Spotify recommendation engine

Capstone project

Final presentation (Akash Choudhari)

Project Goal

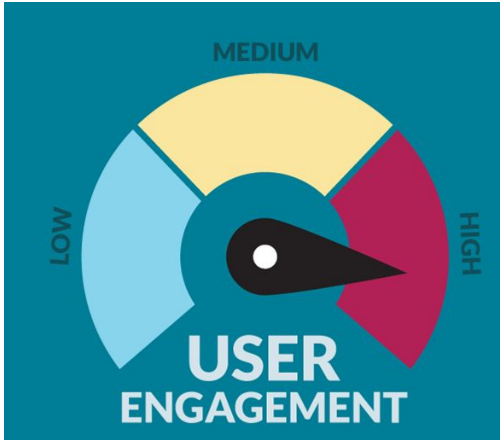
- Create a content-based filtering Spotify Song Recommendation System
- Understand how Spotify understands 'popularity'.
- Go deeper on "Recommended (based on what's in your playlist)" on your Spotify works?
- Run through process of building machine learning pipeline to create a playlist recommendation

Executive summary

Spotify is a platform that makes money through end users via subscriptions.

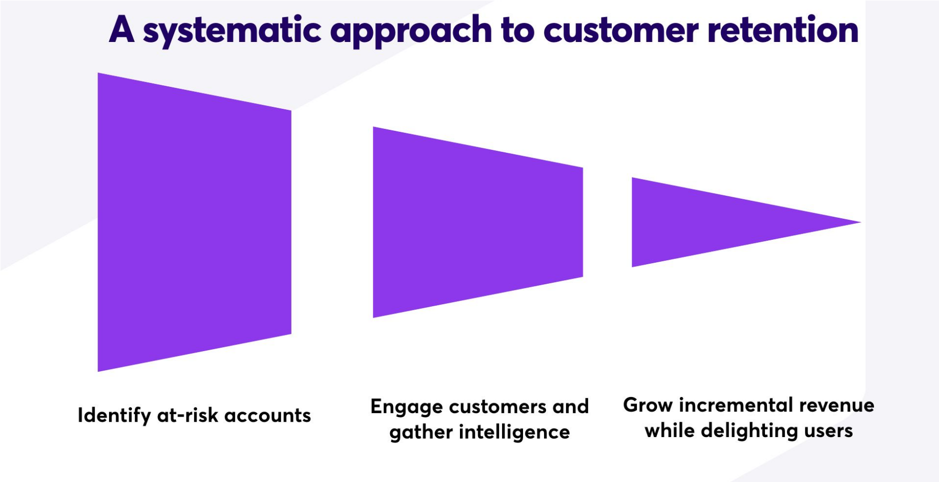
Spotify song recommendation system will help user discover engaging content to increase DAU (daily active user) metrics.

Use machine learning to filter out the content and present engaging content to the user for increasing their product usage.



Rationale

If users are unable to discover good content, they will move to other competitive platforms for finding music. Good recommendations will help with user retention and in-turn higher revenues for the company.



