

Projekt IUM

Antoni Grajek, Marcin Połosak

Styczeń 2025

1 Wprowadzenie

Celem projektu było opracowanie rozwiązań umożliwiających przewidywanie przyszłej popularności utworów muzycznych oraz stworzenie dedykowanego mikroserwisu, który umożliwia użytkownikom interakcje z tymi modelami predykcyjnymi. W ramach projektu zrealizowano dwa modele o różnym poziomie zaawansowania oraz mikroservis oparty na frameworku Flask.

2 Modele

2.1 Model bazowy

Model bazowy został zaprojektowany jako proste rozwiązanie przewidujące popularność utworów muzycznych na podstawie analizy historycznych danych o sesjach użytkowników. Działa on w oparciu o statystyczną analizę liczby odtworzeń utworów w ostatnim tygodniu przed wskazaną datą docelową (`target_date`). Model nie wykorzystuje metod uczenia maszynowego.

2.1.1 Algorytm działania

1. **Wczytywanie danych:** Załadowano dane utworów (`tracks.jsonl`) w celu stworzenia mapy `id` \rightarrow nazwa utworu. Dane sesji użytkowników (`sessions.jsonl`) przetwarzano partiami w celu efektywnego zarządzania pamięcią.
2. **Filtracja i agregacja:** Z danych sesji wybrano zdarzenia typu `play`, ograniczając się do okresu ostatniego tygodnia przed `target_date`. Następnie zliczono liczbę odtworzeń dla każdego `track_id`.
3. **Wybór najpopularniejszych utworów:** Na podstawie liczby odtworzeń wybrano 20 najpopularniejszych utworów, które następnie zamieniono na nazwy utworów przy użyciu wcześniej utworzonej mapy.

2.1.2 Ocena dokładności (opcjonalna)

Model umożliwia ocene trafności przewidywań poprzez porównanie wyników z rzeczywistą popularnością utworów w tygodniu następującym po **target_date**. Dokładność obliczana jest jako stosunek liczby trafionych utworów do liczby przewidywań.

2.1.3 Podsumowanie

Model bazowy oferuje szybkie i intuicyjne prognozy popularności utworów. Prostota algorytmu oraz brak konieczności stosowania uczenia maszynowego czynia go odpowiednim rozwiązaniem w sytuacjach ograniczonej dostępności danych historycznych lub mocy obliczeniowej.

3 Model zaawansowany

Model zaawansowany wykorzystuje szereg czasowy oraz algorytm wygładzania wykładniczego (*Exponential Smoothing*) w celu prognozowania przyszłej popularności utworów. Jest to bardziej zaawansowane podejście w porównaniu z modelem bazowym, pozwalające uwzględnić zmienność i trendy w danych historycznych.

3.1 Algorytm działania

1. **Wczytywanie danych:** Dane o sesjach użytkowników (`sessions.jsonl`) przetwarzane są partiami, a informacje o liczbie odtworzeń (*play counts*) dla każdego utworu agregowane w przedziałach dziennych. Dane utworów (`tracks.jsonl`) służą do mapowania identyfikatorów na nazwy utworów.
2. **Obliczanie szeregów czasowych:** Na podstawie danych sesji tworzony jest zbiór szeregów czasowych opisujących liczbę odtworzeń dla każdego utworu w poszczególnych dniach.
3. **Trenowanie modelu:** Każdy szereg czasowy o odpowiedniej długości (>2 dni) jest trenowany za pomocą algorytmu wygładzania wykładniczego z trendem addytywnym.
4. **Prognozowanie:** Dla wskazanej daty docelowej (`target.date`) model prognozuje liczbę odtworzeń dla każdego utworu na następny tydzień. Na tej podstawie wybiera 20 utworów z najwyższymi przewidywanymi przyrostami popularności.
5. **Ocena dokładności (opcjonalna):** Model pozwala ocenić trafność prognoz, porównując listę przewidywanych utworów z rzeczywistymi wynikami dla tygodnia następującego po dacie docelowej.

