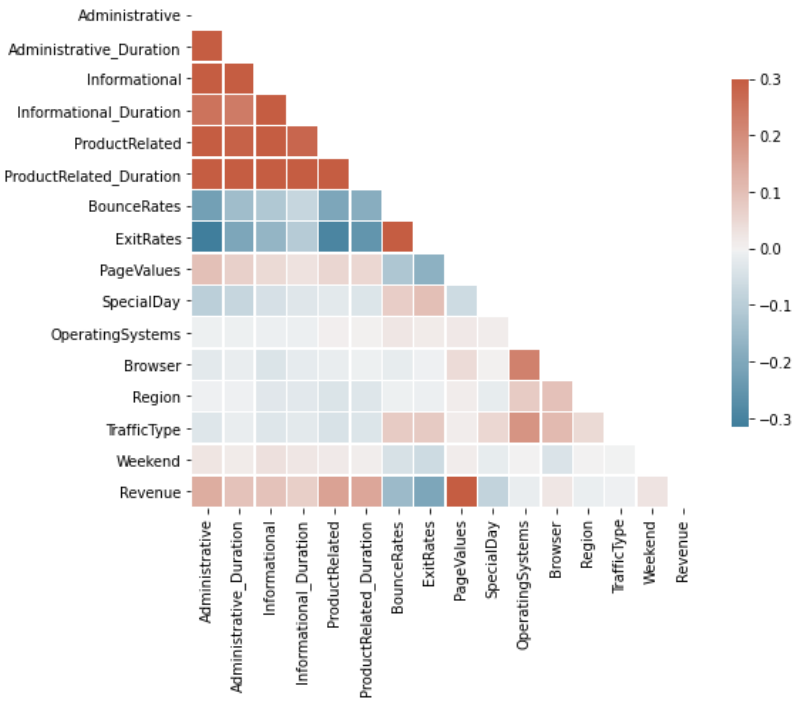
- Zgrupowanie wartości odstających

  Z uwagi na wysoką niejednostajność rozkładów zmiennych takich jak `Operating System` czy `Browser` dokonaliśmy zgrupowania mało licznych obserwacji tych zmiennych do bardziej ogólnego poziomu "inne".

- Usunięcie kolumn wysoce skorelowanych

  W algorytmach uczenia maszynowego, częstym problemem jest wysoka korelacja niektórych zmiennych. W wyborze tych zmiennych, kierowaliśmy się przedstawioną poniżej macierzą korelacji i usuwaliśmy kolumny `Administrative`, `Informational`,

`ProductRelated`, pozostawiając informacje tylko o czasie spędzonym na stronach danego typu.



Rysunek 8: Macierz korelacji zmiennych.

4 Modele

4.1 Zbiory danych

Na etapie konstrukcji, parametryzacji i uczenia algorytmów klastrujących pracowaliśmy na trzech różnie przygotowanych zbiorach danych opartych o transformacje zdefiniowane w sekcji 3:

- zbiór o zmiennych po transformacji logarytmicznej i standaryzacji (bazowy - `df base`),
- zbiór o zmiennych po transformacji logarytmicznej i standaryzacji z mniejszą liczbą kolumn (odrzucone kolumny o wyraźnej korelacji, patrz Rysunek 8) (`df lf`),
- zbiór o zmiennych po transformacji logarytmicznej i standaryzacji z miesącami zakodowanymi biegunowo i "multiple hot encodingiem" (`df cat`).

4.2 Zadania projektowe i sposób walidacji wyników

Ponadto uważny czytelnik mógł zauważyć, że problem, z którym się mierzyliśmy możemy interpertować dwojako:

- Klasteryzacja danych bez zmiennej `Revenue` - identyfikacja klientów ze względu na generowanie przychodu

  Wykorzystaliśmy różne algorytmy klasteryzacji w celu podziału klientów na dwie grupy. Następnie sprawdziliśmy, jaką dokładność, w porównaniu do rzeczywistych etykiet poszczególnych obserwacji osiągnęły wykorzystane modele.

  Do oceny naszych rezultatów wykorzystane zostały metryki `accuracy` oraz `f1 score`.

