

Online Shoppers Purchasing Intention

Clustering + dataset preparation
method discussion

Jakub Kosterna
Zuzanna Mróz
Aleksander Podsiad

Online Shoppers Purchasing Intention – jak zapewne wszyscy pamiętamy z ostatnich dwóch kamieni milowych, umiarkowanie duży zbiór danych zawierający informacje o ruchu w sieci kilkunastu tysięcy użytkowników portalu aukcyjnego wraz z odpowiedzią na pytanie czy firma osiągnęła sukces w kwestii wciśnięcia klientowi produktu – czy nie. Dokonaliśmy już modelowania kilku algorytmów klasteryzacji i osiągnęliśmy pewne wyniki... ale w ciągu ostatnich dwóch tygodni postanowiliśmy nieco zmienić koncepcję. Myślę, że wielu z was może już domyślać się po nieprzypadkowym tytule dzisiejszej prezentacji czym jeszcze poskutkowała nasza praca ostatnich dni, ale zanim do niej przejdziemy – podsumujmy szybko nasz dotychczasowy dorobek.

MILESTONE NO. 1: DATA EXPLORATION

W tygodniu pierwszym skupiliśmy się w najprostszym jak to można rozumieć – zapoznaniu z ramką i jej eksportacją.

Cieźko nie przyznać, że los się do nas uśmiechnął – dostaliśmy całkiem konkretne ponad dwanaście tysięcy obserwacji, każda po osiemnaście cech. Na każdą z nich otrzymaliśmy informacje o zainteresowaniu klienta innymi stronami różnych typów w formie ilości wejść i czasu spędzonego na witrynach podzielonych na trzy grupy,

przygotowane miary od Google Analytics mówiące o współczynnikach wyjść i odrzuceń, dane czasowe takie jak miesiąc, tzw. omówiona miesiąc temu „specjalność” dnia czy binarna informacja o weekendzie, kategoryczne fakty o systemie operacyjnym, przeglądarce, regionie i typie ruchu po sieci, a także odpowiedź na pytanie czy użytkownik znalazł się na witrynie po raz pierwszy czy któryś z kolei i wreszcie – czy przeglądanie oferty poskutkowało zakupem. Warto także podkreślić, że stwierdzenie, że owe dane otrzymaliśmy na każdą z obserwacji nie było żadną koloryzacją – ku naszej uciech okazało się, że ramka nie zawiera żadnych braków danych!



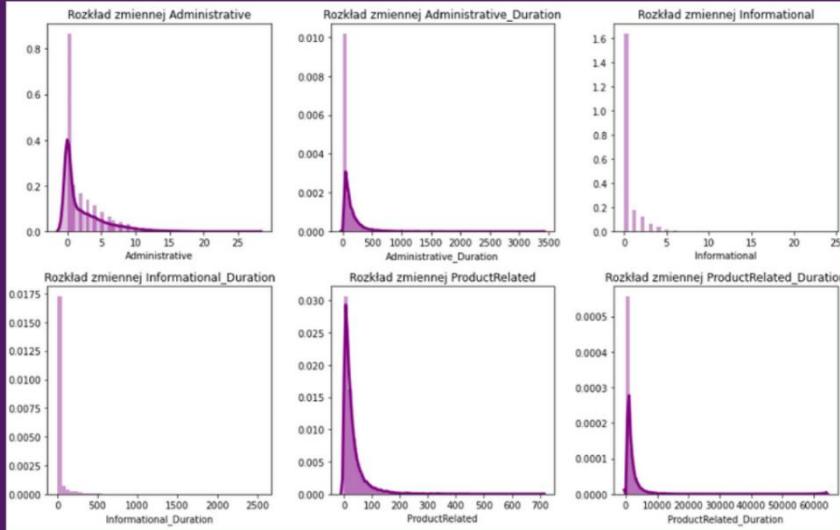
| df.describe() | | | | | | |
|---------------|----------------|-------------------------|---------------|------------------------|----------------|-------------------------|
| | Administrative | Administrative_Duration | Informational | Informational_Duration | ProductRelated | ProductRelated_Duration |
| count | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 |
| mean | 2.315166 | 80.818611 | 0.503569 | 34.472398 | 31.731468 | 1194.746220 |
| std | 3.321784 | 176.779107 | 1.270156 | 140.749294 | 44.475503 | 1913.669288 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 7.000000 | 184.137500 |
| 50% | 1.000000 | 7.500000 | 0.000000 | 0.000000 | 18.000000 | 598.936905 |
| 75% | 4.000000 | 93.256250 | 0.000000 | 0.000000 | 38.000000 | 1464.157213 |
| max | 27.000000 | 3398.750000 | 24.000000 | 2549.375000 | 705.000000 | 63973.522230 |

| BounceRates | ExitRates | PageValues | SpecialDay |
|--------------|--------------|--------------|--------------|
| 12330.000000 | 12330.000000 | 12330.000000 | 12330.000000 |
| 0.022191 | 0.043073 | 5.889258 | 0.061427 |
| 0.048488 | 0.048597 | 18.568437 | 0.198917 |
| 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 0.000000 | 0.014286 | 0.000000 | 0.000000 |
| 0.003112 | 0.025156 | 0.000000 | 0.000000 |
| 0.016813 | 0.050000 | 0.000000 | 0.000000 |
| 0.200000 | 0.200000 | 361.763742 | 1.000000 |

Żeby lepiej zapoznać się z naszymi ramkami, wpierw wygenerowaliśmy tabelę funkcji describe(). Wyczytaliśmy z niej, że użytkownicy najwięcej czasu spędzili na stronach związanych z produktem – zwykle kilkanaście minut, odwiedzając standardowo parę stron do kilkudziesięciu takich witryn. O wiele mniej w przypadku platform administracyjnych i informacyjnych- tam najczęściej blisko minuty, po parę również adresów. Naturalne outlierы również się znalazły – jeden użytkownik spędził bowiem prawie 20 godzin na stronach powiązanych z produktem! Postanowiliśmy jednak nie usuwać takich skrajnych wartości, gdyż... szaleńcy są wśród nas. Podsumowanie ukazało nam także, że współczynniki odrzuceń i wyjść oscylują wokół kilku procent, zaś typowy wskaźnik SpecialDay rzadko kiedy dosiąga jedynki – zwykle mamy styczność z dniami – że tak powiem – normalnymi, nie związanymi z żadnymi świętami czy eventami. Można też było zauważać tendencje do używania systemów operacyjnych czy przeglądarek o indeksach 1 i 2 – pytanie tylko, czy mogło na przykład chodzić o

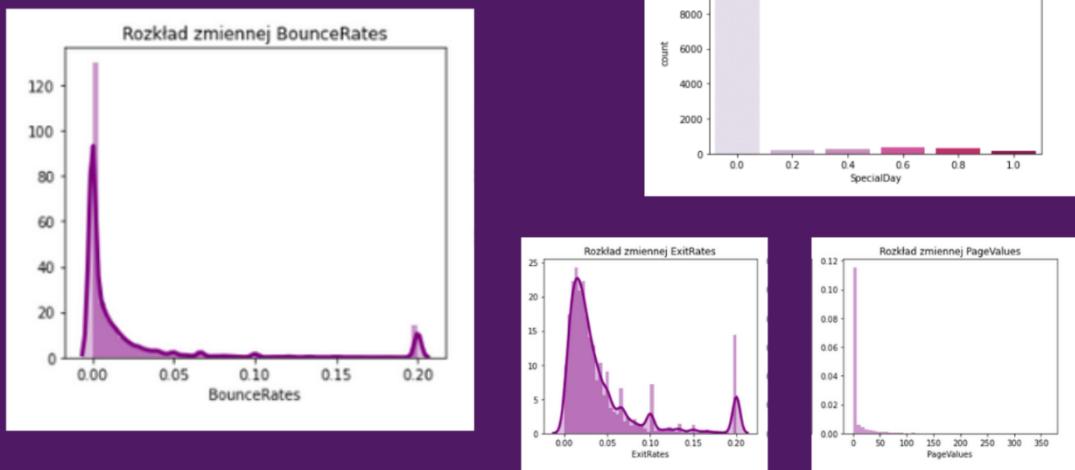
Google Chrome i Safari albo Windows i Mac? A może jakieś telefoniczne Android i Apple? Tego niestety nie wiemy, mamy tylko odróżnienie między kategoriami.