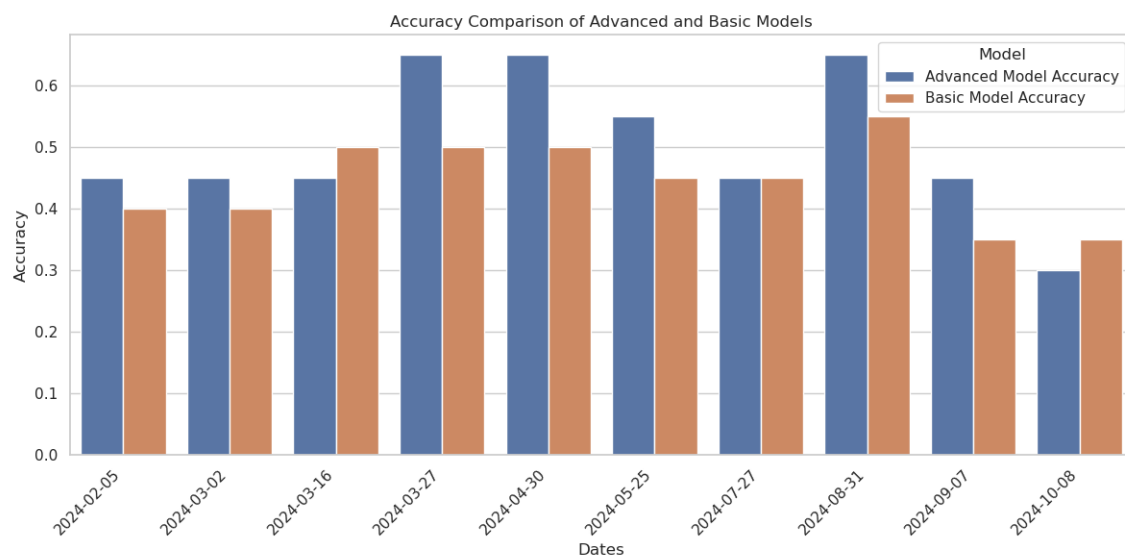
3.2 Zalety modelu

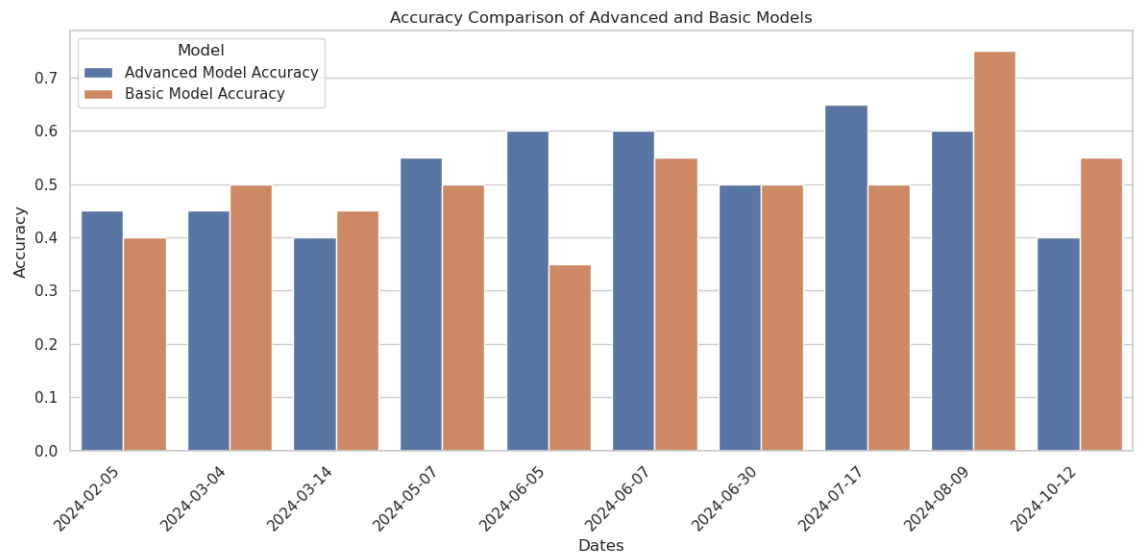
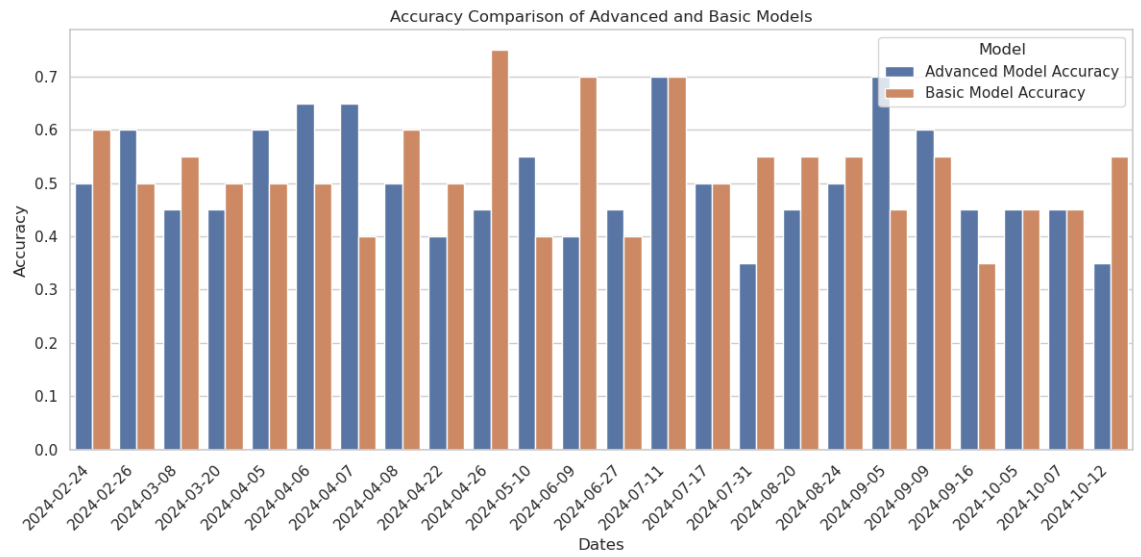
- Uwzględnia trendy i sezonowość w danych.
- Generuje dokładniejsze prognozy w porównaniu z modelem bazowym, szczególnie w przypadku utworów o regularnych wzorcach odtworzeń.
- Elastyczność dzięki wykorzystaniu statystycznych metod analizy szeregów czasowych.

