Politechnika Śląska

Dokumentacja realizowanego projektu

2021/2022

Systemy Sztucznej Inteligencji

Spotify Song Finder

Kierunek: Informatyka
Wydział Matematyki Stosowanej

Członkowie zespołu:
Artur Iwański
Jakub Chrobok
Bartosz Jarzyński

Gliwice, 2021/2022

Część I

Opis programu

Na stronie *kaggle.com* znaleźliśmy interesującą nas bazę danych, czyli *Spotify - All Time Top 2000s Mega Dataset*. Przestawia ona zestaw blisko 2000 najlepszych piosenek z lat 1956 - 2019. Każda z zestawionych piosenek posiada 15 cech, kolumn opisujące daną piosenkę. Cechy opisujące piosenki to:

- Indeks,
- Tytuł,
- Artysta,
- Styl muzyki,
- Długość piosenki - wyrażana w sekundach,
- Rok wypuszczenia piosenki,
- Tempo piosenki (BPM) - wyrażona w uderzeniach na minutę,
- Akustyczność - większa wartość, to piosenka jest bardziej akustyczna,
- Żywotność - jeśli piosenka brzmi jak nagrana z koncertu na żywo, tym większa wartość parametru,
- Głośność (dB) - im głośniejsza piosenka, tym większa wartość parametru,
- Energiczność - im większa wartość, tym piosenka jest bardziej energiczna,
- Taneczność - im większa wartość, tym łatwiej zatańczyć do danej piosenki,
- Pozytywność - im większa wartość, tym bardziej pozytywna wydaje się piosenka,
- Mowa - im więcej słów wypowiedzianych w piosence, tym większa wartość parametru,
- Popularność - im piosenka jest bardziej popularna, tym większa wartość.

Użytkownik ośmiu cechom, takich jak *Energiczność, Taneczność, Głośność, Żywotność, Pozytywność, Akustyczność, Mowa, Popularność* przypisuje wartośc od 0 do 100 w celu klasyfikacji, jakie parametry piosenki pasują do poszukiwanej. Algorytm dobiera w oparciu o wpisane dane najbardziej pasującą piosenkę pod względem statystyk.

Program po obliczeniach wyświetla *ID* oraz *Tytuł* najlepiej dopasowanych piosenek.

```
userchoice = [40,70,20,74,80,11,44,90]
```

Rysunek 1: Linijka kodu odpowiedzialna za ustalenie preferencji użytkownika.

Instrukcja obsługi

Program uruchamia się w środowisku programistycznym, na przykład w *Visual Studio Code*. Należy znaleźć odpowiednią rubrykę *userchoice*, która jest odpowiedzialna za ustalenie preferencji użytkownika. Dane to liczby z przedziału 0 do 100, gdzie im większa wartość, tym bardziej użytkownik chciałby przydzielić danemu atrybutowi większą wagę.

Część II

Opis działania

W danym projekcie między innymi używamy klasteryzacji, normalizacji oraz odległości Mińkowskiego.

- Klasteryzacja polega na przypisywaniu podobnych instancji do tej samej klasy. Dane do zadania klasyfikacji pochodzą z nieetykietowanych zbiorów uczących, gdzie brakuje podanych wprost klas dla instancji.
- Normalizacja zmiennych wejściowych, polega na takim przeskalowaniu wartości każdej zmiennej, aby sprowadzić te wartości do określonego zakresu (przedziału). Głównym celem tej operacji jest nadanie każdej zmiennej wejściowej jednakowego znaczenia (wagi) w stosunku do innych zmiennych.
- Odległość Minkowskiego – uogólniona miara odległości między punktami przestrzeni euklidesowej; niekiedy nazywa się także odległością Lm

Algorytm i implementacja

Z bazy danych pobieramy wartośći z każdej z kolumn, gdzie na początku normalizujemy zbiór poprzez *Klasteryzację*. Następnie dzięki danym podanym przez użytkownika algorytm wyznacza odległości pomiędzy wartościami podanymi a jawnymi, w oparciu o *Odległość Minkowskiego*. Przetworzenie tych danych pozwala nam na koniec wydrukować wszystkie pasujące do klasyfikatora wyniki.

Algorithm 1 Algorytm normalizujący wyniki w bazie danych Spotify.