3/8

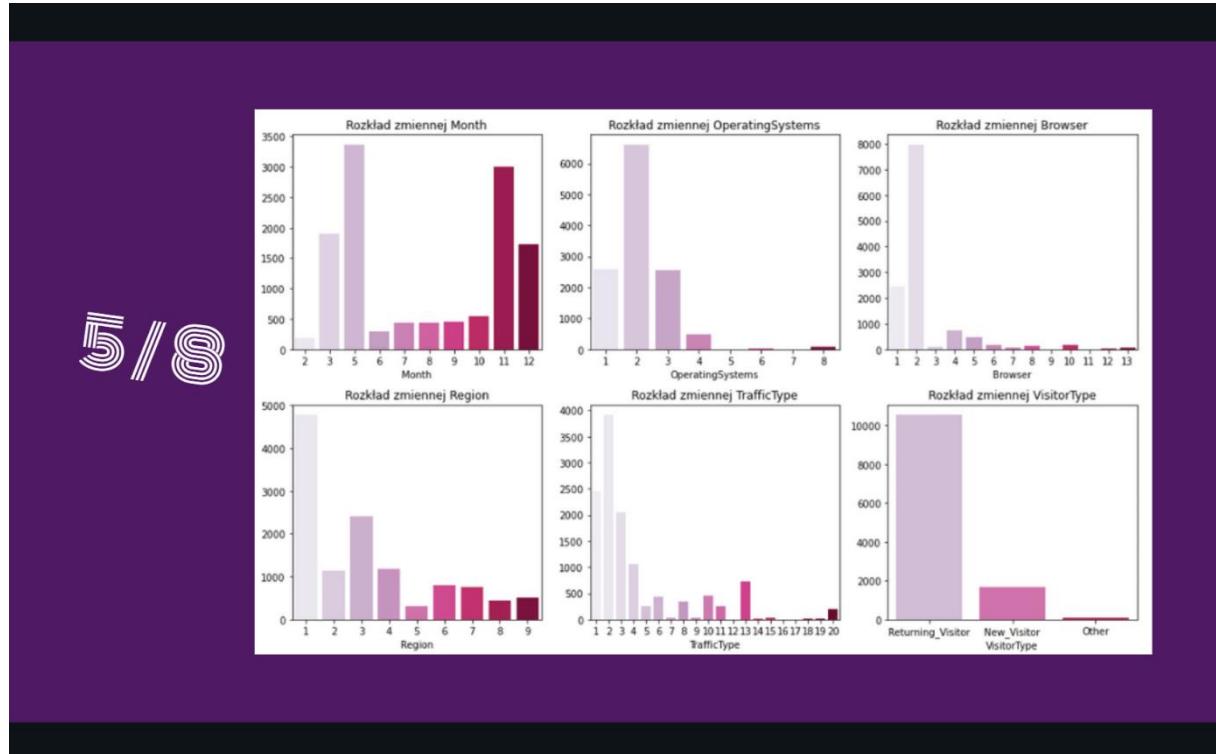


W celu jeszcze lepszego obycia z danymi i uzyskania więcej niż kilka miar położenia, utworzyliśmy także histogramy. W pierwszej kolejności przekonaliśmy się, jak bardzo ekstremalnie duże zainteresowanie innymi stronami odbiega od typowego stosunkowo nikłego,

4/8



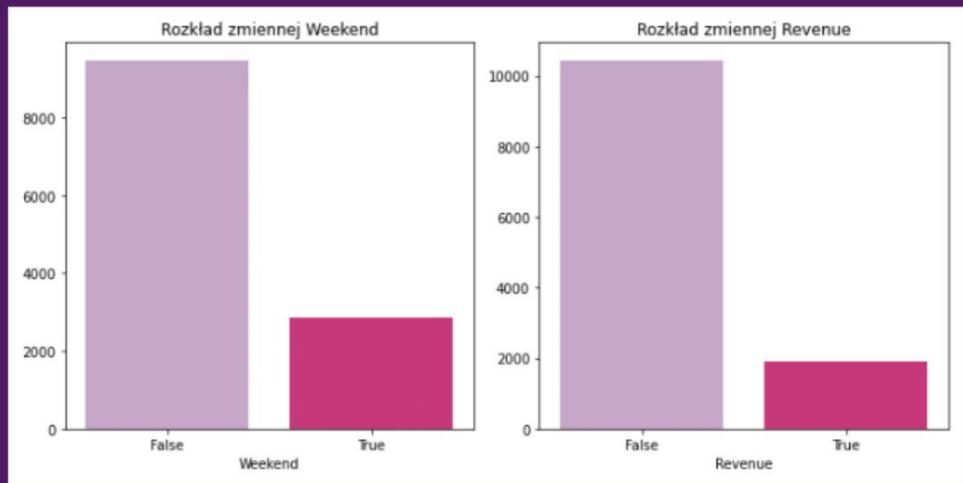
Przekonaliśmy się, że wskaźniki „BounceRates” i „ExitRates” nie przekraczają 20%, „PageValues” niemalże zawsze jest liczbą sięgającą kilkanaście i tylko w ekstremalnych przypadkach więcej niż 20, zaś w przypadku dni przynajmniej trochę specjalnych, największe skupisko jest – co ciekawe – wokół wartości 0.6.



Sprawdziliśmy także liczności zmiennych kategorycznych. Okazało się między innymi, że miesiące są dosyć niezrównoważone – nie ma na przykład żadnych danych ze stycznia i kwietnia, zaś w maju dla odmiany przyjmujemy maksimum. W praktyce taka sytuacja nie byłaby chyba możliwa, no bo jaka firma prowadzi sprzedaż przez cały okres z wyjątkiem dwóch wspomnianych miesięcy? Lampka się zapaliła, ale na etapie tak przygotowanych danych nie zrobiliśmy nic z ową Dys-reprezentacją.

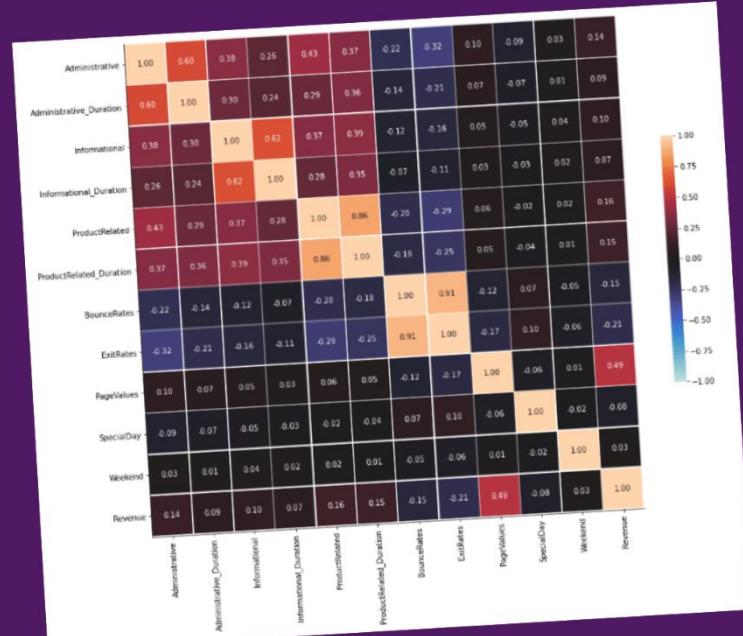
Zaobserwowaliśmy także istnienie kilku wiodących systemów operacyjnych, przeglądarek i regionów pod względem użytkowników, a także wyświetliśmy rozkład typów ruchu po sieci... który niestety za dużo nam nie powiedział, z wyjątkiem tego, że i tu pewne z tajemniczych 20 numerków są bardziej pospolite od innych. Co zaś istotne – okazało się, że absolutna większość klientów to ci, którzy stronę już wcześniej odwiedzali.

6/8



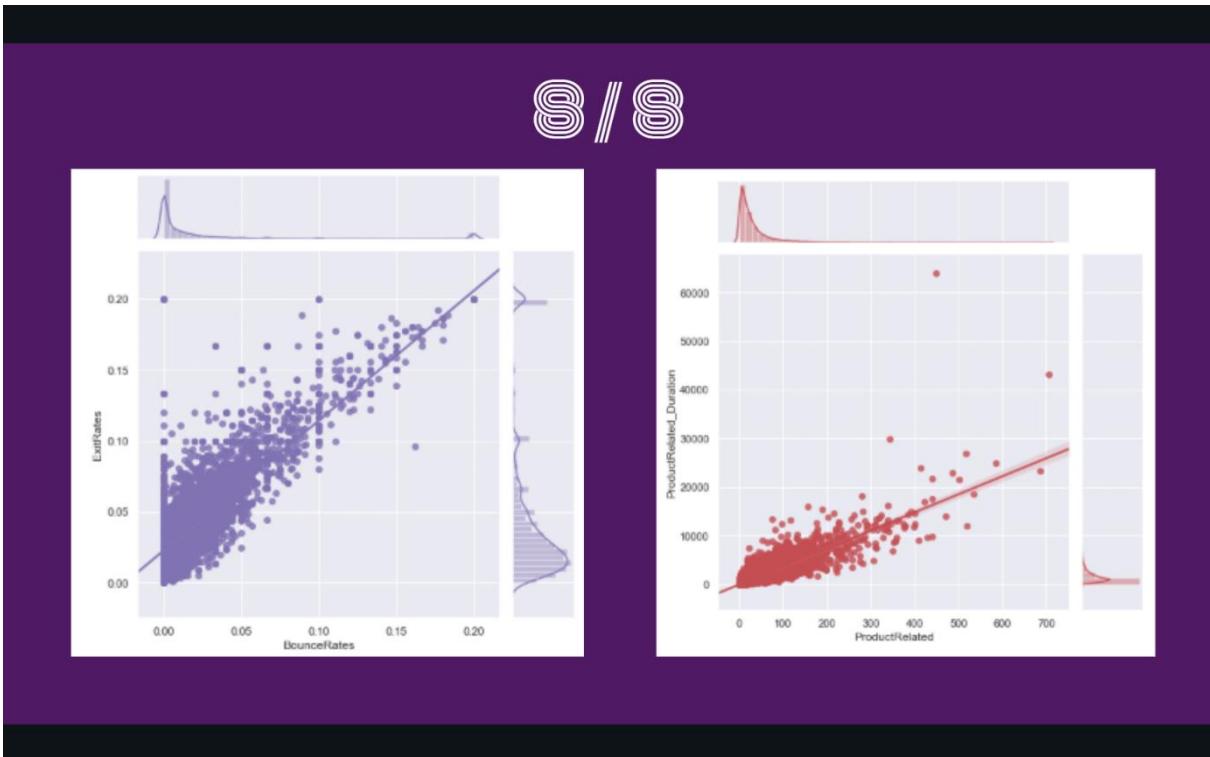
Wizualizacje ukazały także, że przeglądanie ofert zwykle dzieje się na tygodniu, zaś większość przypadków surfowania po sieci nie kończy się jednak zakupem... ale niech podobieństwo tych wykresów nie zmyli,

7/8



... gdyż macierz korelacji nie wykazała dużego podobieństwa między kolumnami „Weekend” i „Revenue”! Bez zaskoczeń odczytaliśmy z niej jednak, że można zaobserwować dużą korelację między liczbą stron odwiedzonych przez użytkowników a

czasem na nich spędzonych – co w innym przypadku byłoby podejrzane. Okazało się także, że klient wnikliwy jest wnikliwym wobec różnych kryteriów – ci bardziej zainteresowani stronami informacyjnymi wykazali także większą uwagę ku witrynom administracyjnym i powiązanym z produktem. Otrzymaliśmy także potwierdzenie definicji PageValues – rzeczywiście owy wynik jest związany z informacją czy z tej strony nastąpiła transakcja.



