IUM-Fijalkowski-Niewiarowski

Repository for realization of 2024Z IUM project

Cytat tematu:

"Gdybyśmy tylko wiedzieli, kiedy użytkownik będzie chciał przesłuchać bieżący utwór w całości, a kiedy go przewinie – moglibyśmy lepiej zorganizować nasz cache"

Analiza tematu

Definicja problemu biznesowego

Stworzenie modelu klasyfikującego czy użytkownik nie pominie danego utworu. Przyda się to w celu stwierdzenia potrzeby cache-owania konkretnego utworu. Czyli jeśli dla danego utworu dany użytkownik będzie go najprawdopodobniej pomijał, nie ma potrzeby cache-owania go. W innym przypadku w celu usprawnienia działania aplikacji dla użytkownika możemy dany utwór cache-ować.

Zdefiniowanie zadania modelowania

Zadanie

Zajmujemy się klasyfikacją binarną akcji użytkowników. Staramy się określić czy dany użytkownik pominie lub nie pominie dany utwór na podstawie jego preferencji i atrybutów danego utworu. ### Założenia - Użytkownicy mają w miarę stałe preferencje muzyczne (nie zmieniają ich co dwa tygodnie) - Analitycy dostarczają poprawne i pełne dane - Nie rozróżnimy typów pominięcia utworu w zależności od czasu przesłuchanego utworu, utwór pominięty w 1/2 lub w 1/4 jego trwania jest tak samo pominiętym utworem

Model bazowy

Opis

Jako model bazowy stworzyliśmy program który sprawdza czy typ danej piosenki jest wśród lubianych typów muzyki danego użytkownika i definiowaliśmy, że pominie ten utwór jeśli nie jest ### Wyniki Tablica pomyłek dla modelu bazowego:

```
[  255  1386]
[  102 3876]
```

Inne metryki: - Dokładność: 0.29 - Recall: 0.71 - Precyzja: 0.06

Kryteria sukcesu

Biznesowe

Celem jest przyspieszenie działania aplikacji poprzez cache-owanie tylko potrzebnych utworów, więc odpowiednim kryterium sukcesu będzie zbieranie metryk dotyczących na ile dokładne były nasze predykcje w środowisku produkcyjnym. Pozwoli to określić czy nasz model w realny sposób usprawnia działanie aplikacji. Ze względu na specyfikę zadania można zbierać te metryki w czasie rzeczywistym po czym prezentować je w zintegrowanych systemach (użyć w tym celu można np. prometeusza i grafany). ### Analityczna Model bazowy ma bardzo niską dokładność 29%, naszym zadniem będzie utworzenie modelu o większej dokładności

Model uczelnia maszynowego

Przygotowanie danych

Zostały połączone dane plików: sessions.jsonl, tracks.jsonl, users.jsonl. W celu wytworzenia jednego zbioru danych.

Informacje o sesjach zostały przefiltrowane następująco: - Nie interesują nas wiersze z akcjami innymi niż 'skip' i 'play' - Jeśli dla danej sesji i dla danego utworu została wykonana akcja 'skip' i 'play' oznacza to, że utwór został pominięty, tak więc zostawiamy tylko wiersz z operacją 'skip'

Na podstawie sesji zostały również wyliczone dwa dodatkowe atrybuty: - Średnio ile procent utworów użytkownik pomijał-pokazuje to na tendencję użytkownika do pomijania utworów - Średnio ile procent dany utwór był pomijany-pokazuje to jak często dany utwór jest pomijany

Ulubione gatunki użytkownika zostały czytane z pliku users.jsonl i zmienione do postaci binarnych atrybutów pokazujących czy użytkownik lubi dany gatunek muzyczny.

Podział danych

Dane zostały podzielone na dane uczące, dane walidacyjne oraz dane przeznaczone do przeprowadzenia testu AB, poprzez wybranie po 1000 danych o losowych indeksach i przyniesienie ich z danych uczących ich do danych walidujących oraz danych do testów AB.

