System rekomendacji w młynie (cz. 3)

Vertex AI Recommendations – usługa rekomendacji Google

Vertex AI Recommendation (czasem nazywana Retail API) to usługa analogiczna do Amazon Personalize, oferowana przez Google, ukierunkowana zwłaszcza do **handlu internetowego**. Podobnie jak AWS Personalize, jest to w pełni zarządzana usługa, w której **Google** odpowiada za trenowanie i hostowanie modeli rekomendacyjnych, a my dostarczamy dane i wywołujemy API, aby uzyskać rekomendacje. Google reklamuje ją tym, że to ta sama technologia, która zasila rekomendacje w produktach Google (np. YouTube czy Google Shopping).

Dane wejściowe

Do uruchomienia Vertex AI Recommendation potrzebujemy dwóch podstawowych rzeczy: katalogu produktów oraz danych o aktywności użytkowników.

W usłudze Vertex AI Recommendations (dawniej Recommendations AI) model uczy się wyłącznie na tzw. *user events* (zdarzenia wykonane przez użytkownika). Są one z góry skategoryzowane; dla typowego sklepu e-commerce Google wskazuje cztery kluczowe zdarzenia (*priority user event types*) – bez nich model nie ruszy lub będzie słabej jakości.

Priorytet	eventType	Co oznacza w praktyce	Typowy log sklepu
1 (-required)	`detail-page-view`	użytkownik otwiera stronę produktu	GET /product/ID
1 (-required)	`add-to-cart`	dodaje produkt do koszyka	POST /cart/add
1 (-required)	`purchase-complete`	finalizuje transakcję	zapis zamówienia
2 (recommended)	`home-page-view`	wizyta na stronie głównej	GET /
(dla wyszukiwarki)	`search`	wyszukanie lub lista kategorii (event opcjonalny dla rekom.)	GET /search?q=…

Typ akcji użytkownika: *detail-page-view* mówi, co użytkownik naprawdę ogląda; *add-to-cart* i *purchase-complete* to najsilniejsze sygnały preferencji; *home-page-view* daje kontekst, ile osób w ogóle widziało rekomendacje, tworzy kontekst ekspozycji – system wie, ilu użytkowników miało szansę zobaczyć rekomendacje i może lepiej rozróżnić. Rozważane są sytuacje: „użytkownik nie kliknął, choć widział" (negatywny sygnał) od „użytkownik nie kliknął, bo w ogóle nie pokazano mu rekomendacji" (brak sygnału). Dzięki temu metryki (CTR, konwersja) i proces uczenia rankingów są dokładniejsze, a model szybciej uczy się, które produkty rzeczywiście przyciągają uwagę.

W Vertex AI Recommendation każdy *home-page-view* rejestrowany w strumieniu User Events oznacza, że użytkownik zobaczył stronę główną sklepu – a więc prawdopodobnie zobaczył również moduł z rekomendacjami, choć niekoniecznie w niego kliknął.

Jak przygotować dane

Krok 1. Zmapuj logi aplikacyjne na powyższe cztery kategorie:

- wejście na /product/{id} → *detail-page-view*
- akcja „Dodaj do koszyka" → *add-to-cart*
- finalizacja zamówienia → *purchase-complete* (tu dodaj revenue, currencyCode, quantity)
- wejście na / → *home-page-view*

Krok 2. Wyciągnij wymagane kolumny (timestamp, userId, productId, qty, revenue) w jobie Spark i zapisz np. do BigQuery w GCP.

Krok 3. Użyj importu BigQuery lub Cloud Storage

Google oczekuje plików JSON lub tabeli BQ o schemacie UserEvent. Import najlepiej robić codziennie, ale w naszym scenariuszu internetowego sklepu z mąką wystarczy raz w tygodniu – pod warunkiem, że nadal dostarczysz powyższe cztery typy eventów.

