DUMITRU RADU-CALIN

SISTEM DE RECOMANDARE DE MUZICA UTILIZAND SPOTIFY API SI MACHINE LEARNING

INTRODUCEREA TEMEI:

In zilele noastre, muzica se aude la orice colt de lume. Fie ca este vorba de spatiile noastre personale sau publice, ea ne influenteaza foarte mult starea in timpul activitatilor de zi cu zi. Si nu cred ca exista o aplicatie care sa produca un asemenea impact, in ceea ce priveste starea noastra personala asa cum procedeaza Spotify

Spotify reprezinta o aplicatie de streaming audio prin care i se permite utilizatorului nu doar sa-si asculte piesele preferate pe banda rulanta, dintr-o selectie fascinanta de genuri si artisti, dar de aseamana ii poate permite sa-si compune propriile playlisturi. Ceea ce apreciez foarte mult la aceasta aplicatie este faptul ca acesta poate sa furnizeze playlisturi avand diferite tipuri de clasificari, de la clasificarile standard precum cele ce tin de genul muzical sau de perioada in care a aparut melodia, la clasificari mai fine, mai subiective, care exploreaza mai mult starea pe care vrea sa ti-l ofere pe care o asculti. Si studiind despre aceasta problema, am constatat faptul ca, la fel ca orice aplicatie moderna, sunt utilizati algoritmi de Machine Learning cu scopul de a contura preferintele utilizatorilor si de a prezice melodii care sa se potriveasca stilului sau starii de ceea utilizatorul asculta.

O alta functie pe care am studiat-o in alcatuirea proiectului a fost si ceea de playlisturi radio, prin care pornind de la o melodie a unui artist sau de la artistul in sine, Spotify adduce alte piese similare cu cea sus mentionata, precum si alte melodii din repertoriul sau. Asa ca intentia mea pentru acest proiect a fost sa creez un sistem de recomandari de piese pe Spotify care sa imite aceste functionalitati

ETAPELE PROIECTULUI

1. PRELUCRAREA SI APROVIZIONAREA CU RESURSE

Pentru a incepe proiectul, a trebuit sa stochez un numar semnificativ de melodii provenite din playlisturi si care sa incorporeze cat mai multe genuri cu putinta. Initial m-am gandit sa apelez la un set gata facut de pe Kaggle, care avea cam tot ce trebuie in materie de proprietati si de numar de piese(vazusem la un moment ca erau stocate undeva spre ordinal milioanelor). Din nefericire, cand am intrat n-am gasit setul de date propus, asa ca a trebuit sa recurg la scrierea in cod a unor functii care sa-mi poata permita extragerea melodiilor de pe

DUMITRU RADU-CALIN

Spotify. Unealta care m-a ajutat la realizarea acestei operatiuni a fost libraria spotipy prin care se face legatura cu API-ul furnizat de cei de la Spotify. Am creat astfel urmatorul obiect care imi permite interactiunea cu acel API :

```
class PLAY:
    def __init__(self, client_id = secret_data.CLIENT_ID, client_secret=secret_data.CLIENT_SECRET):
        '''
        Prin functia de initializare a datelor se face si autentificarea catre Spotify API
        '''

        self.client_id = client_id
        self.client_secret = client_secret

        token = util.spotipy.oauth2.SpotifyClientCredentials(client_id=self.client_id,
                                                             client_secret=self.client_secret)
        cache_token = token.get_access_token(as_dict=False)
        self.sp = spotipy.Spotify(cache_token)


    def get_playlist_tracks(self, sp):
        '''
            Functie care preia rand pe rand toate piesele ce alcatuiesc un playlist curent
        '''
        results = self.sp.user_playlist_tracks(self.creator, self.playlist_id)
        tracks = results['items']
        while results['next']:
            results = sp.next(results)
            tracks.extend(results['items'])
        return tracks
```