- Klasteryzacja "biznesowa" - segmentacja klientów (z uwzględnieniem zmiennej `Revenue` jako standardowej statystyki)

  Realizacja pierwotnego założenia o znalezieniu różnie zachowujących się grup klientów sklepów internetowych. Wykorzystaliśmy wszystkie wersje ramki danych (przekształcone na różne sposoby zgodnie z poprzednim podpunktem). Nie nakładaliśmy na algorytmy (modele) klastrujące żadnych dodatkowych założeń poza zakresem ilości klastrów (2-12).

  W tym przypadku do oceny klasteryzacji używaliśmy miar:

  - silhouette - wykorzystuje średnią odległość pomiędzy obserwacjami wewnątrz grupy (ozn. a) i średnią odległość obserwacji do najbliższej „obcej" grupy (ozn. b). Silhouette obliczany jest dla każdej obserwacji w następujący sposób: $\frac{a-b}{max(a,b)}$, a dla całego zbioru jest to średnia tych wyników. Najlepsza wartość miary to 1, a najgorsza -1. Wartości w pobliżu 0 oznaczają pokrywające się klastry.
  - Caliński-Harabasz score - obliczany jako stosunek sumy kwadratów odległości między klastrami do sumy kwadratów odległości obserwacji od środka ich klastra. Czym wyższa wartość, tym lepiej.
  - Davies-Bouldin score - definiowany intuicyjnie jako średnia miara podobieństwa każdego klastra z najbliższym mu klastrem, gdzie podobieństwo to stosunek odległości obiektów wewnątrz klastra do odległości pomiędzy klastrami. Zatem bardziej oddalone i odseparowane klastry mają lepszy rezultat - minimalna wartość to 0 i czym mniejsza, tym lepsza.

  Są to więc metryki odległości i kondensacji ("zbicia") wygenerowanych klastrów, ich dokładne definicje są dostępne m.in.w dokumentacji pakietu `Sklearn`.

4.3 Wybrane algorytmy

Algorytmy, które wykorzystaliśmy do klasteryzacji to algorytmy pozwalające na dobór liczby klastrów/komponentów przez użytkownika:

- `KMean` – metoda k-średnich;
- klasteryzacja hierarchiczna aglomeracyjna z różnymi metodami połączenia:
  - Warda (`ward`) – odległość między dwoma klastrami liczona jako suma kwadratów odchyleń punktów do centroidów,

  - kompletne (`complete`) – odległość między dwoma klastrami liczona jako maksymalna odległość między obserwacją w jednym klastrze a obserwacją w innym klastrze,
  - średnie (`average`) – odległość między dwoma klastrami liczona jako średnia odległość między obserwacją w jednym klastrze a obserwacją w innym klastrze,
  - pojedyncze (`single`) – odległość między dwoma klastrami liczona jako minimalna odległość między obserwacją w jednym klastrze a obserwacją w innym klastrze;

- `GMM` (`Gaussian Mixture Model`) – z różnymi typami kowariancji komponentów (określających szerokość rozkładów - komponentów):
  - pełny (`full`) – każdy komponent ma swoją własną macierz kowariancji,
  - związany (`tied`) – każdy komponent ma tę samą macierz kowariancji,
  - diagonalny (`diag`) – każdy komponent ma swoją diagonalną macierz kowariancji,
  - sferyczny (`spherical`) – każdy komponent ma swoją własną wariancję.

4.4 Klasteryzacja klasyfikacyjna

Dobierając odpowiednio parametry powyższych algorytmów (tj. ustawiając liczbę klastrów na dwa), przetestowaliśmy 27 różnych kombinacji typu (algorytm - transformowana ramka danych).

	df base	df lf	df cat
KMeans	0.418897	0.760827	0.417843
Agglomerative w/ ward linkage	0.786212	0.755718	0.784915
Agglomerative w/ complete linkage	0.842498	0.839822	0.501622
Agglomerative w/ average linkage	0.842903	0.157097	0.845012
Agglomerative w/ single linkage	0.845174	0.845174	0.845174
GMM w/ full covariance	0.234144	0.476318	0.333171
GMM w/ tied covariance	0.436983	0.780860	0.782806
GMM w/ diag covariance	0.463828	0.476318	0.485969
GMM w/ spherical covariance	0.422871	0.775750	0.786456

Tabela 1: Accuracy osiągane przez wybrane algorytmy klastrujące do identyfikacji, czy dana sesja przyniosła serwisowi zysk. Kolumny to kolejne ramki danych opisane w sekcji 4.1.