Jak wygląda taki przykładowy plik JSON dla jednego zdarzenia „Dodaj do koszyka" → *add-to-cart*

```
{
  "eventType": "add-to-cart",
  "eventTime": "2025-07-29T14:06:31Z",   // RFC-3339
  "userInfo": {
    "userId":   "12345"                  // lub visitorId jeśli anonimowy
  },
  "productEventDetail": {
    "productDetails": [
      { "id": "FLOUR_QNOA_1KG", "quantity": 2 }
    ]
  }
}
```

Pola obowiązkowe dla każdego eventu:

- eventType – jeden z wymienionej wyżej listy 4 zdarzeń: *detail-page-view*, *add-to-cart*, *purchase-complete*, *home-page-view*
- eventTime – znacznik czasu w formacie RFC 3339
- userInfo.userId lub visitorId – identyfikator klienta/sesji
- productEventDetail.productDetails[].id – identyfikator produktu z katalogu.

Krok 4. Zweryfikuj jakość

Po imporcie w konsoli Vertex AI przejdź do Data quality → upewnij się, że masz ≥ 100 unikalnych visitorId dla każdego z detail-page-view, add-to-cart, purchase-complete; w przeciwnym razie model może się nie wytrenować.

Oprócz 4 zdarzeń Vertex AI może użyć również innych informacji. Google nie gwarantuje, że modele rankingowe użyją tych *custom-attributes*, ale dopuszcza je w specyfikacji UserInfo – warto więc je przesłać, zwłaszcza przy mniejszej liczbie klasycznych eventów.

Przykład dodatkowej informacji dla Vertex AI „skłonność do promocji", który nie jest osobnym eventType.

```
"userInfo": {
  "userId": "12345",
  "customAttributes": {
    "promo_affinity": { "stringValue": "HIGH" },
    "decision_speed": { "stringValue": "SLOW" }
  }
}
```

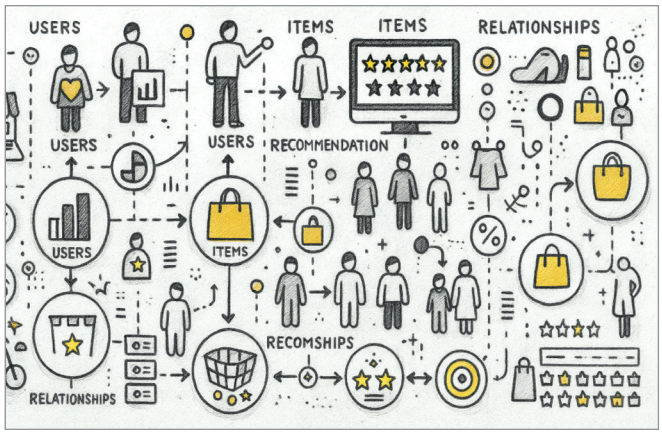
Te cztery typy informacji: *detail-page-view* + *add-to-cart* + *purchase-complete* + *home-page-view* to absolutne minimum, aby Vertex AI Recommendations osiągnął dobrą jakość w scenariuszu e-commerce.

Skąd wziąć wymienione typ plików opisujących zachowania klientów?

Plików typu JASON niezbędnych do pracy Vertex AI Recommendations nie generuje Google. Vertex AI Recommendations (dawniej Recommendations AI) udostępnia schemat JSON i publiczne REST API (projects/locations/catalogs/eventStores/userEvents). To właściciel sklepu lub administrator strony musi zebrać logi, zamienić je na wiersze ND-JSON (newline-delimited JSON) i załadować:

- w czasie rzeczywistym – wysyłając każdy event bezpośrednio do końcówki userEvents:collect (POST HTTP),
- wsadowo, czyli zgodnie z naszymi założeniami sklepu internetowego z mąką – zapisując plik ND-JSON w GCP w Cloud Storage lub tabeli w BigQuery wywołując operację import.

Źródło danych może być dowolne: serwer aplikacji, skrypt JavaScript na froncie, eksport z platformy e-commerce, logi Apache, GA4 → BigQuery, itp. Google tylko wymaga, aby każdy rekord miał wymagane cztery pola (eventType, eventTime, userInfo, productEventDetail).



Czy Vertex AI Recommendations jest „tylko dla sklepów Google"?

Vertex AI Recommendations to usługa PaaS dobudowywana do dowolnego sklepu lub aplikacji retail, o ile:

1. Utrzymuje się katalog produktów w formacie Retail API (ID, nazwa, kategorie, cena ...).
2. Dostarcza się zdarzenia użytkowników zgodne ze schematem 4 zdarzeń: *detail-page-view*, *add-to-cart*, *purchase-complete*, *home-page-view*.