Na koniec etapu eksploracji porównaliśmy jeszcze parę kolumn między sobą – okazało się, że rzeczywiście miary BounceRates i ExitRates, a także ilości odwiedzonych stron wraz z czasami na nich spędzonych mają ze sobą wiele wspólnego.

MILESTONE NO. 2:

FRAME PREPARATION FOR UNSUPERVISED LEARNING AND PRELIMINAR MODELING

W drugim kamieniu milowym dokonaliśmy wstępnego modelowania, lecz w głównej mierze skupiliśmy się na encodingu oraz zmodyfikowaniu ramki pod uczenie maszynowe i dyskusji, jak tego mądrze dokonać. Mimo informacji podanej w postaci w miarę przejrzystej ramki, przeszliśmy dosyć burzliwe dyskusje w kwestii wyboru najlepszej metody przekonwertowania jej na formę odpowiednią do klasteryzacji. Jakich operacji dokonaliśmy?

```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
for i in range(len(months)):
    df[['Month']] = df[['Month']].replace(months[i], i + 1)
df[['Month']].sample(5)
```

| Month | |
|-------|----|
| 2342 | 5 |
| 5780 | 7 |
| 7060 | 10 |
| 5368 | 5 |
| 2024 | 3 |



Już w pierwszej kolejności w wątpliwość wprawiła nas kwestia miesięcy – mogłoby się wydawać, że najbardziej naturalnym podejściem będzie przypisanie im liczb od 1 do 12... ale czy nie będzie to wręcz zakazane, biorąc pod uwagę chociażby sztuczne oddalenie grudnia i stycznia, które jednak naturalnie następują obok siebie? Tak czy inaczej, niedoedokowani w kwestii istnienia lepszej alternatywy, wpierw tak właśnie postąpiliśmy.



```
df['AdministrativeResMan']=df['Administrative']
df['InformationalResMan']=df['Informational']
df['ProductRelatedResMan']=df['ProductRelated']

df.loc[(df['Administrative']>=1) & (df['Administrative']<=4), 'AdministrativeResMan'] = 1
df.loc[(df['Administrative']>=5) & (df['Administrative']<=9), 'AdministrativeResMan'] = 2
df.loc[(df['Administrative']>=10) & (df['Administrative']<=14), 'AdministrativeResMan'] = 3
df.loc[(df['Administrative']>=15) & (df['Administrative']<=18), 'AdministrativeResMan'] = 4
df.loc[(df['Administrative']>=19), 'AdministrativeResMan'] = 5

df.loc[(df['VisitorType']=='New_Visitor'), 'VisitorType0'] = 1
df.loc[(df['VisitorType']=='New_Visitor'), 'VisitorType1'] = 0
df.loc[(df['VisitorType']=='Returning_Visitor'), 'VisitorType0'] = 0
df.loc[(df['VisitorType']=='Returning_Visitor'), 'VisitorType1'] = 1
df.loc[(df['VisitorType']=='Other'), 'VisitorType0'] = 0
df.loc[(df['VisitorType']=='Other'), 'VisitorType1'] = 1

df = df.drop('VisitorType', axis = 1)
```

Operacją niestandardową, a radzącą sobie świetnie z outlierami i rozlewaniem się informacji o podobnym znaczeniu był pomysł obmyślenia własnej miary na kolumny dotyczące ilości wejść na strony poszczególnych kategorii – zamiast zostawiać luźne numery, postanowiliśmy w zależności od liczliwości tych trzech feautures zastąpić oryginalne wartości dyskretne kolumnami zawierającymi nasz autorski wskaźnik „Research maniac” – w zależności od intensywności poruszania się po witrynach, ten przypisywał użytkownikom liczbę naturalną z przedziału $<0; 5>$, gdzie 0 oznaczało absolutnie zerowe zainteresowanie, zaś 5 – maniakalne. Niestandardową operację wykonaliśmy także dla kolumny *VisitorType* – tu zastosowaliśmy encoding binarny jak najbardziej odróżniając od siebie internautów nowych i powracających, zaś tych o tajemniczej etykiecie „Others” – stawiając po środku między dwoma standardowymi. Kolumny binarne typu „True / False” naturalnie zaś bezkonfliktowo przerobiliśmy na zera i jedynki.

```

from scipy import stats
for i in range(len(df[['Administrative_Duration']])):
    df['AdministrativeDurPerc'][i] = stats.percentileofscore(df[['Administrative_Duration']], df[['Administrative_Duration']][i])

df[['AdministrativeDurPerc']].sample(10)

   AdministrativeDurPerc
1795           23.941606
5601           23.941606
6914           91.597729
5119           23.941606
9831           68.276561
7078           23.941606
6902           73.021087
11955          23.941606
2886           56.930251
8428           54.716139

df.loc[(df['Informational_Duration'] > 0) & (df['Informational_Duration'] <= 180), 'Informational_Duration'] = 1
df.loc[(df['Informational_Duration'] > 180), 'Informational_Duration'] = 2

df[['Informational_Duration']].sample(10)

   Informational_Duration
6680             0.0
10862            1.0
2499             0.0
2297             0.0
2276             0.0

df = df.drop('Administrative_Duration', axis = 1)

```



Tak jak dla ilości stron odwiedzanych można było jeszcze utworzyć mądry współczynnik maniakalności, dla czasów na owych witrynach rozkłady okazały się być jeszcze bardziej zastanawiające. Dla kolumn mówiących o liczbie sekund na stronach administracyjnych i związanych z produktem biorąc pod uwagę rozkład nieco przypominający jednostajny ale z pewnymi udziwnieniami, postanowiliśmy zastąpić wartości – percentylami. Totalnie innej operacji dokonaliśmy zaś dla Informational_Duration – tu zauważliśmy, że absolutna większość nie spędza na stronach o danej tematyce niemalże żadnego czasu. Spośród pozostałej mniejszości, przeważająca część jest tam zaledwie kilkanaście sekund do paru minut – zapewne z przypadku. Tylko pozostałe parę procent siedzi na nich więcej niż 3 minuty i tą grupę zdecydowanie należało by wyróżnić. Biorąc pod uwagę taki naturalny podział, tym pierwszym przypisaliśmy wartość 0, tym drugim – 1, zaś trzecim – 2.



A screenshot of a Jupyter Notebook cell. It contains a table of values, some Python code for standardization, and another table showing the result of applying standardization to the 'PageValues' column.

| | |
|------|------------|
| 0.75 | 0.000000 |
| 0.80 | 3.060078 |
| 0.85 | 9.319445 |
| 0.90 | 18.855502 |
| 0.95 | 38.160528 |
| 1.00 | 361.763742 |