Analizując osiągnięte accuracy (wyniki zawarte w tabeli 1), można zauważyć, że w tym wypadku najlepiej poradził sobie algorytm klasteryzacji aglomeracyjnej z połączeniem pojedynczym. Przypomnijmy jednak, że w tym połączeniu odległość między dwoma klastrami jest minimalną odległością między obserwacją w jednym klastrze a obserwacją w innym klastrze.

Zauważmy też, że algorytm osiągnął ten sam wynik na każdym zestawie danych, zatem przy podziale kluczowe musiały być niezmienne cechy. Osiągnięty wynik to ok.

84.5% trafności. Jednak należy pamiętać, że zbiór nie jest zrównoważony (ok. 1900 klientów przynoszących dochód i 10400 nieprzynoszących - patrz Figura 1). Istotnie, wysoki wynik wynikał z przydzielania każdej (poza jedną) obserwacji bardziej popularnej etykiety (czyli brak dochodowości sesji).

Zatem w celu badania zbalansowania osiagniętej klasteryzacji, sprawdziliśmy wyniki w mierze F1 macro (średnia z miar F1 dla obu grup). Wyniki przedstawione są w tabeli 2.

	df base	df lf	df cat
KM	0.334671	0.434074	0.339267
Agglomerative w/ ward linkage	0.441275	0.431407	0.440861
Agglomerative w/ complete linkage	0.465798	0.466318	0.358813
Agglomerative w/ average linkage	0.465444	0.137748	0.457998
Agglomerative w/ single linkage	0.458046	0.458046	0.458046
GMM w/ full covariance	0.204634	0.334301	0.265043
GMM w/ tied covariance	0.363637	0.439564	0.440554
GMM w/ diag covariance	0.329124	0.334301	0.337826
GMM w/ spherical covariance	0.339874	0.437923	0.441353

Tabela 2: F1 macro score osiągany przez wybrane algorytmy klastrujące do identyfikacji, czy dana sesja przyniosła serwisowi zysk. Kolumny to kolejne ramki danych opisane w sekcji 4.1.

Okazało się, że w tej mierze najlepszym modelem również okazał się model aglomeracyjny z połączeniem pojedynczym. Zatem widzimy, że klasteryzacja nie jest dobrym pomysłem, jeśli chodzi o predykcje etykiety. W takim celu powinniśmy sięgać po znane algorytmy uczenia nadzorowanego. Natomiast przykład ten jest też ciekawy z perspektywy nauki – dobitnie pokazuje, by nie skupiać się w modelowaniu na optymalizowaniu pod jak najwyższe `accuracy` i by nie wierzyć naiwnie w predykcje modelu bez głębszego zbadania sytuacji.

4.5 Klasteryzacja właściwa

W ramach klasteryzacji właściwej, wykonaliśmy obliczenia i testy dla wymienionych wcześniej ramek danych i algorytmów przy ograniczeniu poszukiwanej liczby grup między 2 a 12. Celem tego eksperymentu było odnaleźć kombinację ramki danych oraz modelu i jego hiperparametrów, które minimalizowałyby błędy wszystkich wykorzystanych przez nas miar lub osiągający w nich najlepsze rezultaty. W ten sposób osiągnęliśmy ogromny zbiór wyników (dostępny w notatniku w repozytorium przedmiotu), z którego wywnioskowaliśmy, że nie ma jednoznacznego rozwiązania, tj. nie da się wskazać jednego algorytmu i zbioru, który najlepiej poradził sobie z klasteryzacją.

W miarach silhouette i Daviesa-Bouldina w większości przypadków najlepiej wypadała klasteryzacja aglomeracyjna z połączeniem pojedynczym, jednak w mierze Calińskiego-Harabasza osiągała ona bardzo słabe wyniki. Ma to związek z tym, że tworzone przez nią klastry są bardzo mało liczne (tak jak we wcześniejszym zadaniu), a więc nie nadawały się do naszego zadania.

Dla zbioru ze zmiennymi kategorycznymi w nowych kolumnach na tle innych algorytmów najlepiej wypadała klasteryzacja przy użyciu metody KMeans.

Eskperyment pokazał też, że w wielu przypadkach najlepsze wyniki miar mają klasteryzacje na niewielką liczbę grup (2-4). Zależało nam jednak na zidentyfikowaniu większej ilości segmentów rynku klientów.