Dane zostały następnie zapisane do folderu data/processed w postaci następujących plików csv: - merged_data.csv - dane uczące do modelu - y.csv - etykiety dla danych uczących - validation_data.csv - dane walidujące - validation_classes.csv - etykiety dla danych walidujących - ab_test_data.csv

Macierz korelacji cech

Macierz korelacji opracowana z wykorzytsaniem skryptu `corelation.py`

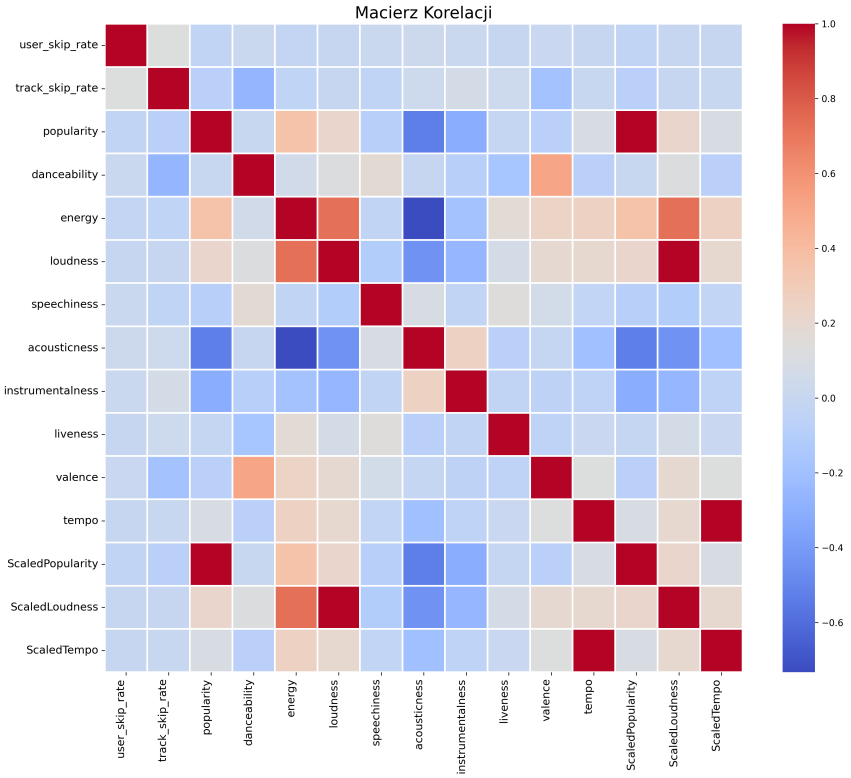


Figure 1: Opis alternatywny

Model

Został wykorzystany model LiniearSVC z domyślnymi parametrami. Pierwsza wersja modelu bez dodatkowych atrybutów opisanych powyżej osiągnęła następujące wyniki:

	precision	recall	f1-score	support
0	0.59	0.05	0.10	430
1	0.58	0.97	0.72	570
accuracy			0.58	1000
macro avg	0.58	0.51	0.41	1000
weighted avg	0.58	0.58	0.45	1000

Po zmienie balansu klas w modelu na 'balanced' otrzymaliśmy następujące wyniki:

	precision	recall	f1-score	support
0	0.45	0.51	0.48	430
1	0.59	0.52	0.55	570
accuracy			0.52	1000
macro avg	0.52	0.52	0.51	1000
weighted avg	0.53	0.52	0.52	1000

Po dodaniu dodatkowych atrybutów opisanych powyżej otrzymaliśmy następujące wyniki:

	precision	recall	f1-score	support
0	0.68	0.74	0.71	429
1	0.79	0.73	0.76	571
accuracy			0.74	1000
macro avg	0.73	0.74	0.73	1000
weighted avg	0.74	0.74	0.74	1000

Mikroserwis

Mikroserwis został zaimplementowany przy użyciu frameworka FastAPI, który umożliwia serwowanie predykcji dwóch modeli: naiwnego i docelowego. Serwis zawiera trzy główne endpointy:

1. /naive - obsługuje predykcje wykonywane przez model naiwny (BaseModel).
2. /recommend - zwraca predykcję wygenerowaną przez model docelowy (NormalModel).

3. /abtest - realizuje test A/B, losowo wybierając jeden z dwóch modeli (z równym prawdopodobieństwem) i zwracając jego predykcję. Informacje o wybranym modelu, użytkowniku, utworze i wyniku predykcji są zapisywane w pliku logów (model_usage_log.json). Dane wejściowe są przesyłane jako obiekt JSON, opisany przez model Pydantic (PredictionRequest). Każda predykcja jest przetwarzana przez odpowiedni model, a jej wynik (np. PLAY lub SKIP) jest zwracany klientowi i zapisywany do logów w celu późniejszej analizy. Logi są inicjalizowane przy pierwszym uruchomieniu aplikacji, jeśli plik logów nie istnieje.