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
df[['PageValues']] = scaler.fit_transform(df[['PageValues']])
```

| | PageValues |
|------|------------|
| 9404 | -0.317178 |
| 6232 | -0.317178 |
| 2033 | -0.317178 |

Miary BounceRates, ExitRates i SpecialDay postanowiliśmy zostawić bez zmian – już dane zostały przygotowane i obrobione w pewien zapewne mądry matematyczny sposób, więc w ich przypadku po prostu zaufaliśmy twórcom i temu, że są dobrze przekminione. Inna sprawa PageValues – tutaj zauważymy nieco większy chaos jak i niektóre wartości tak skrajne, że mogłyby zaburzyć istotę data frame, więc dokonaliśmy typowej standaryzacji rozkładu normalnego.




A screenshot of a Jupyter Notebook cell. It shows a transformation from a simple color-coded list to a one-hot encoding matrix, followed by a table of categorical data and its one-hot encoded version.

| # | Color |
|---|-------|
| 0 | Red |
| 1 | Green |
| 2 | Blue |
| 3 | Red |
| 4 | Blue |

→

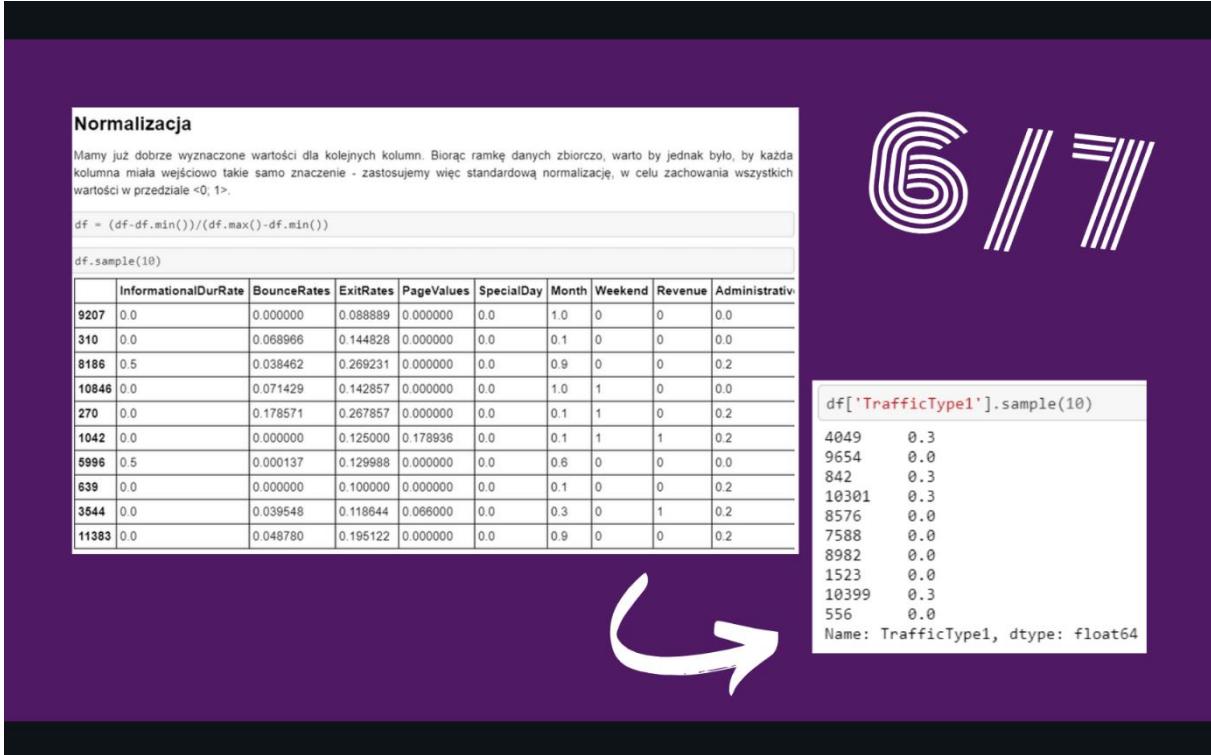
| # | Red | Green | Blue |
|---|-----|-------|------|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 |

| | |
|------------------|-------|
| OperatingSystems | 8 |
| Browser | 13 |
| Region | 9 |
| TrafficType | 20 |
| dtype: | int64 |

dfOperOnehot.sample(10)

| | OperatingSystems0 | OperatingSystems1 | OperatingSystems2 | OperatingSystems3 | OperatingSystems4 |
|-------|-------------------|-------------------|-------------------|-------------------|-------------------|
| 6237 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 8404 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 11779 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 2274 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 7508 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 9609 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3310 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 10338 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 1357 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 9642 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Oprócz kolumn o wartościach numerycznych lub binarnych, na koniec zostały nam jeszcze cztery kategoryczne, o liczbach unikalnych wartości jak na załączonym obrazku. I na pierwszy rzut oka można pomyśleć – one-hot encoding zawsze spoko. Ale czy dla klasteryzacji, w której takie kolumny binarne po owej operacji będą zajmować absolutną większość aby na pewno?



Ze względu na niezdrową mnogość utworzonych feautures postanowiliśmy więc postawić na dotychczas niespotkaną przez nas operację – wpierw dla wszystkich przetworzonych kolumn dokonaliśmy standardowej normalizacji od 0 do 1. lecz 50 z 65 kolumn powstałycych przez one-hot encoding potraktowaliśmy inaczej – tam zamiast 1, postawiliśmy na stałą 0.3. Czemu akurat tak? Czysta intuicja w celu oszacowania optymalnej liczby, która skutecznie zmniejszy sztucznie zwiększoną ważność kolumn odpowiadających te cztery informacje w formie kategorycznej. No bo dlaczego jakiś tam niewiadomy typ podróży po sieci miałby w praktyce obejmować prawie 1/3 wartości w gotowym data secie? Co jednak ważne, byliśmy świadomi, że owa stała jak jest wzięta znikąd, prawdopodobnie nie jest najlepszym rozwiązaniem, dlatego po dyskusji drugiego kamienia milowego, w ostatnich dniach postawiliśmy także na parę alternatyw – ale o nich za chwilę.