4.5.1 Normalizacja wyników metryk

W celu znalezienia odpowiedniego modelu i lepszej (oraz łatwiejszej) oceny sytuacji wprowadziliśmy znormalizowany score dla każdej klasteryzacji. Zdefiniowaliśmy go jako:

$$NormScore = S + norm(CH) + 1 - norm(DB),$$

gdzie:

- S jest wartością silhouette score dla danej klasteryzacji,
- $norm(CH)$ jest znormalizowaną wartością indeksu Calińskiego-Harabasza,
- $norm(DB)$ jet znormalizowaną wartoścca indeksu Daviesa-Bouldina.

Znormalizowane wyniki były obliczane dla każdej ramki danych oddzielnie. Sama normalizacja również była wykonywana dla każdej ilości klastrów oddzielnie. Miało to miejsce przy użyciu normy L_2.

Wyniki dla poszczególnych ramek danych przedstawiają poniższe tabele.

	2	3	4	5	6	7	8	9	10	11	12
KM	1.121	1.250	1.270	1.240	1.386	1.328	1.325	1.297	1.329	1.283	1.296
Agl-ward	1.447	1.225	1.179	1.145	1.302	1.264	1.207	1.232	1.239	1.261	1.271
Agl-cmpl	1.378	1.103	1.077	1.178	1.214	1.175	1.149	1.044	1.036	0.956	0.956
Agl-avg	1.371	1.180	1.223	1.165	1.241	1.217	1.181	1.181	1.134	1.126	1.126
Agl-sngl	1.408	1.306	1.278	1.265	1.277	1.249	1.231	1.204	1.214	1.207	1.183
GMM-full	1.043	1.030	1.039	1.159	0.932	0.958	0.974	0.983	0.919	1.026	0.956
GMM-tied	1.464	1.299	1.337	1.220	1.167	1.134	1.108	1.156	1.150	1.130	1.190
GMM-diag	1.043	1.031	1.039	1.157	0.912	1.020	0.973	0.928	1.097	1.099	0.999
GMM-sphr	1.445	1.320	1.089	0.991	1.012	1.065	1.120	1.202	1.181	1.175	1.117

Tabela 3: Znormalizowany score (`NormScore`) uwzględniający wartości miar `silhouette`, `Caliński-Harabasz score` oraz `Davies-Bouldin score` dla ramki danych `df base`.

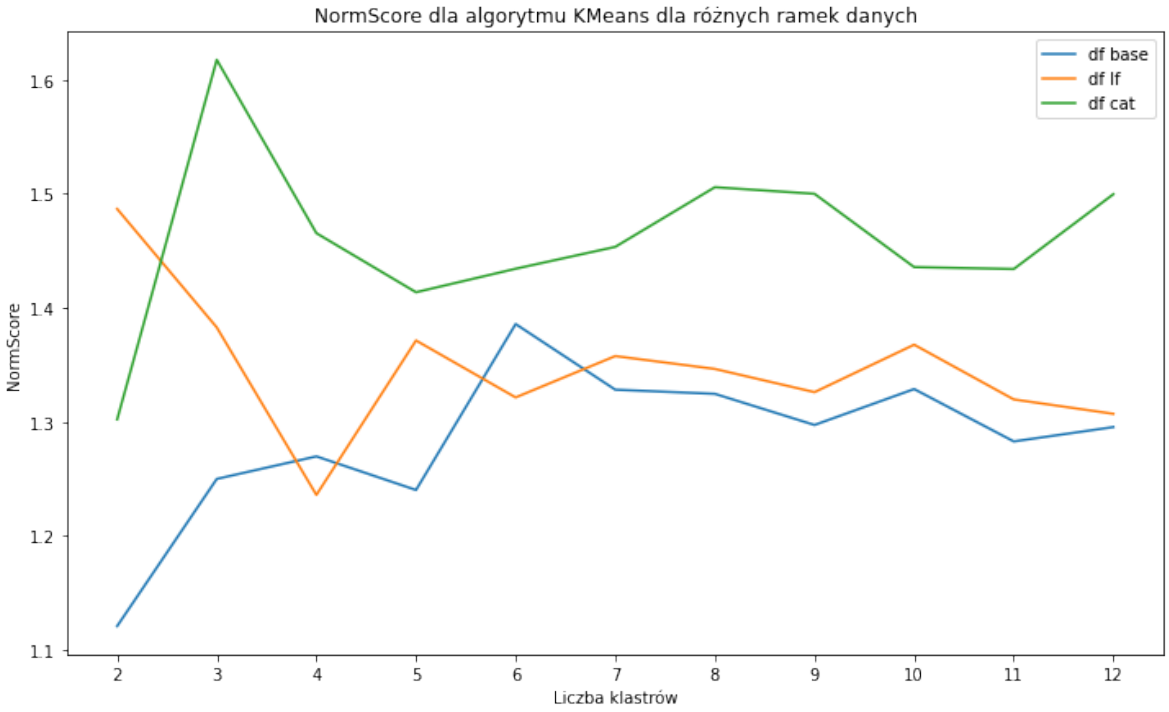
	2	3	4	5	6	7	8	9	10	11	12
KM	1.487	1.383	1.236	1.371	1.322	1.358	1.346	1.326	1.368	1.320	1.307
Agl-ward	1.484	1.342	1.306	1.293	1.248	1.259	1.225	1.249	1.300	1.211	1.204
Agl-cmpl	1.269	1.454	1.365	1.203	1.160	1.111	1.085	1.095	1.137	1.090	1.130
Agl-avg	1.448	1.441	1.266	1.375	1.287	1.259	1.252	1.244	1.255	1.222	1.154
Agl-sngl	1.341	1.364	1.368	1.339	1.313	1.300	1.314	1.284	1.260	1.240	1.234
GMM-full	0.901	0.930	1.003	0.869	0.881	0.838	0.929	0.812	0.738	0.954	0.965
GMM-tied	1.540	1.156	1.252	1.264	1.284	1.294	1.302	1.284	1.320	1.227	1.195
GMM-diag	0.889	0.930	1.003	0.893	0.911	0.881	0.906	0.930	0.717	0.867	0.931
GMM-sphr	1.531	1.037	0.916	1.110	1.186	1.197	1.227	1.196	1.302	1.228	1.207