Instrukcja uruchomienia

Zainstaluj wymagane zależności:

```
pip install -r requirements.txt
```

Dodaj aktualą wersję danych w lokalizacji:

```
./data/raw/v2
```

Lista wymaganych plików: - sessions.jsonl - tracks.jsonl - users.jsonl - artists.jsonl

Uruchom Mikroserwis używając uvicorn:

```
uvicorn api.app:app --host 0.0.0.0 --port 8000
```

Przykładowe zapytanie z poziomu PowerShell:

```
Invoke-RestMethod -Uri "http://127.0.0.1:8000/recommend" `
>> -Method POST `
>> -Headers @{ "Content-Type" = "application/json" } `
>> -Body '{"user_id": 101, "track_id": "31PzY79H10HCgJs533Xq6B"}'
```

Testy AB

Testy A/B w mikroserwisie są realizowane za pośrednictwem dedykowanego endpointu /abtest, który wybiera jeden z dwóch modeli (naiwny lub docelowy) z równym prawdopodobieństwem 50/50. Dla każdego żądania serwis:

Losowy wybór modelu: Decyduje, czy predykcja zostanie wygenerowana przez model naiwny (BaseModel), czy docelowy (NormalModel). Wykonanie predykcji: Wybrany model generuje wynik dla danego użytkownika (user_id) i utworu (track_id), wskazując, czy utwór zostanie odtworzony (PLAY) czy pominięty (SKIP). Logowanie wyników: Informacje o wybranym modelu, wejściowych danych użytkownika i wyniku predykcji są zapisywane w pliku logów (model_usage_log.json). Każdy wpis logu zawiera nazwę modelu, identyfikator użytkownika i utworu oraz wynik predykcji.

Aby ułatwić przeprowadzenie testów opracowany został skrypt wczytujący wcześniej przygotowane dane i wysyłający zapytanie do mikroserwisu.

Wyniki przeprowadzonego testu:

Dla obu modeli otrzymaliśmy następującą accuracy:

```
    model  accuracy
0   naive  0.432075
1  target  0.744681
```

W dobry sposób prezentuje to sporządzony wykres:

