



Jak zbudować prosty system rekomendacji w sklepie e-commerce

Od co najmniej dwóch dekad trwa ekspansja sprzedaży detalicznej do sfery internetowej. Tak długi okres pozwolił wykształcić pewne metody i sposoby przyciągania klientów oraz utrzymywania ich lojalności do sklepów internetowych.

Sklepy e-commerce różnią się od typowych sklepów detalicznych między innymi tym, że klienci nie są tam anonimowi. Oczywiście sklepy detaliczne próbują identyfikować klientów poprzez tak zwane „karty lojalnościowe”. Dzięki temu są w stanie analizować zachowania klientów, ich preferencje lub częstotliwość ich wizyt w sklepie. Jednak to sklepy internetowe mają dużo większe możliwości analizy zachowań klientów. Osoba, która jest zalogowana na platformie sprzedażowej jest również obserwowana przez systemy nadzorujące sprzedaż na tej platformie. Dobrze skonfigurowane systemy e-commerce obserwują na jakie obszary witryny wchodził klient, jak długo obserwował produkty, które produkty wyświetlał, które produkty porównywał ze sobą. W ten sposób gromadzona jest ogromna wiedza historyczna dotycząca każdego klienta, która pozwala budować przewagę konkurencyjną.

Systemy rekomendacji

Systemy rekomendacji to urządzenia, które mają na celu zwiększenie zakupów dokonywanych przez indywidualnego klienta lub utrzymanie lojalności tego klienta w dłuższym terminie. Zazwyczaj próbuje się osiągnąć te dwa cele jednocześnie. W dłuższej perspektywie uważa się, że rozsądnie jest skoncentrować system rekomendacji bardziej na wzmacnianiu lojalności klienta.

Istnieją dwa główne kierunki analizy związanej z tworzeniem systemów rekomendacji. Po pierwsze analizuje się podobieństwo produktów i pod kątem tego podobieństwa proponuje się klientowi kolejny produkt. Po drugie – obserwuje się klientów, szuka ich wspólnych cech, tworzy segmenty, grupy i klastry. W efekcie nie tylko proponuje się konkretne produkty podczas zakupu lecz również wysyła się do klientów zachęty w postaci smsów, mejli i wszelkiego rodzaju powiadomień.

W systemach rekomendacji oba kierunki analizy można stosować jednocześnie. Metodologia rekomendacji oparta na podobieństwach produktów może się przydać kiedy klient wykona pierwszy krok, wybierze określony produkt. Nasz system będzie wtedy

podsuwał mu zdjęcia podobnych produktów. Jeżeli klient tego nie zrobi system rekomendacji oparty wyłącznie na podobieństwie produktów nie zadziała. Aby klient wykonał pierwszy krok, warto przyciągać go do sklepu, do konkretnych grup produktów lub określonych usług, które mogą być świadczone przez witrynę internetową.

W artykule zajmiemy się metodą tworzenia możliwie najprostszego systemu rekomendacji, opartego na podobieństwie produktów. System czerpie wiedzę o produktach z ich opisów oraz zdjęć, na podstawie których będzie tworzył oferty dla klientów. Opiszę w jaki sposób zbudować taki system rekomendacji w języku Python.

Zasada cosinusa

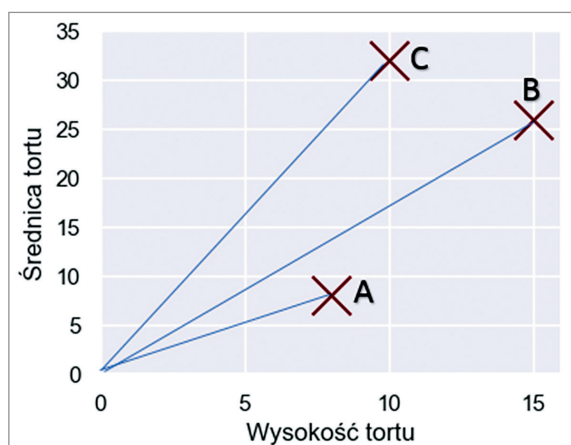
Zasada cosinusa w systemach rekomendacji jest wykorzystywana głównie w metodzie filtracji kolaboratywnej, czyli tam, gdzie porównywane są własności użytkowe sprzedawanych produktów. Metoda polega na obliczeniu podobieństwa między przedmiotami na podstawie ocen użytkowników lub wprowadzonych na stronie platformy e-commerce opisów. Jeśli mamy dwa przedmioty, każdy z nich reprezentowany przez wektor ocen od różnych użytkowników, to cosinus kąta między tymi wektorami w przestrzeni wielowymiarowej jest miarą ich podobieństwa. Im bliższa wartości 1 cosinusa, tym przedmioty są bardziej podobne.