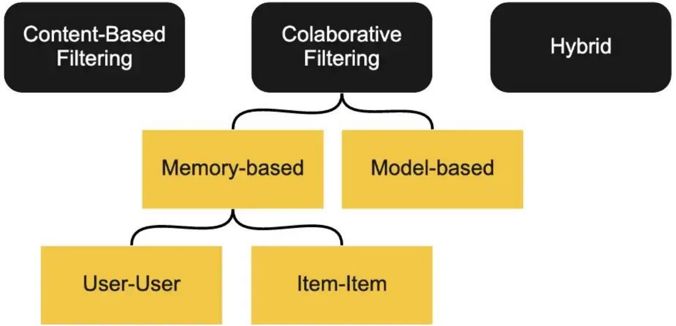
Recommendation system types

- Collaborative filtering

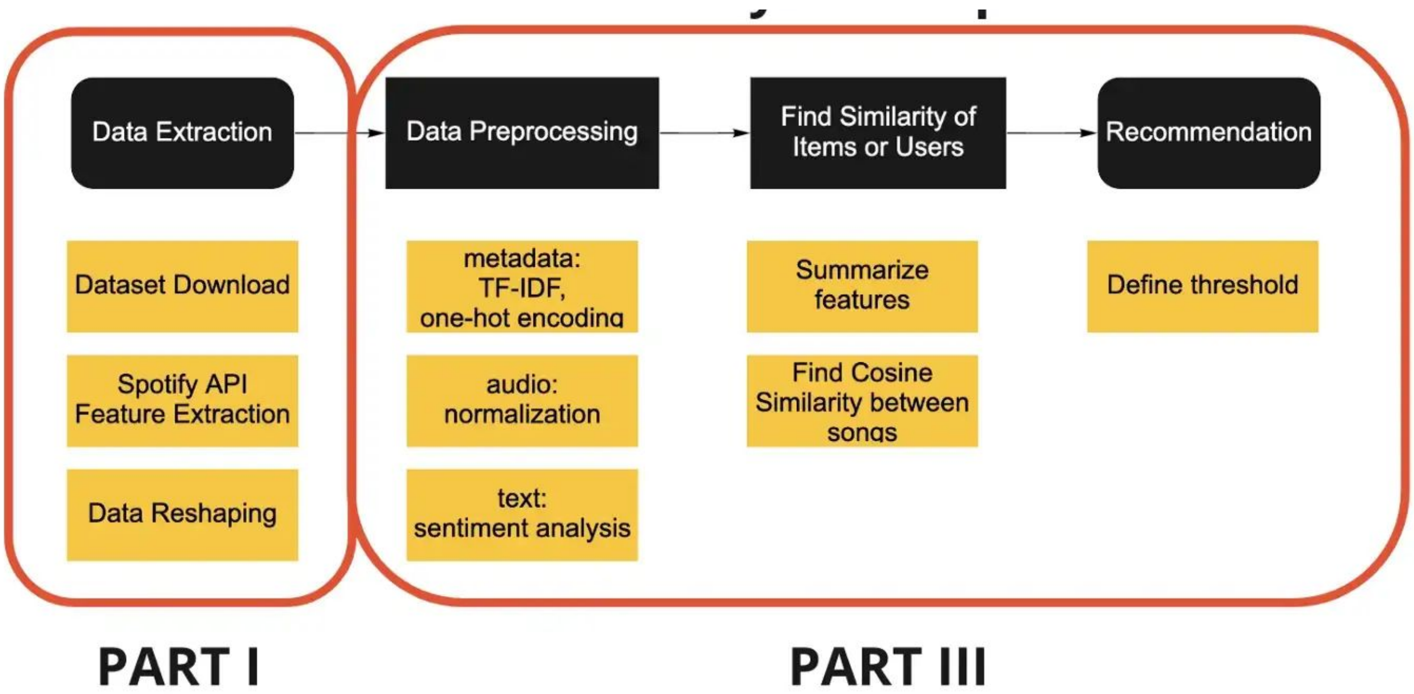
  If a lot of users listens to tracks a, b, c then probably those tracks are similar

- Content-based Filtering

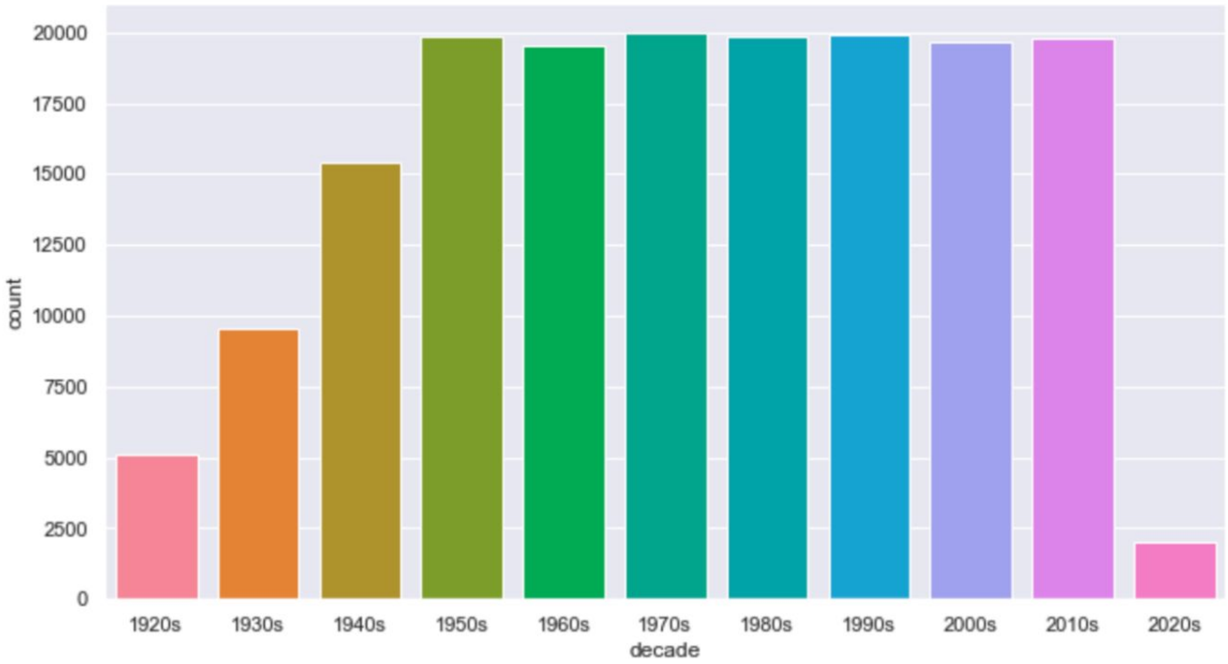
  Recommend songs that are similar to the other songs in the dataset.