Tabela 4: Znormalizowany score (`NormScore`) uwzględniający wartości miar `silhouette`, `Caliński-Harabasz score` oraz `Davies-Bouldin score` dla ramki danych `df lf`.

	2	3	4	5	6	7	8	9	10	11	12
KM	1.302	1.617	1.465	1.414	1.434	1.453	1.506	1.500	1.436	1.434	1.500
Agl-ward	1.645	1.557	1.368	1.333	1.360	1.382	1.391	1.391	1.329	1.351	1.414
Agl-cmpl	1.150	1.174	1.288	1.071	1.034	1.134	1.154	1.155	1.160	1.185	1.263
Agl-avg	1.425	1.480	1.274	1.238	1.184	1.288	1.294	1.273	1.210	1.219	1.343
Agl-sngl	1.425	1.340	1.051	1.002	0.880	0.865	0.895	0.894	0.858	0.867	0.867
GMM-full	1.151	0.593	0.803	1.042	0.964	0.782	0.576	0.752	0.751	0.618	0.538
GMM-tied	1.224	1.459	1.444	1.320	1.407	1.418	1.279	1.261	1.197	1.217	1.336
GMM-diag	0.938	0.823	0.835	1.089	1.019	0.940	0.841	0.729	0.952	0.900	0.812
GMM-sphr	1.302	1.314	1.449	1.310	1.421	1.450	1.456	1.442	1.382	1.417	1.484

Tabela 5: Znormalizowany score (`NormScore`) uwzględniający wartości miar `silhouette`, `Caliński-Harabasz score` oraz `Davies-Bouldin score` dla ramki danych `df cat`.

Po przeanalizowaniu powyższych danych stwierdziliśmy, że to najprostszy z algorytmów - `KMeans` radzi sobie najlepiej z naszym zbiorem, w szczególności dla liczby klastrów większej niż 4, a więc takiej, jaka nas najbardziej interesowała. Wartości znormalizowanego score'a dla tego algorytmu i różnych ramek danych oraz liczby klastrów posłużyły nam w celu wyboru ostatecznej liczby klastrów w poszczególnych przypadkach (patrz Rysunek 9).



Rysunek 9: Wartości znormalizowanego score'a dla algorytmu `KMeans` i różnych ramek danych oraz liczby klastrów.