Aby trochę przybliżyć ten enigmatyczny opis posłużę się przykładem. Mamy trzy torty:

- tort A ma średnicę 8 cm oraz wysokość 8 cm,
- tort B ma średnicę 32 cm i wysokość 10 cm
- tort C ma średnicę 26 cm i wysokość 15 cm.

Wyroby:	Wysokość tortu	Średnica tortu
Tort A	8 cm	8 cm
Tort B	10 cm	32 cm
Tort C	15 cm	26 cm

Wysokość oraz średnicę tortów umieszczamy w układzie współrzędnych.



Na powyższym wykresie zazaczyłem miejsce, w których znajduje się każdy tort. Oś x przedstawia wysokość tortu, oś y – średnicę tortu. Przeprowadziłem linię od punktów obrazujących torty do początku układu współrzędnych. Czym ostrzejszy jest kąt między poszczególnymi liniami, tym bardziej podobne są produkty. Na powyższym wykresie widać, że tort A bardzo różni się od tortu C, natomiast teoretycznie różnica między tortem B i A jest niewielka, ponieważ kąt pomiędzy liniami jest stosunkowo ostry.

Podobieństwa dzięki opisom produktów

Aby stworzyć efektywny system rekomendacji trzeba analizować zachowania klientów oraz szukać podobieństwa wśród sprzedawanych towarów. Jeżeli klient jest nowy nie jest możliwe przyporządkowanie go do określonego segmentu. Klasy takie buduje się na podstawie zachowań klientów. Cechy takie jak płeć, wiek czy miejsce zamieszkania w procesie budowania klastrów mają drugorzędne znaczenie. Jeżeli klient jest nowy nie można stworzyć systemu rekomendacji opartego na przynależności klienta do określonego klastra. Pozostaje więc tworzenie tego systemu na bazie podobieństwa produktów. Znalazienie tego podobieństwa może być możliwe na podstawie zdjęć lub opisów. Podobieństwo produktów uzyskane na drodze analizy zdjęć może wykonać sztuczna inteligencja dzięki sieciom neuronowym rozpoznającym obrazy. Drugą metodą, którą można zastosować równolegle jest identyfikacja podobieństwa na podstawie zamieszczonych opisów produktów. Mogą je tworzyć klienci lub personel sklepu. Klienci stanowią grupę stosunkowo niestabilnych osób. Zwykle decydują się zamieścić komentarze do produktów gdy są wzburzeni. W ich opisach mogą znajdować się rozmaite skrajne

oceny, od euforii po wściekłość. Moglibyśmy szukać podobieństwa pomiędzy produktami na podstawie stopnia zaangażowania emocjonalnego klientów. Takie działania nie miałyby prawdopodobnie większego sensu. Na przykład klienci, którzy wyświetlili wywołujący bardzo dużo emocji tort otrzymaliby jako rekomendowane inne bardzo ekscytujące torty. Wydaje się, że jedynym rozsądnym rozwiązaniem problemu skutecznego systemu rekomendacji jest szukanie podobieństwa produktów w opisach dokonanych przez obsługę sklepu.

Przykład systemu rekomendacji dla cukierni

Wyobraźmy sobie witrynę e-commerce sprzedającą duże ilości produktów cukierniczych. Jedną z kategorii sprzedawanych w tym sklepie są torty. Sklep oferuje 3500 różnych rodzajów tortów. Zazwyczaj, gdy klient ogląda jakiś tort, witryna poniżej proponuje 5 innych podobnych tortów. Od tego na ile trafne będą zaproponowane inne produkty zależy efektywność sprzedaży. Klient ma do dyspozycji wiele podobnych sklepów internetowych z tortami. Jeżeli proponowane przez witrynę torty będą nie-trafne, klient po prostu opuści witrynę i kupi tort w innym miejscu.

W szukaniu podobieństwa produktów będziemy korzystali z opisanej wyżej metody ostrości kątów cosinus. Wykorzystamy bazę tortów, która była bezpośrednio używana do wyszukiwania produktów w witrynie internetowej. Poniżej znajduje się lista pól bazy danych. Pola zostały sporządzone po angielsku, ponieważ narzędzie, które służy do tworzenia systemów rekomendacji nie obsługuje języka polskiego.