W kwestii wstępniego modelowania, postanowiliśmy postawić na trzy modele: poznane na laboratoriach KMeans, DBSCAN i GMM. Biorąc pod uwagę współczynnik sillhouette wnioski były proste – dla naszego zbioru danych KMeans poradził sobie całkiem-całkiem, DBSCAN totalnie zawalił sprawę niezależnie od liczby klastrów, a GMM jakoś zdał egzamin, faworyzując grupowanie na pięć – ale uzyskując jednak wyraźnie gorsze wyniki od KMeans. W tym wypadku dyskusja nie była zażarta i biorąc pod uwagę ten konkretny zbiór danych – stwierdziliśmy, że KMeans zrobił najlepszą robotę...

MILESTONE NO. 3:

**DATASET PREPARATION
METHOD DISCUSSION,
CONFIRMING THE BEST
N FOR KMEANS AND
REDUCING DIMENSIONS**

...i prawdę mówiąc tu nasza analiza różnych algorytmów klasteryzacji się zakończyła. Pierwotnie mieliśmy w planach lepiej przyjrzeć się trzem owym wspomnianym metodom, a może i przetestować parę innych, lecz koniec końców doznaliśmy natchnienia na nieco inny cel kamienia milowego trzeciego. Udało nam się dokonać wstępnego modelowania, które wyszło w miarę satysfakcyjnego, ale jakby nie patrzeć – zbiór generowaliśmy raczej na instynkt i nie uzyskaliśmy informacji, które niestandardowe operacje pozytywnie wpłynęły na klasteryzację, a które wręcz zaskoczyły. W tym celu zmieniliśmy koncepcję i przez ostatnie dwa tygodnie postawiliśmy na analizę działania KMeans na kilku różnych wersjach modyfikacji oryginalnego *online shoppers purchasing intention*. Stąd też ostateczny tytuł naszego projektu: „Clustering + dataset preparation method discussion”.

ONLINE PURCHASING DATA... DATASETS

df1

Original dataset



df2

Frame taken from milestone 2

df3

Original dataset

- PageValues standarization
- conversions into percentiles
- one-hot encoding modification
 - + circular coordinates for Month

df4

Frame taken from milestone 2

- + circular coordinates for Month

df5

Frame taken from milestone 2

- + circular coordinates for Month
- + individual values for one-hot encoding depending on the size of the category

W pierwszej kolejności postanowiliśmy zbadać to, co już wcześniej nas zastanawiało, a także posłuchać paru sugestii ze spotkania sprzed dwóch tygodni. Tak oto po wczytaniu dwóch pierwszych ramek danych – df1, czyli tej oryginalnej nieobrobionej i df2, która była końcowym rezultatem kamienia milowego drugiego – postawiliśmy jeszcze na dodatkowe 3. W df3 zdecydowaliśmy się na klasykę – zrezygnowaliśmy z operacji standaryzacji, przydzielania wartościom ich perentyli i niestandardowego enkodowania, zaś dokonaliśmy jedynie normalizacji, typowego one-hot encoding i co zostało nam zasugerowane po wniesionym do dyskusji dilemacie z problemem miesięcy – zamieniliśmy wartości <1; 12> na współrzędne kołowe tworząc dwie nowe kolumny – „MonthSin” i „MonthCos”. Idąc dalej df4 to zaś swego rodzaju zagubione ognisko między df2 a df3 – jest to ramka utworzona na wcześniejszym etapie, uzupełniona o mądrzejsze przechowywanie informacji o miesiącach. Dla df5 postanowiliśmy za to postawić na coś jeszcze bardziej innowacyjnego niż stała 0.3.

0 - not is, 1 - is



0 - not is,

[2 / <number of unique values in category column>] - is

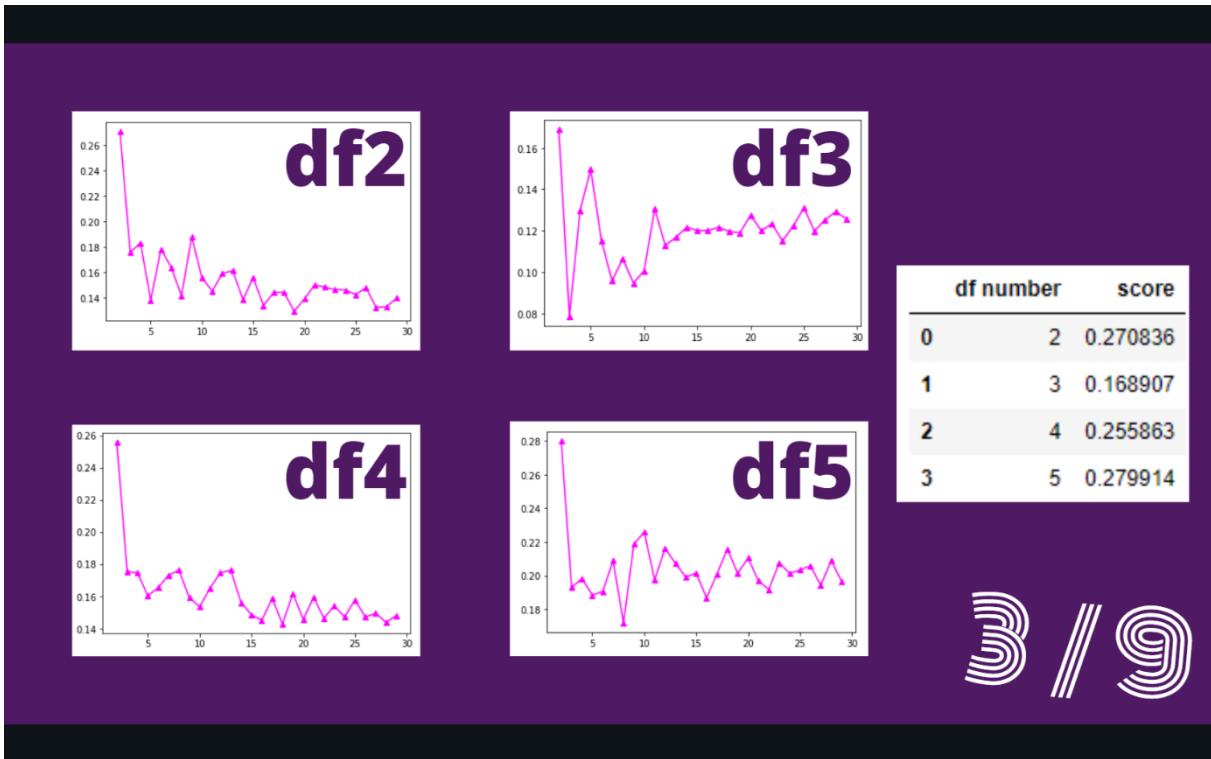
| | |
|------------------|----|
| OperatingSystems | 8 |
| Browser | 13 |
| Region | 9 |
| TrafficType | 20 |
| dtype: int64 | |

| df5['Region0'].sample(10) | |
|-------------------------------|----------|
| 12300 | 0.000000 |
| 2866 | 0.000000 |
| 12319 | 0.222222 |
| 4013 | 0.222222 |
| 7887 | 0.222222 |
| 439 | 0.000000 |
| 9994 | 0.222222 |
| 11111 | 0.222222 |
| 11357 | 0.000000 |
| 8319 | 0.000000 |
| Name: Region0, dtype: float64 | |

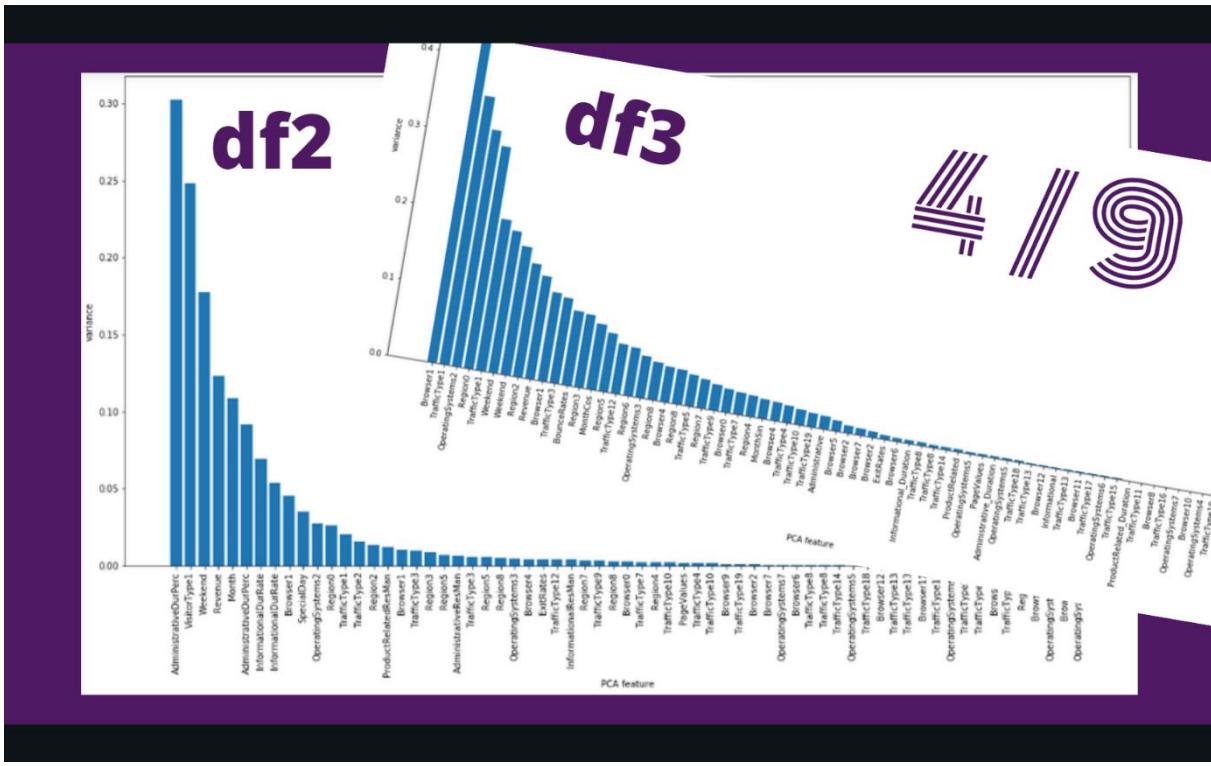


2 / 9