Prin metodele de mai sus am reusit sa ma autentific la interfata de dezvoltare(este necesar sa ai un cont pe Spotify si sa ai o aplicatie deja creeata pentru a a avea accesul la prelucrarea datelor) si de asemenea sa preiau pana la peste 15000 de piese(Spotify impune o limita de pana la maximum 10000 de piese pe playlist asa ca a trebuit sa fac cu randul). Toate datele preluate au fost stocate in 2 2 csv-uri: unul cu datele meta si celalalt cu proprietatile uzuale ale melodiilor(de ex.instrumentalness, speechness, danceability, etc).

```
def scatter_tracks(self, songs):
    playlist_features_list = ["artist", "album", "track_name", "release_date", "track_id", "acousticness"
                              "danceability", "energy",
                              "key", "loudness", "mode", "speechiness",
                              "instrumentalness", "liveness", "valence", "tempo", "duration_ms",
                              "time_signature"]
    playlist_features = {}
    playlist_df = pd.DataFrame(columns=playlist_features_list)
    for track in songs:
        # Mai intai incepem sa preluam informatiile meta
        playlist_features["artist"] = track["track"]["album"]["artists"][0]["name"]
        playlist_features["album"] = track["track"]["album"]["name"]
        playlist_features["track_name"] = track["track"]["name"]
        playlist_features["release_date"] = track["track"]["album"]["release_date"]
        playlist_features["track_id"] = track["track"]["id"]

        # Preluam proprietatile importante ale melodiilor
        audio_features = self.sp.audio_features(playlist_features["track_id"])[0]
        for feature in playlist_features_list[5:]:
            playlist_features[feature] = audio_features[feature]

            # Concatenarea tablourilor

        track_df = pd.DataFrame(playlist_features, index=[0])
        playlist_df = pd.concat([playlist_df, track_df], ignore_index=True)

    playlist_df["release_date"] = pd.to_datetime(playlist_df["release_date"])
    playlist_df["year"] = pd.DatetimeIndex(playlist_df["release_date"]).year
    playlist_df = playlist_df.drop(columns=['release_date'])
    return  playlist_df
```

2. ANTRENAREA SETULUI DE DATE

Acum ca avem setul de date pregatit, m-am gandit la metoda pe care s-o pot folosi pentru a creea un model cat mai viabilin vederea formarii playlisturilor cat mai clare. Am folosit metode prin care am determinat numarul optim de grupuri de piese, avand in vedere faptul ca nu dispuneam de un set foarte mare de date pentru a antrena datele cu ajutorul retelelor neurale. Prima metoda folosita a fost calcularea PCA-ului(analiza de componenta principala). Am facut transformarea urmatoarelor coloane:

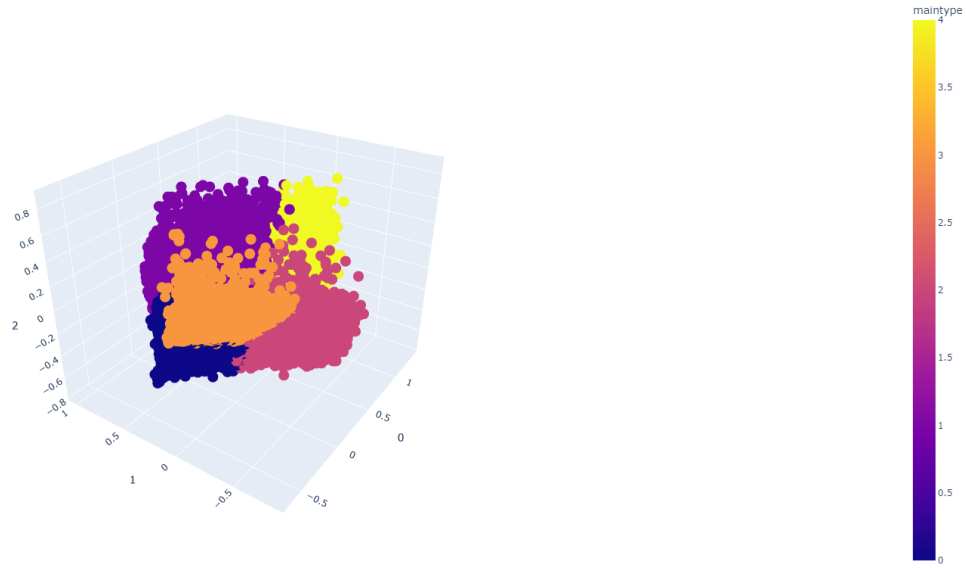
	danceability	energy	loudness	speechiness	instrumentalness	liveness	valence	acousticness
0	0.531	0.409	-7.574	0.0269	0.185000	0.1110	0.392	0.832000
1	0.695	0.261	-7.062	0.2540	0.917000	0.1310	0.424	0.409000
2	0.563	0.599	-8.718	0.0426	0.317000	0.1480	0.153	0.375000
3	0.178	0.884	-4.936	0.0602	0.015300	0.4390	0.184	0.000012
4	0.620	0.625	-7.438	0.5530	0.000000	0.3140	0.665	0.287000
...	...	...	...	...	...	...	...	...
15814	0.446	0.834	-4.014	0.0341	0.000000	0.1140	0.363	0.079400
15815	0.673	0.143	-18.598	0.0561	0.891000	0.0743	0.181	0.686000
15816	0.535	0.680	-7.093	0.0361	0.000000	0.1110	0.332	0.008410
15817	0.604	0.851	-4.780	0.0274	0.000366	0.0511	0.649	0.006160
15818	0.551	0.881	-6.099	0.0542	0.079100	0.1520	0.387	0.186000