Gender (*female, male*) – pole to oznacza płeć; klient przy wyszukiwaniu tortu najprawdopodobniej zwróci na to uwagę;

Type (*birthday, communion, professional, anniversary*) – typ tortu jest bardzo istotny dla klientów. Inny kształt mają torty komunijne, ślubne, urodzinowe lub związane z obchodzeniem uroczystości zawodowych;

Size (*tiny, small, medium, large, huge*) – wielkość;

Color (*black, white, yellow, green, brown, blue, lemon*) – kolor;

Writings (*string, no string*) – napis na torcie lub brak napisu; dla jednych klientów hasło na torcie ma bardzo duże znaczenie, dla innych jego brak;

Height (*high, low*) – pole to pokazuje czy tort jest wysoki czy niski;

Figure (*flowers, animals, cars, none, people*) – na torcie pojawić się mogą figurki ze zwierzętami, kwiatami, samochodem lub parą młodą;

Medical (*high-calorie, low-calorie, lactose intolerance, gluten intolerance, histamine intolerance*) – dla niektórych klientów istotne jest, aby tort spełniał ich zalecenia medyczne. Zaleceniem takim może być: niska ilość kalorii czy uwzględnienie nietolerancji laktozy lub histaminy;

Sport (*none, ball, basketball, cycling, swimming*) – częstym motywem tortów dla dzieci są dyscypliny sportowe, postanowiłem uwzględnić to w mojej bazie tortów.

Baza ta została stworzona przeze mnie w języku Python. Każdy może stworzyć sobie taką bazę treningową. Poniżej przedstawiam kod w jaki sposób stworzyłem tę bazę.

```
import pandas as pd

np.random.seed(148)
type = np.random.default_rng().integers(0, 4, size=3500)
gender = np.random.default_rng().integers(0, 2, size=3500)
size = np.random.default_rng().integers(0, 5, size=3500)
color = np.random.default_rng().integers(0, 6, size=3500)
writings = np.random.default_rng().integers(0, 2, size=3500)
height = np.random.default_rng().integers(0, 2, size=3500)
figurine = np.random.default_rng().integers(0, 5, size=3500)
medical = np.random.default_rng().integers(0, 5, size=3500)
sport = np.random.default_rng().integers(0, 5, size=3500)

index = list(np.random.permutation(np.arange(0,3500)))

key = ['index', 'type', 'gender', 'size', 'color', 'writings', 'height', 'figurine', 'medical', 'sport']
vals=[index,type, gender, size, color,writings,height, figurine,medical, sport]

data = {key:vals[n] for n, key in enumerate(key)}

df=pd.DataFrame(data)

df['gender'] = df['gender'].map({0:'female', 1:'male'})
df['type'] = df['type'].map({0:'birthday', 1:'communion', 2:'professional', 3:'anniversary'})
df['size'] = df['size'].map({0:'tiny', 1:'small', 2:'medium', 3:'large', 4:'huge'})
df['color'] = df['color'].map({0:'black', 1:'white', 2:'yellow', 3:'green', 4:'brown', 5:'blue', 6:'lemon'})
df['writings'] = df['writings'].map({0:'string', 1:'no string'})
df['height'] = df['height'].map({0:'high', 1:'low'})
df['figurine'] = df['figurine'].map({0:'flowers', 1:'animals', 2:'cars', 3:'none', 4:'people'})
df['medical'] = df['medical'].map({0:'high-calorie', 1:'low-calorie', 2:'lactose intolerance', 3:'gluten into'})
df['sport'] = df['sport'].map({0:'none', 1:'ball', 2:'basketball', 3:'cycling', 4:'swimming'})

df['index'] = 'Id_' + df['index'].astype(str)
df
```

Baza tortów po jej wygenerowaniu z powyższego kodu wygląda następująco:

	index	type	gender	size	color	writings	height	figurine	medical	sport
0	Id_109	birthday	male	small	yellow	no string	high	flowers	high-calorie	swimming
1	Id_3252	birthday	female	tiny	white	string	high	flowers	histamine intolerance	swimming
2	Id_792	birthday	male	large	white	string	high	people	gluten intolerance	none
3	Id_1704	communion	female	large	yellow	string	low	cars	lactose intolerance	cycling
4	Id_3380	communion	male	large	black	string	high	flowers	low-calorie	ball
...	...	...	...	...	...	...	...	...	...	...
3495	Id_1881	communion	female	small	yellow	string	high	flowers	low-calorie	ball
3496	Id_1488	birthday	female	medium	yellow	no string	high	animals	lactose intolerance	cycling
3497	Id_115	communion	male	tiny	brown	no string	low	flowers	low-calorie	swimming
3498	Id_19	communion	male	small	black	no string	high	animals	histamine intolerance	none
3499	Id_3155	anniversary	female	large	blue	no string	high	people	low-calorie	basketball

Index oznacza numer identyfikacyjny tortu. W bazie nie występują nazwy tortów, ponieważ informacja ta nie ma znaczenia dla systemu rekomendacji.