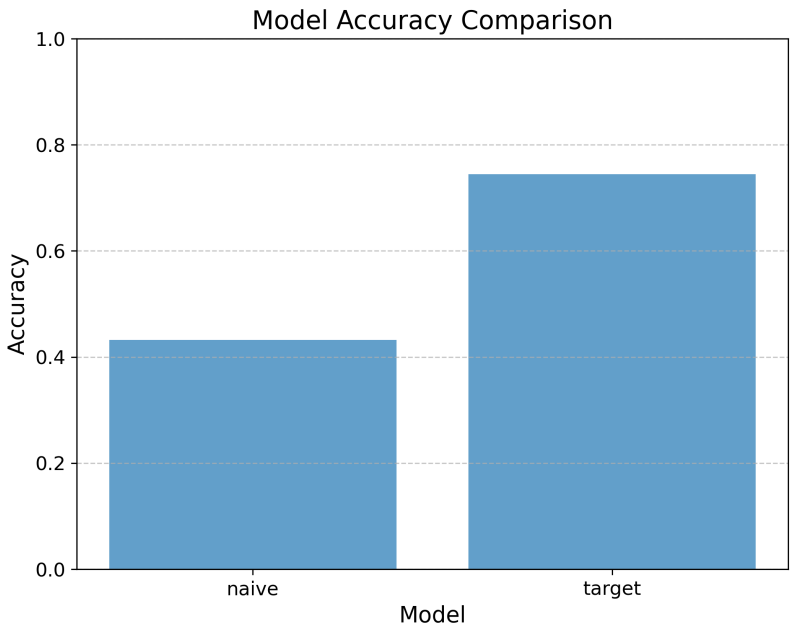


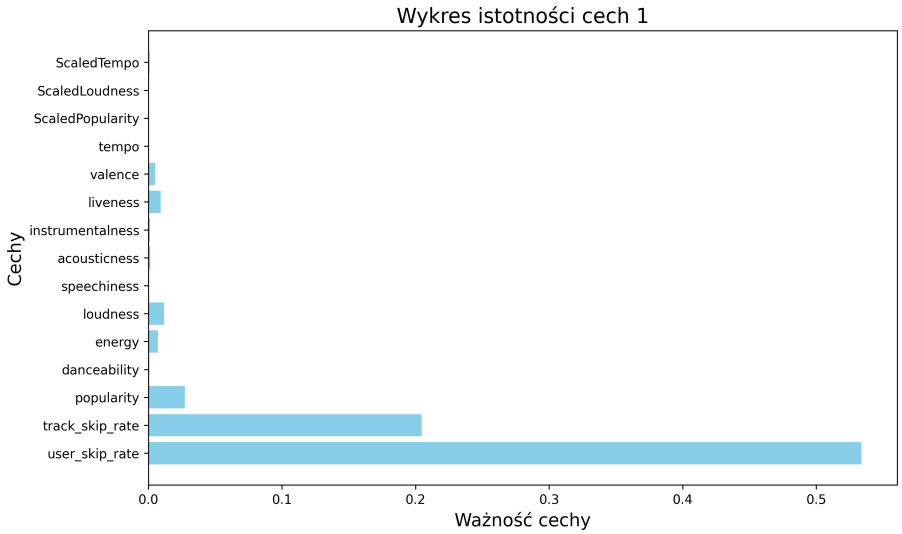
Figure 2: Opis alternatywny

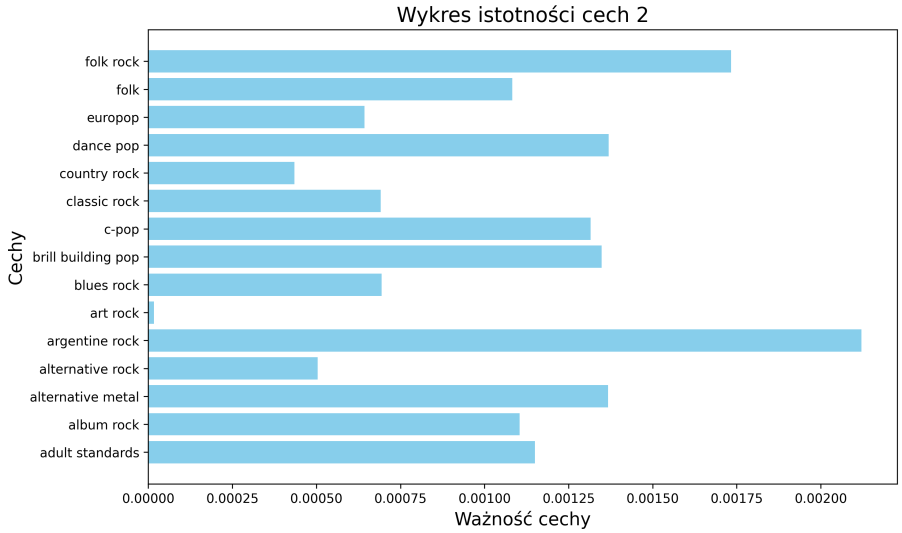
Wnioski na podstawie testów AB

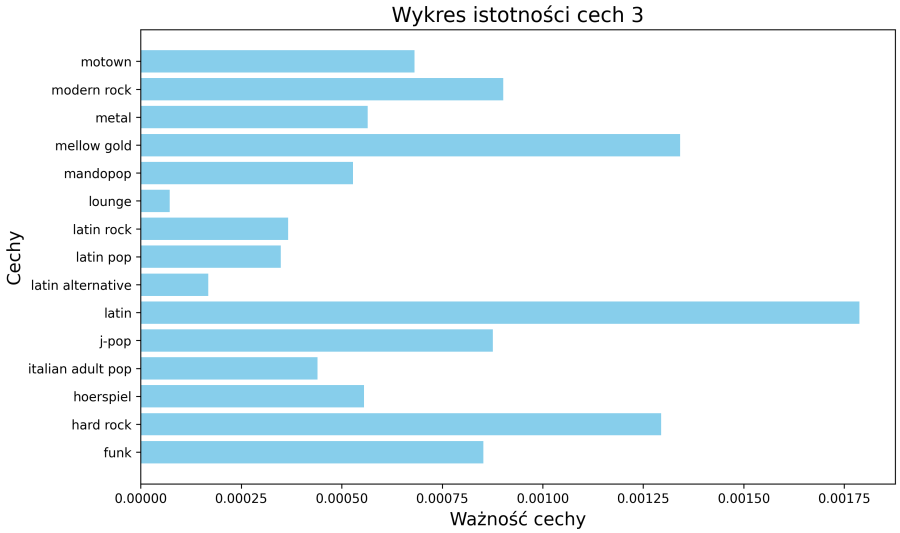
1. Uzyskane wyniki testu A/B wyraźnie wskazują, że model docelowy (target) osiąga znacznie lepszą skuteczność (accuracy: 74,47%) w porównaniu do modelu naiwnego (naive), którego dokładność wynosi jedynie 43,2%. Taka różnica sugeruje, że model docelowy lepiej przewiduje zachowania użytkowników, co czyni go bardziej wartościowym w praktycznym zastosowaniu, np. w systemach rekomendacyjnych.