Usługa nie wymaga hostingu sklepu na platformie Google ani użycia Google Merchant Center, choć integracja

z Merchant Center ułatwia import katalogu. Można więc zastosować to rozwiązanie do każdego sklepu internetowego. Wystarczy tylko wysyłać właściwe dane we właściwym formacie. W zamian otrzymamy rekomendacje.

Przykład minimalnego rekordu pliku JASON pojedynczego produktu w formacie Retail API

```
{                    // ↓ minimum wymagane pola
  "id": "FLOUR_QNOA_1KG",
  "categories": "Food, Baking > Flour > Gluten-Free",
  "title": "Mąka z quinoa BIO 1 kg"
}
```

Przykład bogatego rekordu pliku JASON pojedynczego produktu w formacie Retail API

```
{                    // ↓ bogatszy rekord z polami opcjonalnymi
  "id": "FLOUR_AMAR_500G",
  "categories": "Food, Baking > Flour > Ancient Grains",
  "title": "Mąka z amarantusa 500 g",
  "description": "Ekologiczna, wysokobiałkowa mąka z amarantusa – idealna do pieczywa be:
  "brands": ["Eko-Zboże"],
  "tags": ["organic", "high-protein"],
  "priceInfo": { "currencyCode": "EUR", "price": 4.79, "originalPrice": 5.50 },
  "availability": "IN_STOCK",
  "attributes": {
    "grain_type": { "text": ["amarantus"] },
    "country_of_origin": { "text": ["Peru"] },
    "protein_pct": { "float": [14.2] }
  },
  "languageCode": "pl"
}
```

Każdy wiersz musi być poprawnym obiektem JSON bez znaków nowej linii.

Minimalne wymagania: id, title, categories.

Zalecane pola handlowe: priceInfo, availability, description, tags, attributes (dowolne własne cechy, np. „bezglutenowa", „bio-certyfikowana").

Jak stworzyć i załadować katalog?

Wyciągasz dane ze swojej bazy produktów (Spark SQL w Synapse):

```
SELECT
  product_id        AS id,
  'Food, Baking > Flour > ' || main_category AS categories,
  name              AS title,
  description,
  brand,
  price             AS price,
  1                 AS availability,
  grain_type,
  origin_country,
  protein_pct
FROM dw_products;
```

Trzeba wyeksportować dane do formatu ND-JSON i dalej zaimportować je do Vertex AI.

Typowa ścieżka integracji danych z Vertex AI

1. Eksport danych: z logów aplikacyjnych serwera, bazy transakcyjnej lub GA4 wyciągane są zdarzenia (*view*, *add-to-cart*, *purchase*).

2. Transformacja: w Spark, Dataflow lub Python konwertuje się dane do ND-JSON zgodnego z Retail UserEvent.

3. Import: raz w tygodniu uruchamiany jest importUserEvents z pliku w Cloud Storage – batch lub przesyłany streamingowo podczas sesji użytkownika (collect API).

4. Trenowanie Vertex AI automatycznie retrenuje model, gdy pojawi się nowa partia zdarzeń lub wg harmonogramu.

5. Pobieranie rekomendacji: Aplikacja sklepu wywołuje endpoint predict podając userId lub visitorId i – opcjonalnie – kontekst (np. oglądany produkt).

Model Vertex AI Recommendations działa dla każdego hostowanego sklepu e-commerce, pod warunkiem że zostaną spełnione dwa warunki: dostarczony katalog oraz prawidłowo sformatowane zdarzenia użytkowników.

Jak wyglądają współpraca GCP ze stroną sklepu?

Z mojej pracy z chmurą GCP wynika, że środowisko to lubi pracować w czasie rzeczywistym, a nie w systemie bachowym. Oznacza to, że sklep z naszego przykładu jest na stałe połączony protokołem API z Vertex AI Recommendations. Taka współpraca sklepu z chmurą Google wyglądała by miej więcej tak.