-fiecare rand reprezinta proprietatile de masurat pentru fiecare melodie stocata

	danceability	energy	loudness	speechiness	instrumentalness	liveness	valence	acousticness
0	0.541837	0.409243	0.813112	0.028021	0.187247	0.112576	0.395960	0.835341
1	0.709184	0.261053	0.824433	0.264583	0.928138	0.132860	0.428283	0.410643
2	0.574490	0.599487	0.787816	0.044375	0.320850	0.150101	0.154545	0.376506
3	0.181633	0.884852	0.871443	0.062708	0.015486	0.445233	0.185859	0.000012
4	0.632653	0.625520	0.816119	0.576042	0.000000	0.318458	0.671717	0.288153
...	...	...	...	...	...	...	...	...
15814	0.455102	0.834788	0.891830	0.035521	0.000000	0.115619	0.366667	0.079719
15815	0.686735	0.142902	0.569353	0.058438	0.901822	0.075355	0.182828	0.688755
15816	0.545918	0.680591	0.823748	0.037604	0.000000	0.112576	0.335354	0.008444
15817	0.616327	0.851810	0.874892	0.028542	0.000370	0.051826	0.655556	0.006185
15818	0.562245	0.881849	0.845727	0.056458	0.080061	0.154158	0.390909	0.186747

-aici am aplicat transformare prin scalare folosind scalarea in minim si maxim(scalarea fiecarei proprietati in limitele date)

Cu ajutorul rezultatelor din PCA a rezultat urmatorul grafic 3D, unde am luat in considerare numarul de grupuri de 5 calculat folosind Kmeans:



Am considerat ca numarul de grupuri, conform graficului, se potriveste cu ceea ce imi doream.

Cele 5 grupuri reprezinta categoria principala a pieselor. Problema este ca sunt pana la 3000 de piese in unele grupuri, asa ca a trebuit sa alimentezi setul de date folosit('muzica') cu alte coloane precum tempo, durata cantecului si anul aparitiei si am reapelat Kmeans. Urmatoarea clasare a fost determinate prin metoda Silhouette(desi graficul imi dadea puncta mai joase, am ales numarul care determina distantele cele mai mici). Am constatat faptul ca va fii necesara o impartire in 200 de categorii mijlocii. Ultima categorie consta scalare standard a tuturor datelor numerice colectate, si impartirea in 280 de piese cu scopul de a putea avea cel putin 50 de piese per playlist.

3. CREEAREA DE PLAYLISTURI

Avand la indemana toate transformarile si grupele create, a urmat partea cea mai interesanta a acestui proiect si anume crearea playlistului. Apeland inca o data la conexiunea cu API-ul celor de la Spotify, am creeat urmatoarele metode prin care sa-mi pot crea playlisturile asa cum mi-am dorit. Mai intai am testat metodele prin care sa pot creea playlisturi folosind piese deja existente in setul de date. Asa ca luand la intamplare cel putin o piesa, am reusit sa creez un playlist de piese care sa apartina unei clase mai mici si atunci cand sunt

luate toate piesele din clasa repsectiva, sa preia piesele de la cea mai apropiata clasa fata de ultima piesa selectata.