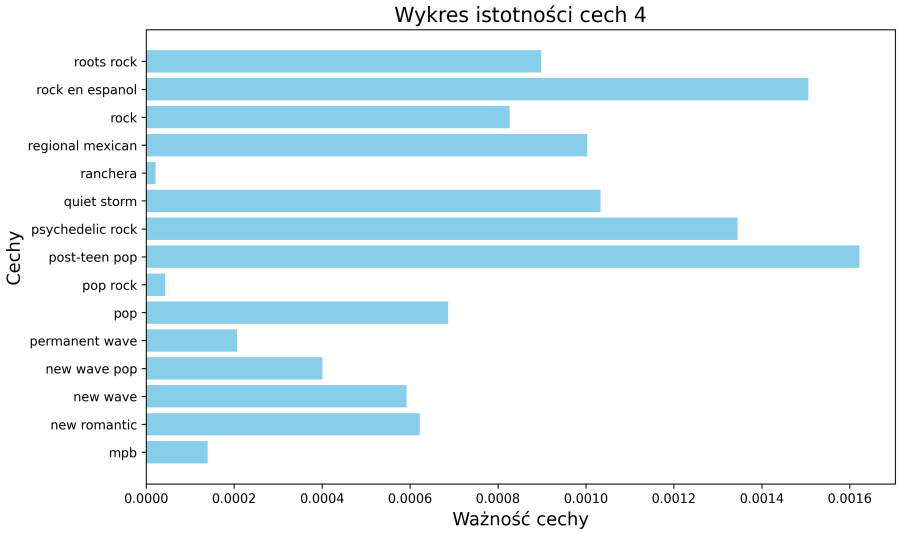
2. Sporządzony wykres nie tylko wizualizuje tę różnicę, ale także podkreśla istotność prowadzenia testów A/B jako narzędzia do podejmowania decyzji opartych na danych. Wyniki wskazują, że wdrożenie modelu docelowego może znacząco poprawić jakość rekomendacji i zadowolenie użytkowników.

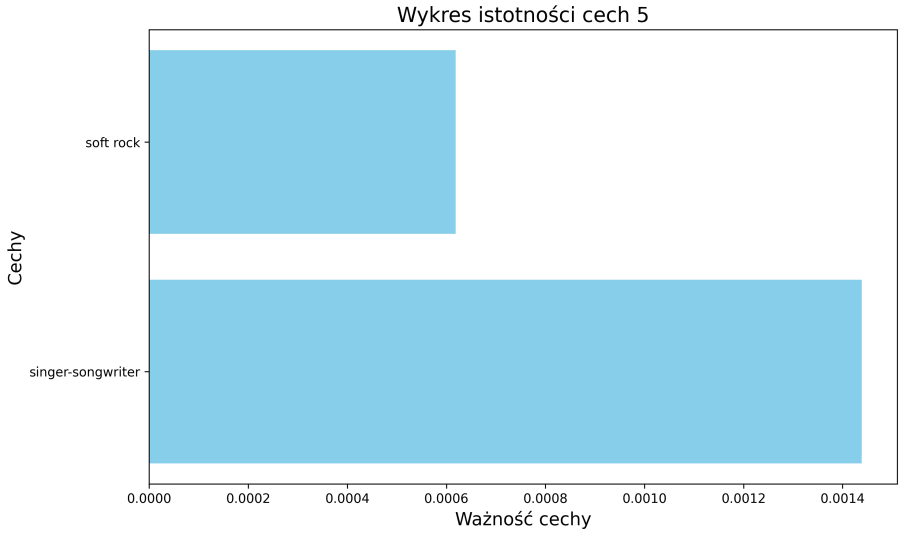
Istotność Cech:






Wykres istotności cech 3
motown
modern rock
metal
mellow gold
mandopop
lounge
latin rock
latin pop
latin alternative
latin
j-pop
italian adult pop
hoerspiel
hard rock
funk
Cechy
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
Ważność cechy


Wykres istotności cech 4
roots rock
rock en espanol
rock
regional mexican
ranchera
quiet storm
psychedelic rock
post-teen pop
pop rock
pop
permanent wave
new wave pop
new wave
new romantic
mpb
Cechy
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0.0016
Ważność cechy



Wnioski na podstawie wykresów istotności cech: 1. Analizując przedstawione wykresy można dostrzec że największy wpływ na działanie modelu mają cehcy takie jak: track_skip_rate, user_skip_rate oraz popularność danego utworu.

2. Gatunek muzyczny jaki dany utwór reprezentuje mają nieznaczny wpływ na predykcje modelu, jednak występują między poszczególnymi gatunkami istotne różnice.