4 Porównanie Modeli

Playlisty generowane przez oba modele nie spełniają kryteriów sukcesu. W obu przypadkach listy utworów nie spełniają tak jak zakładano 90 procentowej skuteczności. Porównanie dwóch modeli do generowania playlist wskazuje, że bardziej zaawansowany model (wykorzystujący prognozowanie szeregów czasowych) oferuje lekką poprawę dokładności przewidywań, jednak kosztem znacząco dłuższego czasu obliczeniowego, zarówno na etapie treningu, jak i predykcji. Z kolei podstawowy model, oparty na prostej logice i analizie popularności utworów w danym okresie, działa znacznie szybciej dzięki liniowej złożoności obliczeniowej $O(n)$, ale jego wyniki są tylko nieznacznie mniej precyzyjne.

Poniższe wykresy przedstawiają porównanie precyzji predykcji obu modeli, dla losowo dobranych dat:





4.1 Potencjalne przyczyny niepowodzenia

Model zaawansowany okazał się niewystarczający prawdopodobnie przez nieidealny wybór architektury. Problemem było również to, że metoda Exponential Smoothing wymaga dostatecznej ilości danych historycznych, co sprawia, że nowsze utwory mogły być niedoszacowane. Aby poprawić predkość predykcji, można ograniczyć liczbę utworów analizowanych przez model, skupiając się jedynie na tych najpopularniejszych, jednak to podejście może skutkować obcięciem z predykcji utworów o dużej zależności sezonowej.

5 Atrybuty użyte do budowy modelu zaawansowanego

W trakcie realizacji projektu skupiono się na dokładnym określeniu, które atrybuty dostępne w danych wejściowych będą najbardziej wartościowe dla uczenia modelu predykcyjnego. Proces ten obejmował szczegółową analizę dostępnych pól w pliku `sessions.jsonl`, mającą na celu ocenę ich przydatności w przewidywaniu trendów muzycznych.

Po wstępnej eksploracji danych zidentyfikowano, że pole `event type` zawiera kluczowe informacje o interakcjach użytkowników z utworami. Spośród różnych wartości tego pola, szczególnie istotna okazała się kategoria `play`, która wskazuje na odtworzenie utworów przez użytkowników. W związku z tym zdecydowano się na utworzenie szeregów czasowych poprzez sumowanie liczby odtworzeń dla poszczególnych utworów w określonych jednostkach czasu (np. dni, godziny).