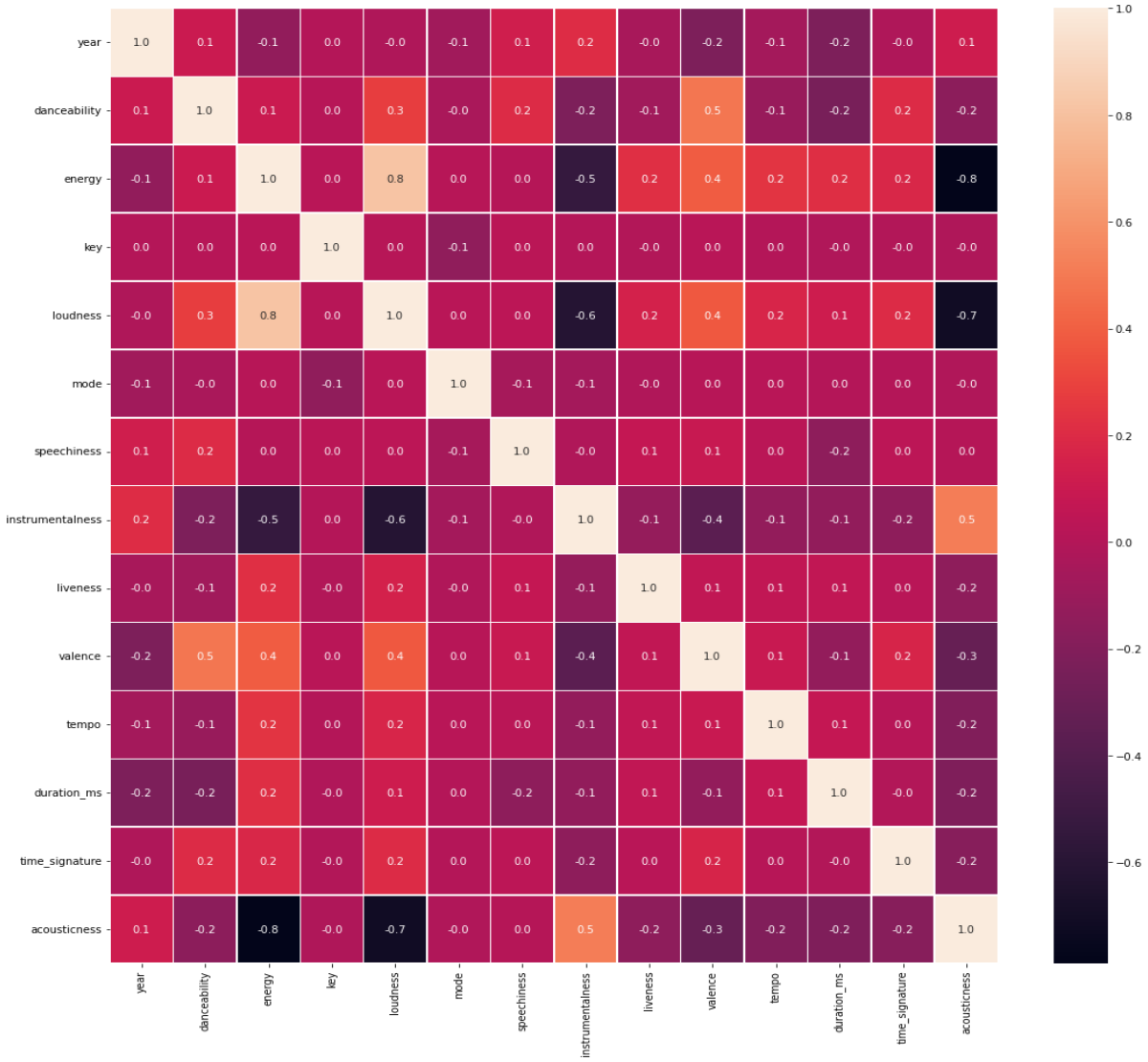
O alta de idee de creeare de playlisturi la care m-am gandit intre timp tine cont maim ult de distantele dintre melodii. Astfel, fiecare melodie are, datorita calculelor facute in sectiunea precedenta, inca 2 valori: cea mai scurta distanta de la punct si indexul vecinului celui mai apropiat. Playlisturile create vor avea cate o limita stabilita de numar de piese. De asemenea, am luat in considerare si ideea de a folosi muzica salvata de utilizator ca set de date pentru crearea acestor tipuri de liste.

Crearea propriu-zisa de playlisturi se refera la preluarea variabilelor returnate din apelul functiilor noastre si convertirea acestora in playlisturi Spotify folosind API-ul.

4. ALTE OBSERVATII

In afara crearii de liste, am mai facut si niste analize pe baza relatiilor dintre proprietatile unei melodii.

Aceasta constatare s-a facut folosind un heatmap:



Conform diagramei de mai sus vedem faptul ca sunt influente mult mai puternice in ceea ce privesc relatiile dintre zgomot(loudness) si energie – scor 0.8

- Valenta(valence) si dansabilitate(danceability) – 0.5
- Acusticitate(acousticness) si instrumentalitate(instrumentalness) – 0.5

De asemenea, se remarca faptul ca acusticitatea nu determina deloc nivelul de energie transmis intr-o piesa si se opune zgomotului. Acelasi lucru putem spune si despre nivelul de muzicitate(instrumentalness) .

CONCLUZII

Toate metodele explicate si implementate pentru acest proiect se concentreaza asupra melodiilor pe care le-ar asculta un utlizator. Iar in viitor, mi-ar placea sa experimentez mai mult cu aceste proprietati cu conditia de a incerca si o comparare a playlisturilor altor utlizatori. Am in plan crearea unor metode care sa permita crearea unei liste de recomandari in cazul pieselor preferate oferite de mai multi utilizatori.