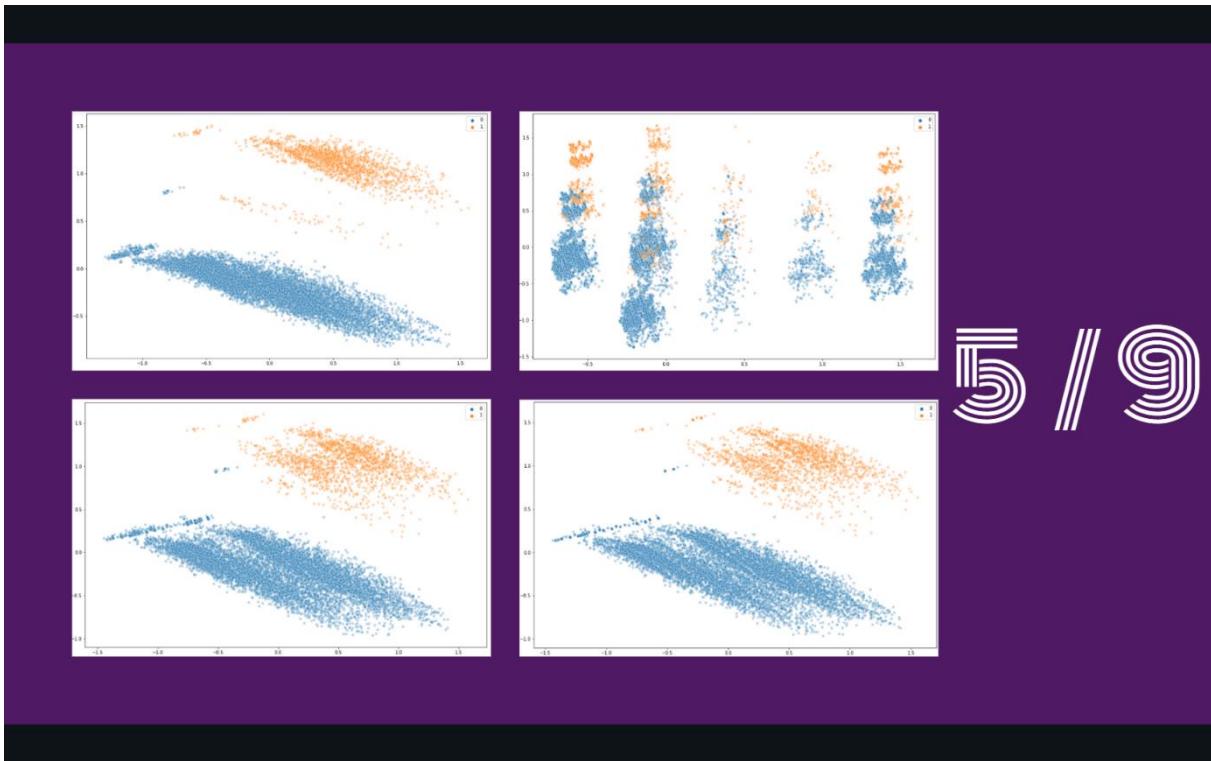
Problemem robienia one-hot encodingu niezależnie z jaką stałą jest to, że kolumny mające pierwotnie dużą ilość unikalnych wartości, po operacji są odpowiedzialne za tyle kolumn, ile owych unikatów znajdziemy. Tutaj na przykład dla df2 otrzymaliśmy 65 kolumn, z czego aż 50 dotyczyło tych binarnych zrobionych przez one-hot, a 20 z nich – samego tzw. „TrafficType”, który w praktyce na chłopski rozum powinien mieć znakomity wpływ na podział użytkowników czy nawet jakąkolwiek klasyfikację! Żeby uniknąć sztucznego napędzania ważności kolumn mających dużo unique values, postanowiliśmy postawić na lepszą alternatywę: każdej wartości powstałej z one-hot encodingu zamiast 0.3 dla występowania danej kategorii, przyznaliśmy nie tą randomową stałą niezależnie od zmiennej – lecz wartość 2 podzieloną przez liczbę unikalnych wartości danej kolumny. Dzięki tej mądrzej idei automatycznie features posiadające mniej wymiarów, wykazały większe porównanie między sobą i dały efekty bardziej zbliżony do typowego one-hot encodingu. W praktyce kolumny takie jak Browser0, Browser1, Browser2, itd. skończyły z wartościami 0 dla nieobecności danej kategorii i ok. 0.16 dla jej występowania – gdyż w oryginalnych danych mogliśmy zaobserwować 13 różnych unikatów. Za to taki TrafficType, którego jest 20 – otrzymał liczbę 0.1 dla obecności. Warto zauważyć, że idea ta jest o tyle mądra, że rzeczywiście zmniejsza sztuczną ważność cech, które pierwotni dotyczyły tylko jednej kolumny mając wiele różnych wartości, a koniec końców zajmują dużą część wszystkich słupów przy znacząco mniejszym wpływie na klasteryzację.



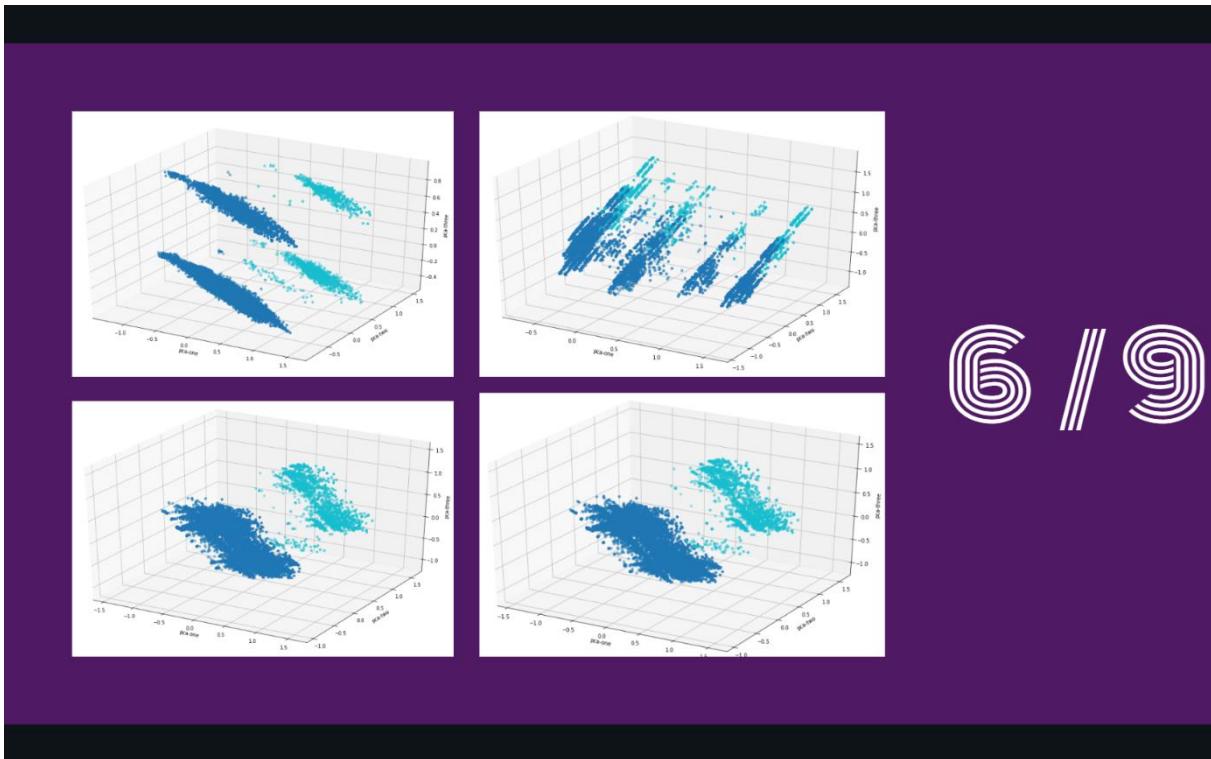
Następnie upewniliśmy się czy dwa to aby na pewno najlepsza liczba klastrów dla algorytmu KMeans na tych zbiorach – na szczęście okazało się, że porównując silhouette score, dwójka przodowała w każdym wypadku. Przy okazji dostaliśmy odpowiedź na pytanie, które zbiory według owej metody dostały najlepsze pogrupowanie. Okazało się, że operacje standaryzacji, przekształtowanie wartości numerycznych na percentile paru kolumn i zmodyfikowanego one-hot-encodingu zdecydowanie zrobiły swoje – ramka rezygnująca z tych operacji została daleko w tyle! Co ciekawe, porównując wykresy df2, df4 i df5 mimo podobnych wyników dla n = 2, można zaobserwować widoczne różnice – co jeszcze bardziej interesujące, modyfikacja miesięcy ze sposobu najbardziej instynktownego na współrzędne kolumnowe, w naszym wypadku widocznie zaskoczyła. Opcja spersonalizowanych, że tak to ujmę, wartości dla kolumn z one-hot-encoding zgodnie z przewidywaniami przyniosła zaś zdecydowanie zadowalający skutek!



Zanim jeszcze przyjrzaliśmy się wizualizacjom redukcji wymiarów, postanowiliśmy zainteresować się najbardziej znaczącym kolumnom dla podziału PCA. W aż 3 na 4 przygotowanych przez nas ramkach prym wiodła... ilość czasu spędzona na stronach administracyjnych! Nie była to jednak główna zmienna, bo do wyznaczenia wspólnych cech i różnic między użytkownikami duży wpływ miał też typ użytkownika – powracający lub nie; a także weekend nie-weekend, miesiąc czy informacja o tym czy przeglądanie oferty zakończyło się zakupem. Tutaj już wprost widać, że podział dla df3 okazał się niefortunny – największy wpływ okazały się mieć przeglądarki czy systemy operacyjne, które w praktyce nie powinny mieć dużo do czynienia z klasteryzacją. Jakby się lepiej przyjrzeć, okazuje się także, że wpływ czterech kolumn kategorycznych wydaje się być i tu bardziej naturalny dla df5 – zachęcamy do przyjrzenia się wszystkim wykresom w naszym raporcie kamienia milowego trzeciego.