- Strona klienta woła do chmury: „Użytkownik ID:12345. Jest 17:13 i właśnie patrzy na stronę główną na produkt XYZ. Co warto mu teraz pokazać?". Czyli strona wysyła paczkę danych JSON z trzema rzeczami: kto, kiedy, jaki kontekst.
- W odpowiedzi Vertex odsyła listę kilku identyfikatorów produktów BC, XZ i BZ ustawionych od „najpewniej trafionego" do „może też się spodoba". Vertex dołącza znaczek – sekret

attributionToken – coś jak przypinka „ta lista dotyczy zapytania numer 987”.

- Sklep to otrzymuje i wyświetla propozycje klientowi: Zamienia ID produktów: BC, XZ i BZ na zdjęcie, nazwę i cenę mąki z bazy; pokazuje je klientowi.
- Sklep obserwuje jak zareagował klient i melduje do Vertex AI co się stało. Sklep wysyła do Vertex krótką notkę „lista z tokenem 987 wyświetlona”.
- Jeżeli klient kliknął polecony produkt lub kupił, sklep wysyła drugą notkę: „klient kliknął/kupił produkt BZ z listy 987”.

Dzięki temu chmura wie, które sugestie były trafne, a które nie. Informacja ta jest podstawą do nauki modelu. Model uczy się, by następnym razem trafiać lepiej. Czyli cały pipeline sprowadza się do wymiany protokołów przez internetowe API:

- Zapytanie (daj rekomendacje)
- Odpowiedź (lista produktów + token)
- Raporty (wyświetlono, kliknięto, kupiono z tokenem)

PODSUMOWANIE

Google Cloud Vertex AI Recommendations to gotowy silnik, który sam uczy się i serwuje podpowiedzi produktowe. Cała praca sklepu sprowadza się do dwóch technicznych obowiązków. Najpierw raz opisuje się ofertę – każdy produkt trafia do pliku ND-JSON z identyfikatorem, nazwą, kategorią, ceną i ewentualnymi dodatkowymi cechami. Później, w regularnym (np. tygodniowym) rytmie trzeba wygenerować drugi strumień danych: prostą listę zdarzeń użytkowników zapisanych w tym samym formacie JSON. Każda linijka mówi, że konkretnego dnia i godzinie użytkownik X obejrzał stronę produktu, dodał go do koszyka albo sfinalizował zakup. Te dwa pliki wrzuca się do Cloud Storage lub BigQuery, a Vertex AI sam rozpoznaje, co jest katalogiem, co jest historią zachowań i trenuje model bez dalszej ingerencji.

Kiedy strona potrzebuje poleceń, serwer sklepu wysyła niewielkie zapytanie HTTP do interfejsu predict: „to ja, użytkownik 123, oglądam stronę główną – co mi polecisz?”. Chmura odsyła zgrabną listę identyfikatorów produktów posortowanych według trafności plus specjalny token, który trzeba odesłać z powrotem, gdy klient zobaczy listę, kliknie albo kupi. Ten token pozwala Google'owi zrozumieć, które sugestie okazały się skuteczne. System sam analizuje setki takich sygnałów i co pewien czas (najczęściej co kilka godzin, lecz w razie potrzeby wystarczy raz w tygodniu) odświeża własny model tak, by kolejne listy były celniejsze.

Z punktu widzenia zespołu e-commerce cała sztuczna inteligencja znajduje się w zamkniętym pudełku. Nie trzeba – i nie można – zmieniać architektury sieci ani dobierać hiperparametrów. Człowiek pełni rolę dostawcy dobrze sformatowanych plików i opiekuna prostego API. To podejście jest idealne, gdy liczy się szybkość wdrożenia i minimalny kod ML, ale nie pozwala na dogłębne eksperymenty ani na włączanie niestandardowych cech wprost do modelu. Wszystko, co dzieje się po wgraniu danych, pozostaje niewidoczne, a efekty przychodzą w postaci gotowych list produktów. Jeśli więc potrzebna jest czysta wygoda „wgraj pliki i zapomnij”, Vertex AI Recommendations spełni oczekiwania; jeśli natomiast celem jest pełna kontrola nad algorytmem i przestrzeń do badań, trzeba wybrać bardziej otwarte rozwiązanie, np. własny pipeline w Azure ML czy Amazon Personalize.

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.