Przeprowadzona analiza wzajemnej informacji między różnymi atrybutami a przyrostem popularności utworów wykazała, że inne dostępne pola, takie jak dane preferencji użytkowników czy metadane utworów, nie wносиły istotnych informacji, które mogłyby znacząco poprawić jakość modelu. Uwzględnienie tych danych mogłoby jedynie zwiększyć złożoność modelu bez proporcjonalnego wpływu na jego skuteczność.

Podsumowując, finalnie do uczenia modelu wybrano jedynie szeregi czasowe powstałe z sumowania pola `event type = play`. Dzięki takiemu podejściu możliwe było skupienie się na analizie szeregów czasowych, jednocześnie ograniczając złożoność obliczeniową modelu.

6 Podejście do modeli oraz ich strojenie

W celu wybrania najbardziej odpowiedniego modelu do przewidywania trendów muzycznych przeprowadzono analize dostępnych metod i przetestowano różne architektury. Wśród rozważanych modeli znalazły się między innymi Holt-Winters, LSTM, LightGBM, Gradient Boosting Regressor oraz modele ARIMA.

Podczas testowania napotkano jednak istotne trudności. Uczenie niektórych modeli, w szczególności LSTM oraz modeli opartych na drzewach boostowanych, wymagało znacznych zasobów obliczeniowych i czasu ze względu na dużą ilość danych w zbiorze treningowym. Dodatkowo, mimo licznych prób dostrajania hiperparametrów, uzyskiwane wyniki wskazywały jedynie na niewielkie zmiany w jakości predykcji modeli. To ograniczenie wpłynęło na dalszy wybór podejścia, kładąc większy nacisk na efektywność obliczeniową przy zachowaniu zadowalającej skuteczności modelu.

Do zzipowanego repozytorium zostana dołączone również modele, które nie zostały wybrane jako finalny model produkcyjny. Ich obecność ma na celu potwierdzenie przeprowadzonych iteracji oraz pokazanie szerokiego zakresu metod testowanych podczas realizacji projektu. Dzięki temu można zweryfikować, że proces wyboru modelu był kompleksowy i uwzględniał różne podejścia.

7 Mikroserwis predykcyjny

W ramach projektu opracowano mikroserwis oparty na frameworku Flask, który umożliwia prognozowanie popularności utworów muzycznych, korzystając z dwóch modeli: bazowego i zaawansowanego. Funkcjonalność mikroserwisu została podzielona na trzy kluczowe elementy:

- **Endpointy predykcyjne:** Mikroserwis udostępnia dwa endpointy:
 - `/predict/modelA` – obsługuje prognozy generowane przez model bazowy.
 - `/predict/modelB` – obsługuje prognozy generowane przez model zaawansowany.

Każdy z nich zwraca listę 20 najpopularniejszych utworów dla zadanej daty predykcji.

- **Test A/B:** Endpoint `/ABtest` umożliwia porównanie skuteczności obu modeli. Można wydzielić 3 zachowania serwisu.

Endpoint `/ABtest/begin` który służy do zainicjalizowania testu, który dla uzyskanej od użytkownika daty wyliczy predykcje modelu podstawowego i zaawansowanego.

Endpoint `/ABtest/getplaylist` zwraca użytkownikowi playlistę jednego z modeli z testu AB jednocześnie przypisując go do grupy testowej na podstawie odpowiedniego hashowania id użytkownika:

```
def hash_id(self, num_id):
    if int(hashlib.md5(str(num_id).encode()).hexdigest(), 16) % 2 == 0:
        return "A"
    else:
        return "B"
```

Endpoint `/ABtest/finish` kończy przeprowadzony test AB wyliczając jednocześnie rezultatem biznesowego testu AB. Jako kryterium jakości przyjęto średni wzrost odtworzeń utworów zaproponowanych w predykcji od daty stworzenia playlisty.