Teraz muszę stworzyć pole z opisem. Zazwyczaj w sklepach istnieje obszerne pole z opisem, które zawiera cechy użytkowe, wymiary oraz przeznaczenie produktów. Stworzenie takiego pola dla 3500 tortów byłoby dla

mnie bardzo kłopotliwe. Z drugiej strony może zdarzyć się sytuacja, w której nie istnieje takie pole opisowe, albo pole to z jakichś powodów nie nadaje się do wykorzystania w tworzonego systemu rekomendacji. Aby stworzyć efektywne pole opisowe dla systemu rekomendacji należy połączyć wszystkie cechy tortów w jeden ciąg

wyrazów. Nowe pole bazy składające się z poszczególnych elementów będzie nazywało się *description*.

Teraz, aby system mógł wyszukać podobieństw musi zostać przeprowadzony proces tokenizacji. Tokeny to wyrazy, które łączą się z innymi tokenami – wyrazami. System wyszukuje wyrazy wskazanego przez klienta produktu z pola *description* i szuka najbardziej podobnych produktów również w polu *description* w bazie 3500 tortów. System robi to za pomocą poniższego kodu.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['description'])
```

Następnie tworzona jest macierz tokenów, w której można już zastosować opisaną wyżej metodę wyszukiwania podobieństwa za pomocą kontry kosinusa.

Poniżej znajduje się kod, za pomocą którego utworzono macierz o wymiarach 3500 na 3500 tortów. Wewnątrz macierzy znajdują się wartości podobieństw pomiędzy poszczególnymi tortami.

Wartość 1 oznacza doskonałe podobieństwo. Czym bardziej wartość kosinusa zbliża się do zera, tym niższe jest podobieństwo tortów.

```
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim.shape
(3500, 3500)

cosine_sim
array([[1., 0.21562226, 0.36307865, ..., 0.26796626, 0.56518665,
        0.53794042],
       [0.21562226, 1., 0.18403603, ..., 0.27142974, 0.19726848,
        0.19086834],
       [0.36307865, 0.18403603, 1., ..., 0.24778215, 0.33373893,
        0.34091757],
       ...,
       [0.26796626, 0.27142974, 0.24778215, ..., 1., 0.38988561,
        0.37945701],
       [0.56518665, 0.19726848, 0.33373893, ..., 0.38988561, 1.,
        0.19629135],
       [0.53794042, 0.19086834, 0.34091757, ..., 0.37945701, 0.19629135,
        1.]])
```

Możemy teraz sprawdzić jak podobne są do siebie dwa wybrane torty. Wystarczy podstawić do poniższego kodu indeksy wybranych tortów, aby dowiedzieć się jaki jest stopień podobieństwa obu produktów. Podstawiliśmy torty o indeksach: Id_3360 i Id_1486.

```
a = df.loc[df['index'] == 'Id_3360'].index.tolist()
b = df.loc[df['index'] == 'Id_1486'].index.tolist()
cosine_sim[a,b]

array([0.55889216])
```

Aby zbudować użyteczny system rekomendacji oparty na podobieństwie produktów musimy wpisać poniższy kod.

```
def get_recommendations(index, cosine_sim=cosine_sim, num_recommend = 10):

    idx = indices[index]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    top_similar = sim_scores[1:num_recommend+1]
    movie_indices = [i[0] for i in top_similar]

    return df['index'].iloc[movie_indices]
```

Teraz, aby uzyskać 5 podobnych tortów, wystarczy wpisać poniższy kod, w który wpisaliśmy również kod tortu wybranego przez klienta.

```
indices = pd.Series(df.index, index=df['index']).drop_duplicates()

kot = get_recommendations('Id_3252', num_recommend = 5).to_dict()
ind = list(kot.keys())
Id = list(kot.values())
```

Klient obsługujący wyszukiwarkę wybrał tort o indeksie Id_3252. Jest to duży tort komunijny dla dziewczynki w kolorze niebieskim. Dla klienta istotne było, żeby tort miał mało kalorii. Na zdobieniach tortu miały znajdować się elementy związane z piłką. Oto dokładna charakterystyka tortu o indeksie Id_3252 wybranego przez klienta.