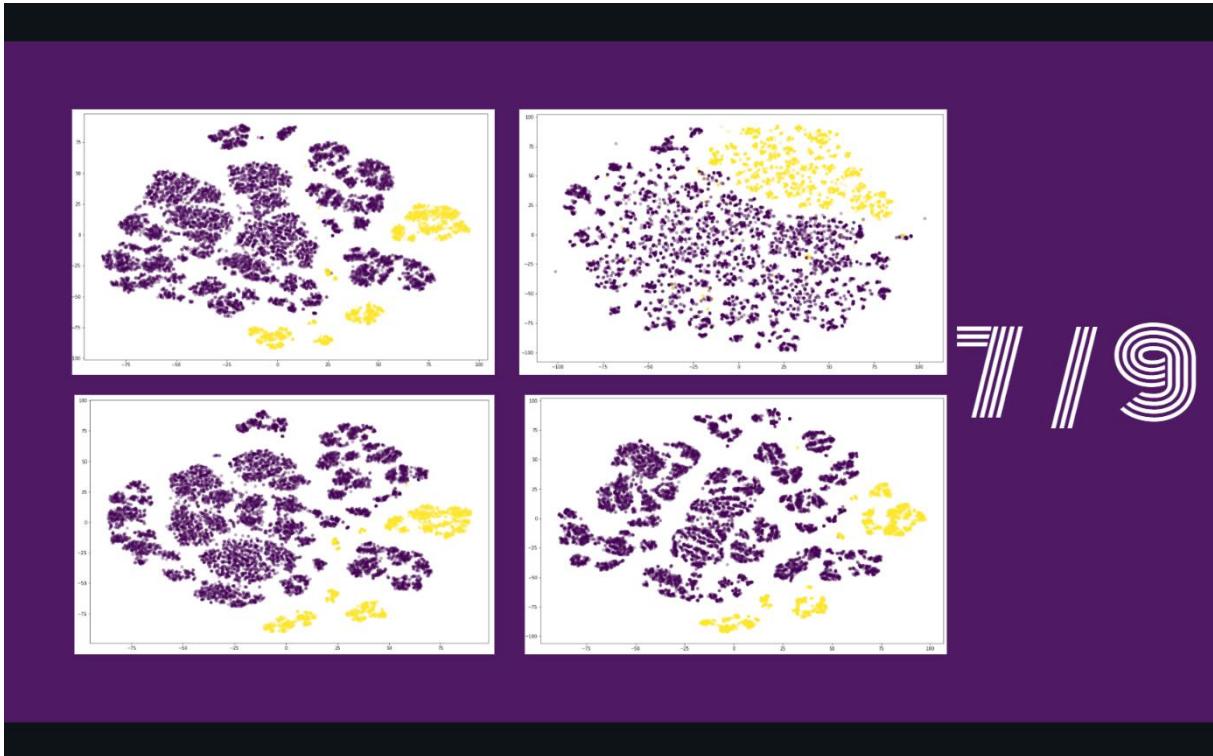


W kwestii PCA postanowiliśmy postawić na dwie wersje wygenerowanych sztucznych podziałów na zredukowanych wymiarach – na płaszczyźnie i w 3D. Tak prezentuje się podział w 2D...

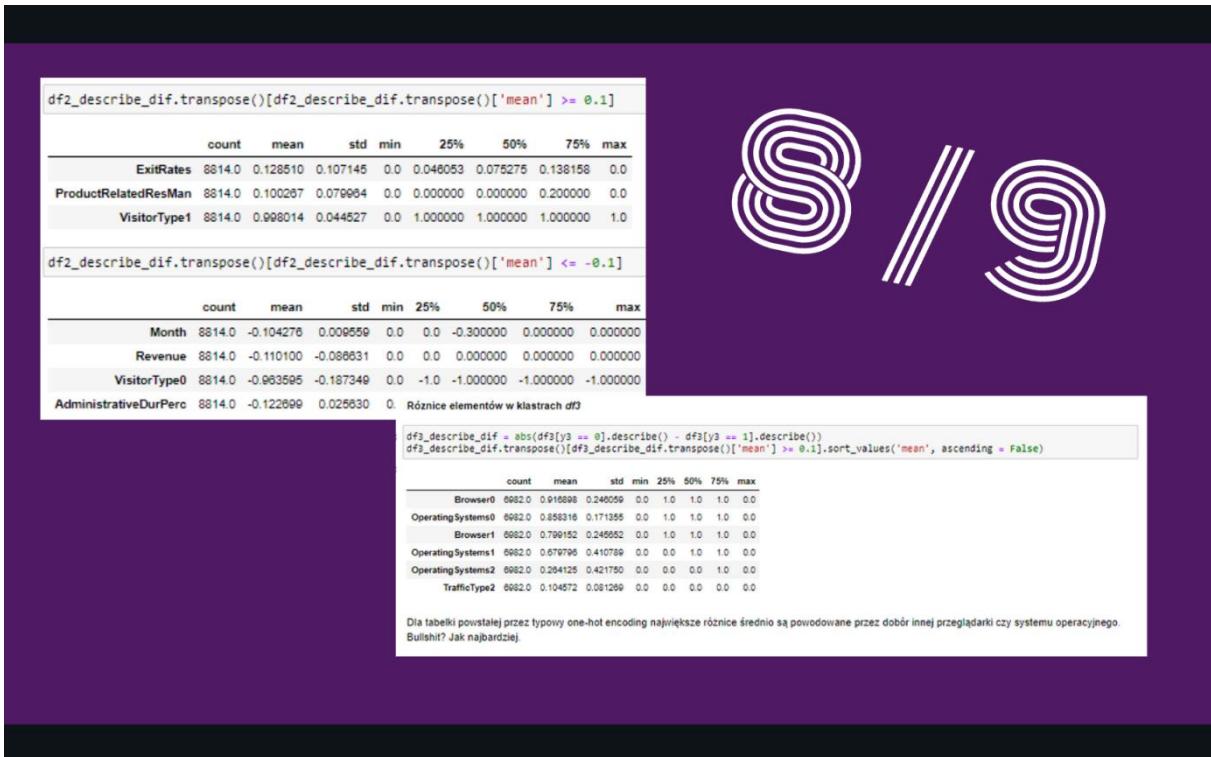


... a tak w trzech wymiarach. Zauważmy, że mimo pozornie bardzo podobnego efektu dla tych pierwszych czterech obrazków, tutaj widzimy już inne rezultaty – tak jak w obu przypadkach podział df3 zdecydowanie możemy wyrzucić do kosza,

wprowadzenie współrzędnych kołowych poskutkowało jednak ładniejszym podziałem w R3. W tym wypadku akurat ciężej o widoczne różnice między dwoma metodami encodingu, a wręcz możemy zaobserwować niemalże ten sam efekt – bierze się to prawdopodobnie z samej koncepcji i implementacji PCA.



Oprócz PCA dokonaliśmy także redukcji wymiarów i wizualizacji tSNE – co ciekawe tutaj inne metody poskutkowały tym, że to właśnie df3 wydaje się być najsensowniejsze.

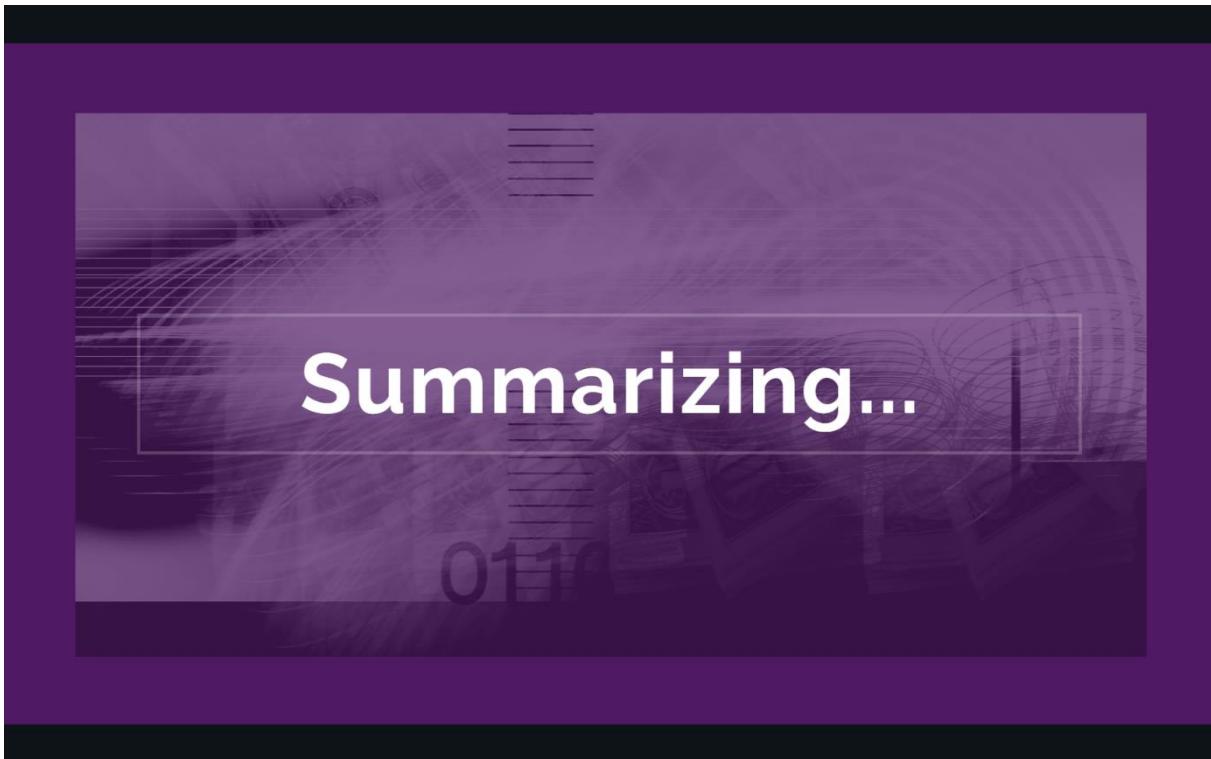


Redukcja wymiarów redukcją wymiarów, ale jak właściwie mają się do siebie nasze klastry? Żeby się o tym przekonać, dla każdej ramki danych obliczyliśmy liczności mniej bogatej w obserwacje grupy, a także wywołaliśmy funkcję `description()` dla wierszy przyporządkowanych do obu klastrów ku odkryciu podstawowych miar położenia dla wartości z kolumn. Następnie odjeliśmy wyniki `description` od siebie i przyjeliśmy, że mamy styczność z dosyć odmiennymi wartościami z kolumn wtedy, kiedy różnica wartości między nimi będzie wynosić przynajmniej 0.1 – mogliśmy śmiało przyjąć tą stałą, gdyż dokonaliśmy w końcu normalizacji, a dowiedzieliśmy się już, że przeglądarki czy systemy operacyjne nie wykazały dużego wpływu na KMeans. Także i tutaj okazało się, że wyniki dla df3 i pozostałych trzech analizowanych ramek wyszły z grubsza inne – i dla tej pierwszej efekt nie był zbyt satysfakcjonujący. Jak było bardzo dużo kolumn utworzonych przez one-hot encoding... tak tu użytkownicy zostali podzieleni po przeglądarkach i systemach operacyjnych. I tak jak w kontekście faktu, że większość jak już zauważaliśmy używa po dwóch z nich taki podział ma sens, jest on raczej niepraktyczny i niesatyfakcjonujący zestawiając inne features z ramki danych, które są już niemalże absolutnie losowe.

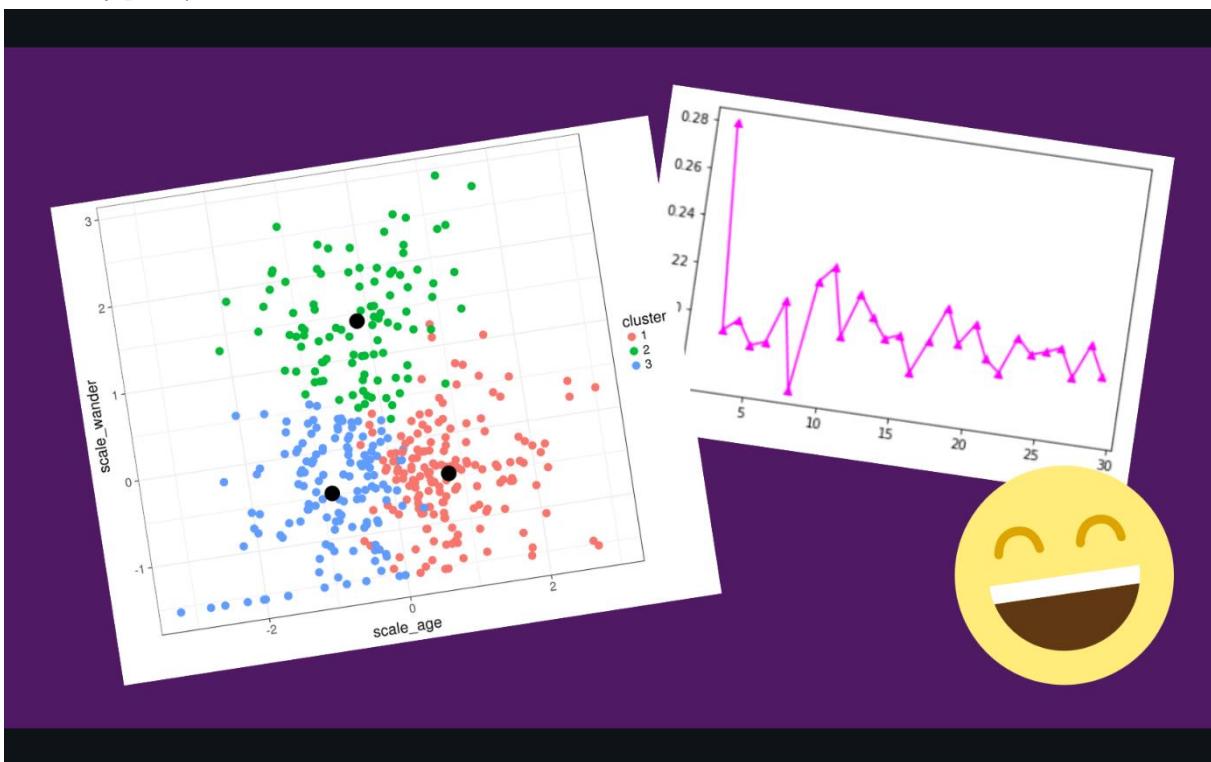
9 / 9

| Różnice elementów w klastrach df5 | | | | | | | | |
|-----------------------------------|--------|----------|----------|-----|----------|----------|----------|----------|
| | count | mean | std | min | 25% | 50% | 75% | max |
| VisitorType1 | 8808.0 | 0.998297 | 0.041235 | 0.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| VisitorType0 | 8808.0 | 0.981953 | 0.191363 | 0.0 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| ExitRates | 8808.0 | 0.128258 | 0.107149 | 0.0 | 0.045993 | 0.075214 | 0.138158 | 0.000000 |
| AdministrativeDurPerc | 8808.0 | 0.121844 | 0.025412 | 0.0 | 0.000000 | 0.516102 | 0.082913 | 0.001088 |
| Revenue | 8808.0 | 0.109838 | 0.086343 | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| MonthSin | 8808.0 | 0.104860 | 0.009619 | 0.0 | 0.000000 | 0.300000 | 0.000000 | 0.000000 |
| MonthCos | 8808.0 | 0.104860 | 0.009619 | 0.0 | 0.000000 | 0.300000 | 0.000000 | 0.000000 |
| ProductRelatedResMan | 8808.0 | 0.100641 | 0.079968 | 0.0 | 0.000000 | 0.000000 | 0.200000 | 0.000000 |