Data: Dane wejściowe *kolumny*
for *Każda kolumna* **do**
  if *kolumna* == *'Loudness'* **then**
    $Tymczasowa := Baza[kolumna].min()$
    $Baza[kolumna] := 100 * Baza[kolumna]/Tymczasowa$
  end
  $Tymczasowa := Baza[kolumna].max()$
  $Baza[kolumna] := 100 * Baza[kolumna]/Tymczasowa$
end
Result: Znormalizowane wyniki

Algorithm 2 Algorytm tworzenia dystansów pomiędzy wartościami dot. przydzielenia piosenek.

Data: Dane wejściowe *Baza*, *WyboryUytkownika*, *Kolumny*, *Dystanse*
$WyboryUytkownika := [Danewpisaneprzezuytkownika]$
for *Długość bazy* **do**
  Acc jest tablicą
  for *Długość wyboru użytkownika* **do**
    Dodaj do *Acc* dystans Listy Minkowskiego.
  end
  Tymczasowe := Dystans Minkowskiego pomiędzy dwoma punktami
  Dodaj *Tymczasowe* do *Dystanse*
end
Result: Lista dystansów

Testy

Przedstawienie działania programu w oparciu o trzy testy.

```
userchoice = [40,70,20,74,80,11,44,90]
```

(a) Wybory użytkownika.

```
361                                        Feel Good Inc.
1900    Sgt. Pepper's Lonely Hearts Club Band - Remast...
968     No Woman, No Cry - Live At The Lyceum, London/...
Name: Title, dtype: object
```

(b) Wyniki, dopasowane piosenki.

Rysunek 2: Test nr 1.

```
userchoice = [20,10,90,70,50,15,90,25]
```

(a) Wybory użytkownika.

```
1373    Fire - Live at the Winterland, San Francisco, ...
920                              La Grange - 2005 Remaster
523                                              Formidable
Name: Title, dtype: object
```

(b) Wyniki, dopasowane piosenki.

Rysunek 3: Test nr 2.

```
userchoice = [87,50,51,12,17,88,66,42]
```

(a) Wybory użytkownika.

```
1935    Voodoo Child (Slight Return)
1462          Sympathy For The Devil
19                      Cry Me a River
Name: Title, dtype: object
```

(b) Wyniki, dopasowane piosenki.

Rysunek 4: Test nr 3.

Pełen kod aplikacji

```
# Import libraries

import pandas as pd
import numpy as np

# Import the Spotify CSV

spotify = pd.read_csv("Spotify-2000.csv")


# Display columns with their max values
columns = spotify.columns[6::]

for col in columns:
    name = "max" + str(col) + " = "
    temp = spotify[col].max()

df = spotify.copy()
del df["Length (Duration)"]
columns=df.columns[6::]


# Normalizing each result - values between 1 and 100
for col in columns:
```

```
25      if(col == "Loudness (dB)"):
26          temp = spotify[col].min()
27          spotify[col] = spotify[col]/temp * 100
28          continue
29
30      temp = spotify[col].max()
31      spotify[col] = spotify[col]/temp * 100
32
33
34  # Choosing right song for user
35  # Compare each user's value with songs' values
36  userchoice = [40,70,20,74,80,11,44,90]
37
38  df = spotify[
39                ["Energy",
40                 "Danceability",
41                 "Loudness (dB)",
42                 "Liveness",
43                 "Valence",
44                 "Acousticness",
45                 "Speechiness",
46                 "Popularity"]
47                ]
48
49  distancelist=[]
50  for i in range(len(df)):
51      acc = []
52
53      for j in range(len(userchoice)):
54          acc.append(abs(userchoice[j] - df.iat[i,j])**len(userchoice))
55      # Acc is a list of distances between userchoice and each song
56      # Acc root of distance list (Minkowski) - distance from picked point
            to user's choice
57
58      temp = sum(acc)**(1/len(userchoice))
59      distancelist.append(temp)
60
61  spotify["distance"] = distancelist
62
63
64  # The best songs are the ones with the lowest distance, are closest to
      the selected by the user
65  listofbestsongs=spotify.nsmallest(3,["distance"])
66
67  print(listofbestsongs["Title"])
```