Plik generujący wynik AB testów:

```
BASE_DIR = os.path.abspath(os.path.join(os.path.dirname(__file__), ".."))

def calculate_count(users_list, date, track_list, file_path_tracks, file_path_sessions):
    .join(BASE_DIR, "dane", file_path_tracks)
    th.join(BASE_DIR, "dane", file_path_sessions)
    data_tracks = pd.read_json(file_path_tracks, lines=True)

    id_list = users_list
    count = 0
    track_id_list = []
    id_elem = 0
    for track in track_list:
        id_elem = data_tracks.loc[data_tracks["name"] == track, "id"]
        track_id_list.append(id_elem)

    for chunk in tqdm(
        pd.read_json(file_path_sessions, lines=True, chunksize=100000),
        desc="Przetwarzanie",
        unit="chunk",
    ):
        chunk["timestamp"] = pd.to_datetime(chunk["timestamp"])
        play_events = chunk[
            (chunk["event_type"] == "play")
            & (chunk["track_id"].isin(track_id_list))
            & (chunk["user_id"].isin(id_list))
            & (chunk["timestamp"] >= pd.to_datetime(date))
        ]
        count += play_events.shape[0]
    return count
```

Figure 1: Skrypt do AB testów

- **Obsługa logów:** Wszystkie operacje, w tym prognozy i testy, są rejestrowane w pliku logów, co pozwala na monitorowanie działania aplikacji i diagnostykę błędów.

Procedura testu AB w zapisie logów:

```
81 2025-01-24 19:52:00,150 - INFO - Test AB. Test rozpoczęty dla daty 2024-01-08.
82 2025-01-24 19:54:38,843 - INFO - Test AB. Playlist model A: ['bad guy', 'drivers license', 'Put Your Records On', 'Golden', 'La Tóxica', 'The Nights']
83 2025-01-24 19:54:38,843 - INFO - Test AB. Playlist model B: ['Maniac', 'My Head & My Heart', 'drivers license', 'Anyone', 'Dreams - 2004 Remaster', 'I
84 2025-01-24 19:55:23,366 - INFO - Test AB. User 101 added to group A
85 2025-01-24 19:55:27,512 - INFO - Test AB. User 128 added to group A
86 2025-01-24 19:55:30,384 - INFO - Test AB. User 129 added to group B
87 2025-01-24 19:55:38,096 - INFO - Test AB. User 134 added to group B
88 2025-01-24 19:55:43,723 - INFO - Test AB. Test zakończony.
89 2025-01-24 19:55:43,723 - INFO - Test AB. Model A users: ['101', '128']. Users count: 2.
90 2025-01-24 19:55:43,723 - INFO - Test AB. Model B users: ['129', '134']. Users count: 2.
```

Figure 2: Logi z przeprowadzonego testu AB

Mikroserwis działa na danych wejściowych w formacie JSON. Dzięki modularnej budowie może być łatwo rozwijany i integrowany z innymi aplikacjami. Implementacja uwzględnia także możliwość ponownego wczytywania wytrenowanego modelu zaawansowanego, co zwiększa elastyczność rozwiązania.

7.1 Endpointy predykcyjne

Zasada działania obu endpointów jest identyczna, w odpowiedzi na komendę zawierającą datę w której będzie przewidywana popularność system zwraca wynik przewidywań modelu.

Komenda:

```
marcin9047@LAPTOP-AVCRHESH:~$ curl -X POST -H "Content-Type: application/json" -d '{"input_data": "2024-05-24"}' --max-time 360 http://localhost:8060/predict/modelB
```

Odpowiedź:

```
{
  "input": {
    "input_data": "2024-01-08"
  },
  "model_used": "Model Zaawansowany",
  "prediction": [
    "Maniac",
    "My Head & My Heart",
    "drivers license",
    "Anyone",
    "Dreams - 2004 Remaster",
    "Believer",
    "The Business",
    "Up",
    "telepat\u00f3da",
    "In Your Eyes",
    "Streets",
    "Say You Won't Let Go",
    "Stressed Out",
    "Take You Dancing",
    "Follow You",
    "The Less I Know The Better",
    "Why'd You Only Call Me When You're High?",
    "Therefore I Am",
    "Cupid's Chokehold / Breakfast in America",
    "goosebumps"
  ]
}
```

7.2 Endpointy A/B test

Procedura przeprowadzenia testów opiera się na pierwotnej generacji playlist za pomocą obu modeli oraz analizie aktywności użytkowników związanej z ut-

worami w liście po predykcji. Oczekuje się, że utwory lepszego modelu powinny po zaproponowaniu utworów generować większą łączną ilość odtworzeń przez użytkowników do niego przypisanych.