4.5.2 Wybór ostatecznego modelu

Należy jednak pamiętać, że `KMeans` jest metodą niedeterministyczną i wynik zależy od początkowego wyboru punktów, w szczególności w przypadku zbioru o tak wielu wymiarach. Dlatego kolejnym krokiem, jaki podjęliśmy było znalezienie takich punktów

startowych dla środków klastrów, które umożliwią jak najlepszy ("najczystszy") podział na segmenty względem zmiennej `Revenue`.

Optymalnego rozwiązania szukaliśmy dla każdego zbioru danych. Dla ramki bazowej (`df base`) i z mniejszą ilością kolumn (`df lf`) zrobiliśmy to dla liczby klastrów równej 10 (lokalne maksima, największe wartości dla liczby klastrów większej od 6, patrz Rysunek 9). Natomiast dla ramki z innym kodowaniem zmiennych kategorycznych (`df cat`) wybraliśmy jako liczbę klastrów 8.

W celu znalezienia podziału dającego najczystsze segmenty skonstruowaliśmy własną metrykę, która wyraża się wzorem:

$$N = \frac{\sum_{i \in T(k)} R_i}{\sum_{i \in F(k)} R_i},$$

gdzie:

- $T(k)$ to klastry o frakcji klientów przynoszących dochód większej niż k,
- $F(k)$ to pozostałe klastry,
- R_i to frakcja klientów przynoszących dochód z i-tego klastra.

	size	revenue_ratio
0	1118	0.991950
1	817	0.026928
2	1471	0.000000
3	820	0.004878
4	784	0.062500
5	1211	0.069364
6	1125	0.039111
7	1969	0.021331
8	2490	0.012450
9	525	0.996190

(a) Wynik dla ramki danych `df base`

	size	revenue_ratio
0	870	0.025287
1	1422	0.028833
2	1161	0.000861
3	1298	0.996918
4	1315	0.045627
5	812	0.003695
6	422	0.971564
7	2778	0.004320
8	1896	0.000000
9	356	0.182584

(b) Wynik dla ramki danych `df lf`

	size	revenue_ratio
0	2894	0.061852
1	783	0.003831
2	982	0.113035
3	1817	0.025867
4	1218	0.320197
5	1471	0.687967
6	853	0.044549
7	2312	0.055363

(c) Wynik dla ramki danych `df cat`

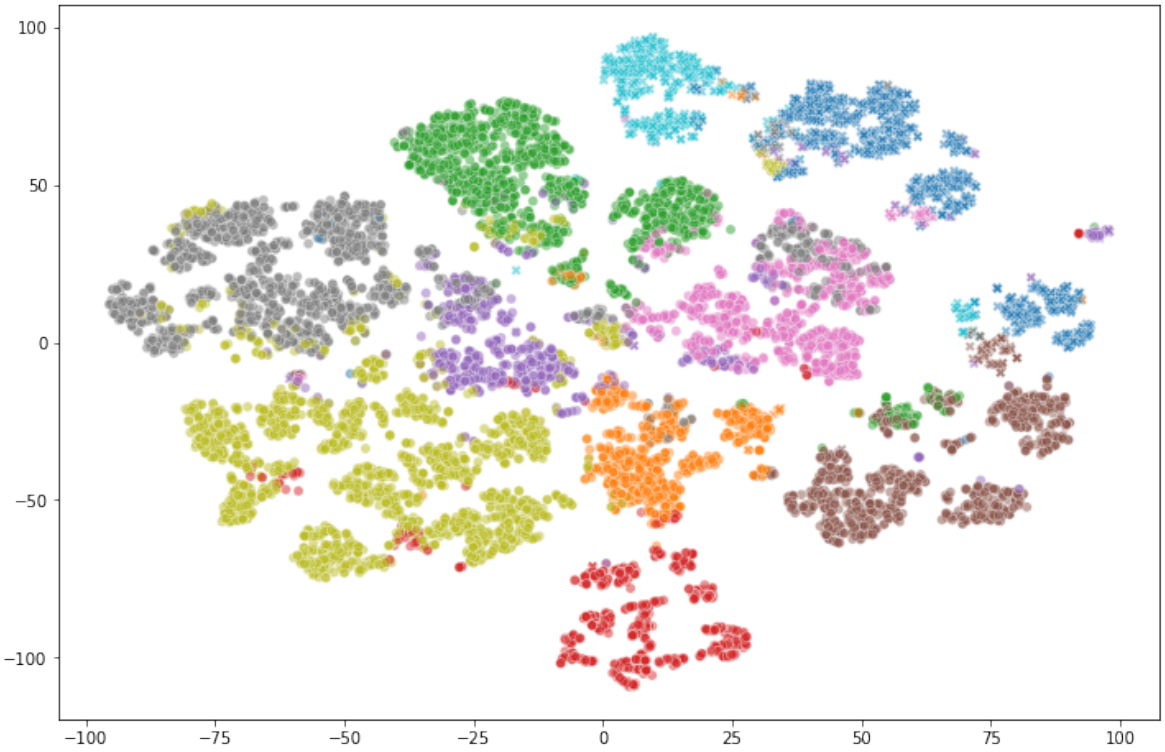
Tabela 6: Sposób podziału na klastry przez metodę KMeans dla liczby klastrów wybranej przy pomocy znormalizowanego score – Rysunek 9 i dla punktów startowych maksymalizujących metrykę N czystości segmentacji. Size oznacza rozmiar klastra, a revenue ratio frakcję klientów przynoszących dochód w klastrze.