Dla df2, df4 i df5 wynik za to okazał się już zdecydowanie satysfakcyjny – klienci zostali bowiem pogrupowani po... fakcie czy byli na witrynie po raz pierwszy, czy może zaś już na nią wracali! Może niekoniecznie w 100%, ale jak wskazują statystyki – w wartości bliżej stu. Tutaj jednak taki podział jest zdecydowanie uzasadniony – jak mogliśmy się zresztą przekonać patrząc na załączoną różnicę tabel od describe(), okazało się przy tym, że wraz z ową własnością obserwacji wiąże się wiele innych właściwości – i tak oto na przykład ci powracający okazali się być także częściej kupującymi, przeglądającymi więcej stron związanych z produktem czy spędzającymi mniej czasu na witrynach administracyjnych. Ma to sens? Zdecydowanie, osiągnęliśmy sukces!

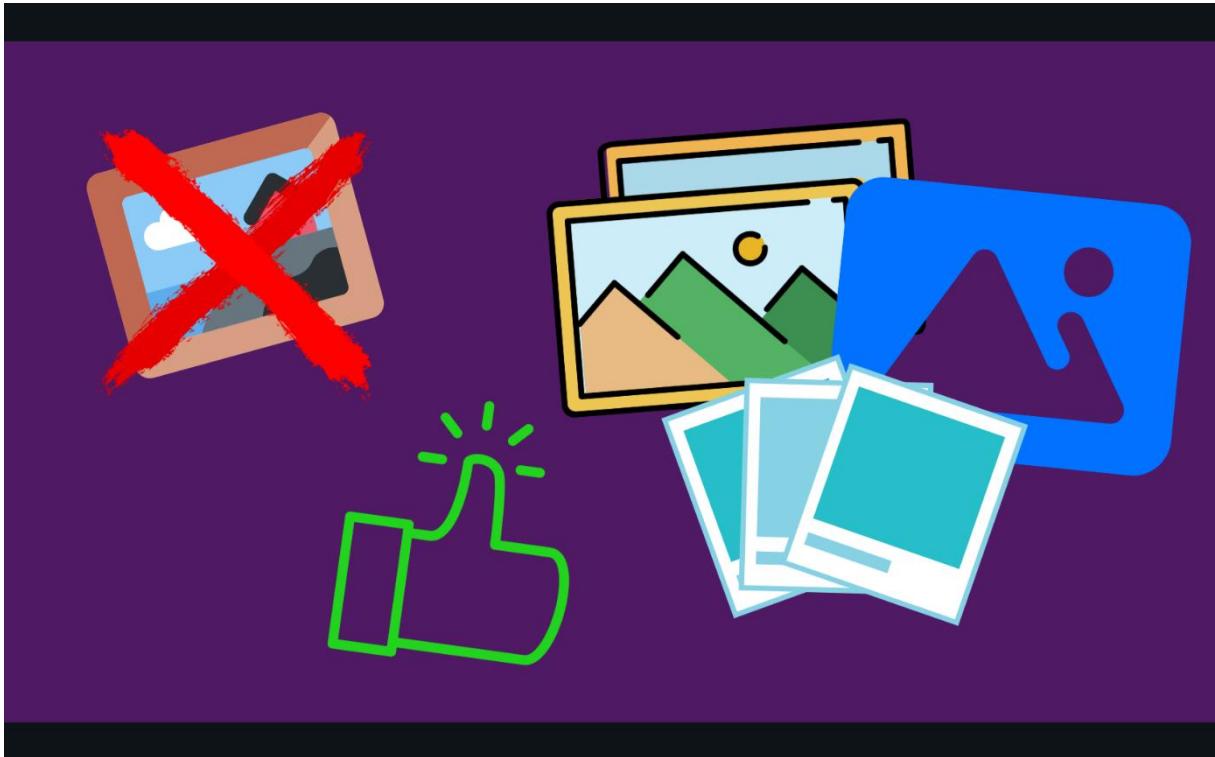


Tyle w kwestii utworzonego kodu. Ale podsumowując, co właściwie możemy powiedzieć o naszej pracy? Jakie wartościowe wnioski nam ona dała?



Po pierwsze – KMeans po odpowiedniej obróbce danych radzi sobie całkiem dobrze. Uzyskane klastry mają sens, a dobre miary, wizualizacje i porównanie miar położenia potwierdzają, że algorytm zrobił porządną robotę – w naszym wypadku zostali odróżnieni użytkownicy powracający i będący na witrynach po raz pierwszy. Silhouette

score jest zaś całkiem satysfakcjonującą miarą, jeżeli szukamy odpowiedniej liczby klastrów.



W drugiej kolejności – warto wspomnieć, że redukcja wymiarów jednego typu zdecydowanie nie załatwia sprawy. Dla wszystkich czterech ramek danych, wizualizacje PCA i tSNE dały nam zupełnie odwrotne efekty – w celu uzyskania pełnej satysfakcji z wygenerowanych obrazków należy lepiej zrozumieć owe metody, bądź dokonać kilku innych sposobów zapoznania się z właściwościami klastrów i porównania elementów między nimi – chociażby tak, jak dokonaliśmy tego z chwytrem różnic tabelek `describe()` i przyjrzenia się najbardziej różniącym miarom położenia.



Użycie współrzędnych kołowych dla miesięcy to dobry ruch, który może – lecz nie musi – przynieść fajniejsze rezultaty dla klasteryzacji. W naszym wypadku z jednej strony wprowadzenie ich dało gorszy sillhouette score, ale z drugiej przy redukcji wymiarów można było już zaobserwować o wiele bardziej satysfakcjonujące efekty.

Podsumowując: współrzędnym kołowym dla miesięcy, tygodni czy innych podobnych mówimy TAK.

0 - not is, 1 - is

0 - not is, $[2 / \text{<number of unique values in category column>}] - is$

df5['Region0'].sample(10)

| 12300 | 0.000000 |
|-------|----------|
| 2866 | 0.000000 |
| 12319 | 0.222222 |
| 4013 | 0.222222 |
| 7807 | 0.222222 |
| 439 | 0.000000 |
| 9994 | 0.222222 |
| 11111 | 0.222222 |
| 11357 | 0.000000 |
| 8319 | 0.000000 |

OperatingSystems
Browser 8
Region 13
TrafficType 9
dtype: int64 20

I wreszcie! Nieodpowiednie i zbyt automatycznie użycie one-hot encoding może destrukcyjnie wpływać na proces uczenia nienadzorowanego. Pomysł zastąpienia wartości odpowiadających za obecność danej kategorii przy wygenerowanych kolumnach jakąś stałą, chociażby 0.3, nie jest zły, ale jeszcze lepiej jest przypisać wartość zależną od liczby unikalnych kategorii – w naszym wypadku postawiliśmy na liczbę obliczoną z prostego wzoru, jakim jest iloraz dwójki i liczby unikalnych wartości w danej kolumnie. Przynajmniej dla *online shoppers purchasing intention* był to strzał w dziesiątkę. Dzięki takiemu podejęciu ważności kolumn kategorycznych o podobnej istotności się bilansują i jest to proces niezmiernie pomocny zwłaszcza w sytuacji, gdy mało wiemy o naszych kategoriach i nie są one równomiernie rozłożone. Tej metodzie naprawdę warto poświęcić pięć minut!