7.3 Begin

W ramach tej komendy mikroserwis generuje środowisko do testów AB. Tworzy playlisty obu modeli dla zadanej przez użytkownika daty. Jest to rozpoczęcie testu AB.

Komenda generująca test:

```
marcin9047@LAPTOP-AVCRHESH: $ curl -X POST -H "Content-Type: application/json" -d '{"input_data": "2024-01-08"}' --max-time 360 http://localhost:8060/ABtest/begin [h!]
```

Figure 3: Komenda do rozpoczęcia testu

Rezultat:

```

{
  "Data testu": "2024-01-08",
  "Playlist model A": [
    "bad guy",
    "drivers license",
    "Put Your Records On",
    "Golden",
    "La T\u00f3xica",
    "The Nights",
    "positions",
    "BICHOTA",
    "Star Shopping",
    "Whoopty",
    "Heartbreak Anniversary",
    "Lucid Dreams",
    "The Box",
    "Film out",
    "Up",
    "Hayloft",
    "Electric Love",
    "Martin & Gina",
    "The Less I Know The Better",
    "Sweater Weather"
  ],
  "Playlist model B": [
    "Maniac",
    "My Head & My Heart",
    "drivers license",
    "Anyone",
    "Dreams - 2004 Remaster",
    "Believer",
    "The Business",
    "Up",
    "telepat\u00eda",
    "In Your Eyes",
    "Streets",
    "Say You Won't Let Go",
    "Stressed Out",
    "Take You Dancing",
    "Follow You",
    "The Less I Know The Better",
    "Why'd You Only Call Me When You're High?",
    "Therefore I Am",
    "Cupid's Chokehold / Breakfast in America",
    "goosebumps"
  ]
}

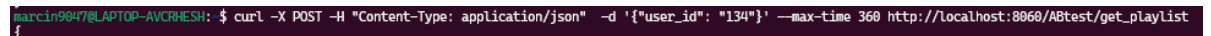
```

Figure 4: Rezultat rozpoczecia testu

7.4 Get playlist

Komenda ta w ramach testu AB przypisuje losowo użytkownika do jednej z 2 grup *A/B*. Dodaje go również do listy użytkowników danej grupy w pliku. Ostatecznie zwraca mu również wcześniej otrzymany przy rozpoczęciu wynik predykcji dla dobranej przez niego grupy. W celu poprawności działania użytkownik musi przekazać do systemu swoje id.

Komenda przypisująca do grupy:



```
marcin9847@LAPTOP-AVCRHESH: $ curl -X POST -H "Content-Type: application/json" -d '{"user_id": "134"}' --max-time 360 http://localhost:8060/ABtest/get_playlist
```

Figure 5: Komenda przypisująca do grupy

Rezultat:

```
{
  "Picked group": "B.Model Zaawansowany",
  "Playlist": [
    "Maniac",
    "My Head & My Heart",
    "drivers license",
    "Anyone",
    "Dreams - 2004 Remaster",
    "Believer",
    "The Business",
    "Up",
    "telepat\u00feda",
    "In Your Eyes",
    "Streets",
    "Say You Won't Let Go",
    "Stressed Out",
    "Take You Dancing",
    "Follow You",
    "The Less I Know The Better",
    "Why'd You Only Call Me When You're High?",
    "Therefore I Am",
    "Cupid's Chokehold / Breakfast in America",
    "goosebumps"
  ]
}
```

Figure 6: Rezultat przypisania do grupy

7.5 Finish

Komenda ta odpowiedzialna jest za zakończenie testu. Przesyła ona ocene uzyskanych wyników umożliwiającą porównanie metod, oraz czyści baze predykcji i użytkowników dla testu AB.

Komenda kończąca test: Z racji dużej objętości zbioru danych całej sesji aplikacji

```
marcin9047@LAPTOP-AVCRHESH:~$ curl -X POST -H "Content-Type: application/json" --max-time 360 http://localhost:8060/ABtest/finish
```

Figure 7: Zakończenie testu - komenda

nie został wyliczony końcowy rezultat. Poprawność działania sprawdzono tylko dla mniejszych zbiorów. Cała ocena jakości sprowadza się do wzoru:

Ocena(Model)=suma odtworzeń utworów z playlisty przez użytkowników danej grupy po dokonaniu predykcji / liczba użytkowników grupy.