Algorytm KMeans dał podobne rezultaty dla dwóch pierwszych ramek, jednak to w pierwszym przypadku udało się osiągnąć nieco lepszy wynik – klastry klientów przynoszących dochód są bardziej jednorodne i nie ma widocznego klastra mieszanego jak w drugiej wersji klasteryzacji (jeden z klastrów w 18% składa się z osób przynoszących dochód).

Dlatego zdecydowaliśmy się jako ostateczny model wybrać właśnie pierwszy algorytm, tj. KMeans dla 10 klastrów z `random_state = 127` działający na ramce danych `df base` (bez odrzuconych kolumn i innego kodowania zmiennych kategorycznych).

4.5.3 Wizualizacja wyniku ostatecznego modelu

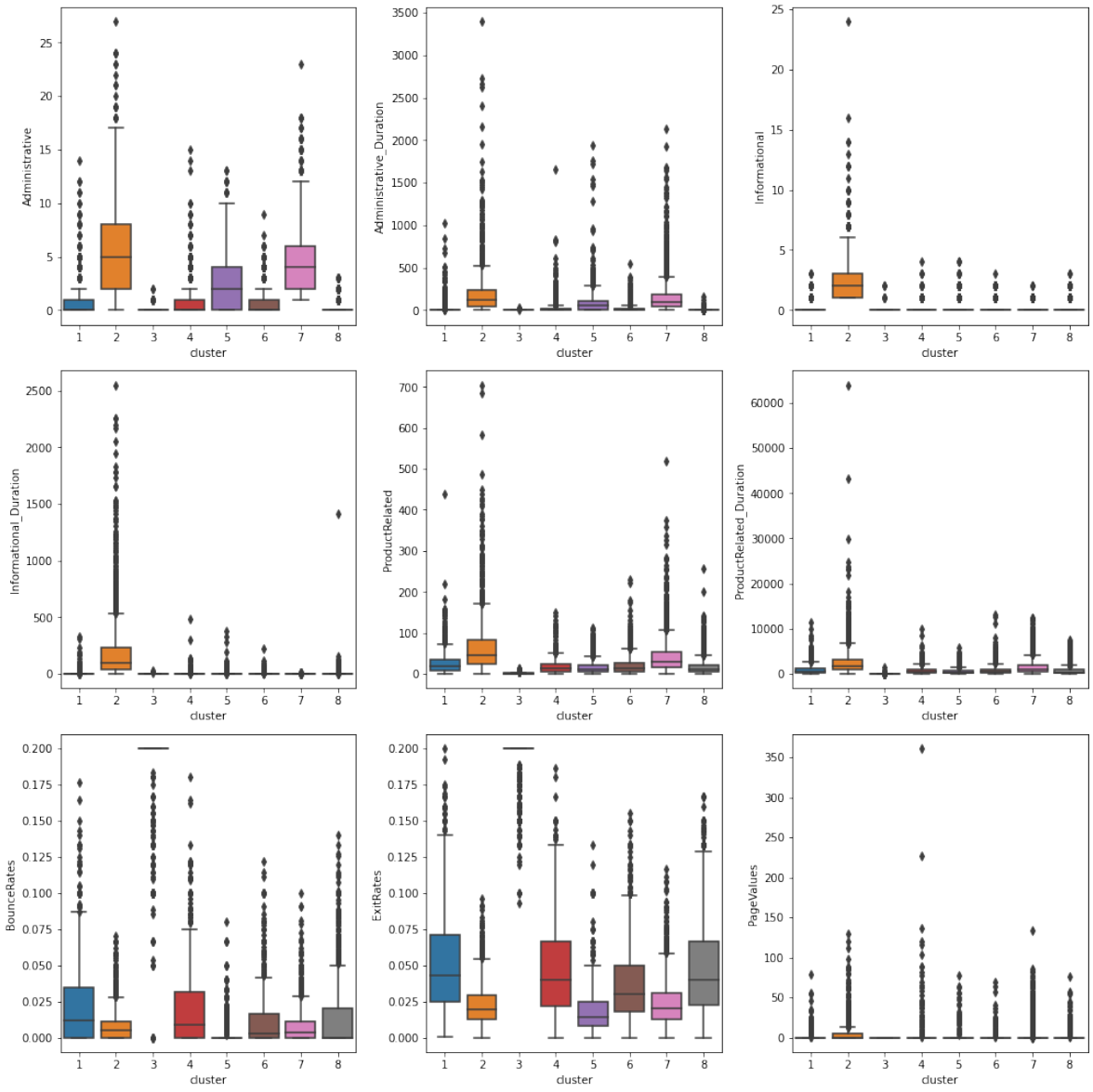
Ostateczną klasteryzację nanieśliśmy na wizualizację wygenerowaną przy pomocy metody `t-SNE`. Pomogło nam to się utwierdzić w przekonaniu, że jest ona sensownym wyborem.