```
df[df["index"] == 'Id_3252']
```

	index	type	gender	size	color	writings	height	figurine	medical	sport
1	Id_3252	communion	female	large	blue	no string	low	none	low-calorie	ball

Jak widzimy w poniższej tabelce system zaproponował pięć tortów, spośród których cztery są dla dziewczynki i jeden dla chłopca. Wszystkie torty są duże i niebieskie. Co ważne wszystkie torty znajdują się w kategorii produktów niskokalorycznych. Część tortów zawiera opcję napisu oraz elementy związane z piłką.

```
df[df["index"].isin(Id)]
```

	index	type	gender	size	color	writings	height	figurine	medical	sport
455	Id_1891	communion	female	large	blue	string	high	none	low-calorie	none
837	Id_2488	communion	female	large	blue	no string	high	people	low-calorie	ball
1042	Id_1952	communion	female	large	blue	string	low	none	low-calorie	cycling
1146	Id_124	communion	female	large	blue	string	low	animals	low-calorie	ball
2335	Id_2283	communion	male	large	blue	no string	low	flowers	low-calorie	ball

Jak widzimy, system rekomendacji zaproponował dosyć podobne produkty. Zdarza się jednak, że proponują przedmioty, które wprawdzie są podobne, ale są nie do zaakceptowania przez klienta.

Praca na wydzielonych bazach

W naszym przykładzie wśród zaproponowanych tortów pojawił się tort dla chłopca.

Na przykład klient wybrał kurtkę męską XXL. W odpowiedzi system rekomendujący podpowiada klientowi inne

kurtki, w tym kurtkę w rozmiarze L albo kurtkę damską. Każdy źle zaproponowany produkt wiąże się z utratą części prawdopodobieństwa zakupu produktu. Pomijam już fakt, że klienta może zrytować zaproponowana mu przez system różowa kurtka w rozmiarze S.

Aby uniknąć takich sytuacji wyszukiwania podobnych produktów powinny być prowadzone na wydzielonych

zbiorach. Jeżeli klient wybrał kurtkę męską w rozmiarze XXL, to system rekomendacji powinien wybierać podobne produkty tylko i wyłącznie w wydzielonym zbiorze wśród męskich kurtek XXL. Aby dokonywać szybkich wybierania zbiorów przed właściwym procesem wyboru podobieństw należy zastosować poniższy linijkę kod.

```
dk = df.loc[(df['gender'].apply(lambda x: x in ['male'])) & (df['size'].apply(lambda x: x in ['XXL']))]
```

Gdzie dk oznacza wydzielony zbiór z ogólnej bazy df, zawierający tylko męski rozmiar XXL.

Podsumowanie

Jak widzimy, w pełni sprawny system rekomendacji, który może zwiększyć prawdopodobieństwo sprzedaży produktów o kilkadziesiąt procent składa się z zaledwie kilkunastu linijek kodu. Kod ten wystarczy skopiować do systemu obsługującego witrynę sklepu internetowego, aby witryna ta proponowała klientom trafne produkty.

Handel internetowy w Polsce staje się coraz dojrzalszy. Nawet najdrobniejsza poprawa efektywności komunikacji z klientem może przyczynić się do dodatkowej sprzedaży. Przedstawiony dzisiaj system rekomendacji składa się z kilku-

nastu linii kodu. Ten w pełni efektywny system może spowodować uzyskanie istotnej przewagi konkurencyjnej i sukces finansowy sklepu internetowego.

Niedawno byłem w sklepie internetowym pewnej dużej firmy budowlanej. Szukałem wiertarki w dziale elektronarzędzi. Za każdym razem system proponował mi rękawice robocze oraz przedłużacze. Oznaczało to, że strona ta nie miała systemu rekomendacji, a proponowane produkty były produktami najczęściej kupowanymi.

Wojciech Moszczyński

Wojciech Moszczyński – absolwent Katedry Ekonometrii i Statystyki Uniwersytetu Mikołaja Kopernika w Toruniu, specjalista z zakresu ekonometrii, finansów, *data science* oraz rachunkowości zarządczej. Specjalizuje się w optymalizacji procesów produkcyjnych i logistycznych. Prowadzi badania w obszarze rozwoju i zastosowania sztucznej inteligencji. Od lat zaangażowany w popularyzację *machine learning* oraz *data science* w środowiskach biznesowych.