Rysunek 10: Wizualizacja wyniku klasteryzacji finalnej za pomocą metody `t-SNE`. Kolorem zostały oznaczone różne klastry, kształt odpowiada wartości zmiennej `Revenue` (kółka – 0, krzyżyki – 1).

4.5.4 Przykładowa analiza różnic między grupami klientów nieprzynoszących dochodu

Prezentujemy również przykładową analizę stworzonych klastrów. Dokonujemy jej na ośmiu klastrach o małej frakcji klientów przynoszących dochód wśród zgrupowanych klientów. Dogłębna analiza i odszukanie złożonych różnic między klastrami byłaby materiałem na inne, ciekawe zadanie.



Rysunek 11: Histogramy zmiennych numerycznych dla klastrów o małej frakcji klientów przynoszących dochód.

Nawet z prostej analizy histogramów zmiennych numerycznych mamy informację, że klienci zakwalifikowani do klastra numer 2 spędzają dużo czasu na wszystkich stronach (długie sesje). Jednak wyraźną różnicę widać w czasie spędzonym na stronach o charakterze informacyjnym. W takim wypadku warto kierować do nich reklamy właśnie na takich stronach, w komunikacji odnosić się do faktów etc.

Widzimy też, że klienci z klastra numer 7 spędzają więcej czasu na stronach związanych z produktem - być może byliby oni bardziej skłonni do zakupu, gdyby zaoferować im odpowiednią, nawet niewielką promocję.

Klienci z klastrów o numerach 1, 4, 8 widocznie często odwiedzają strony o wyższych wartościach `BounceRate` i `ExitRate`, zatem może warto zaoferować im specjalne oferty właśnie na nich.

5 Podsumowanie

W ramach projektu numer 2 z WUM, udało nam się, zaczynając od surowych, rzeczywistych danych, zbudować szereg modeli i porównując, wybrać ten, który według przyjętych metryk spisuje się najlepiej. Korzystaliśmy z gotowych metryk, ale stworzyliśmy też swoje podejście do problemu, swoje metryki i własny sposób wyboru ostatecznego z nich. Realizowaliśmy zarówno klasteryzację klasyfikacyjną, jak i właściwą (nazwaną przez nas "biznesową").

Wnioskujemy jednak, że to nie sam rezultat czy liczbowa reprezentacja jego skuteczności jest w tym miejscu najważniejsza. Problem przede wszystkim uświadamia nas, że złożoność prawdziwych procesów społecznych i natury człowieka prowadzi do ogromnych trudności w "szufladkowaniu" metodami klastrowania. Ludzie i ich zachowania przenikają się, a proste podziały na "bardziej" i "mniej" chętnych do realizacji transakcji internetowej są niemożliwe (lub bardzo trudne). Jesteśmy przekonani, że doświadczenia wyniesione z powyższej pracy, przydadzą się nam na dalszej ścieżce kariery.