Recommendation system pipeline



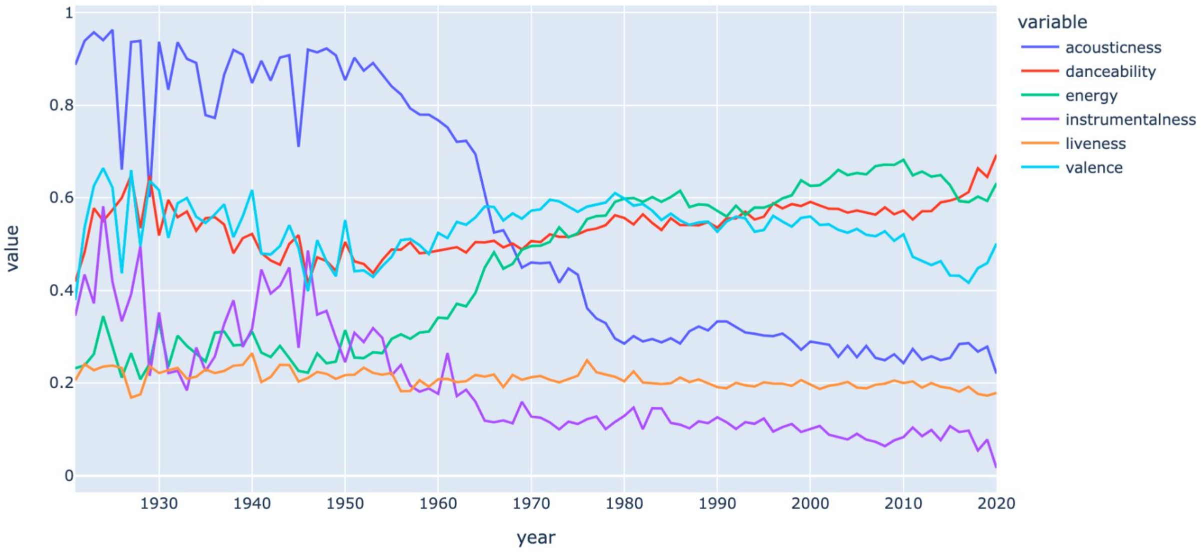
Data analysis - Music over time



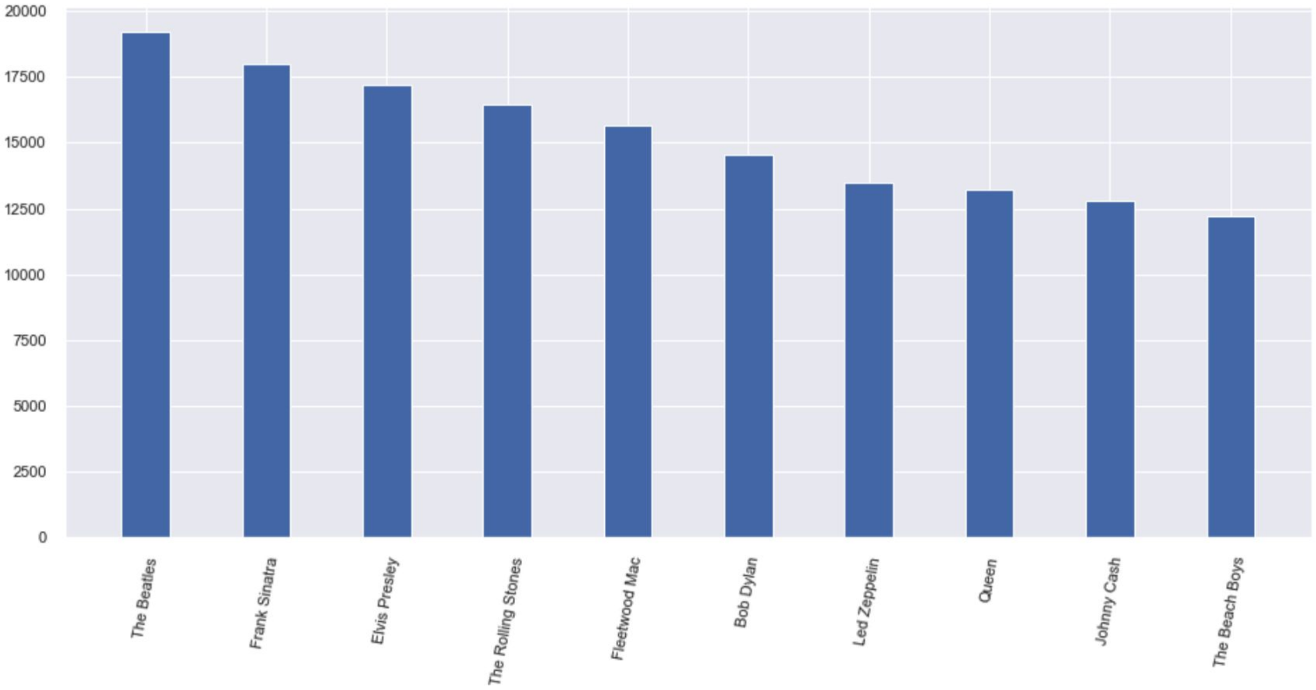
Data analysis - Sound features

In [23]:

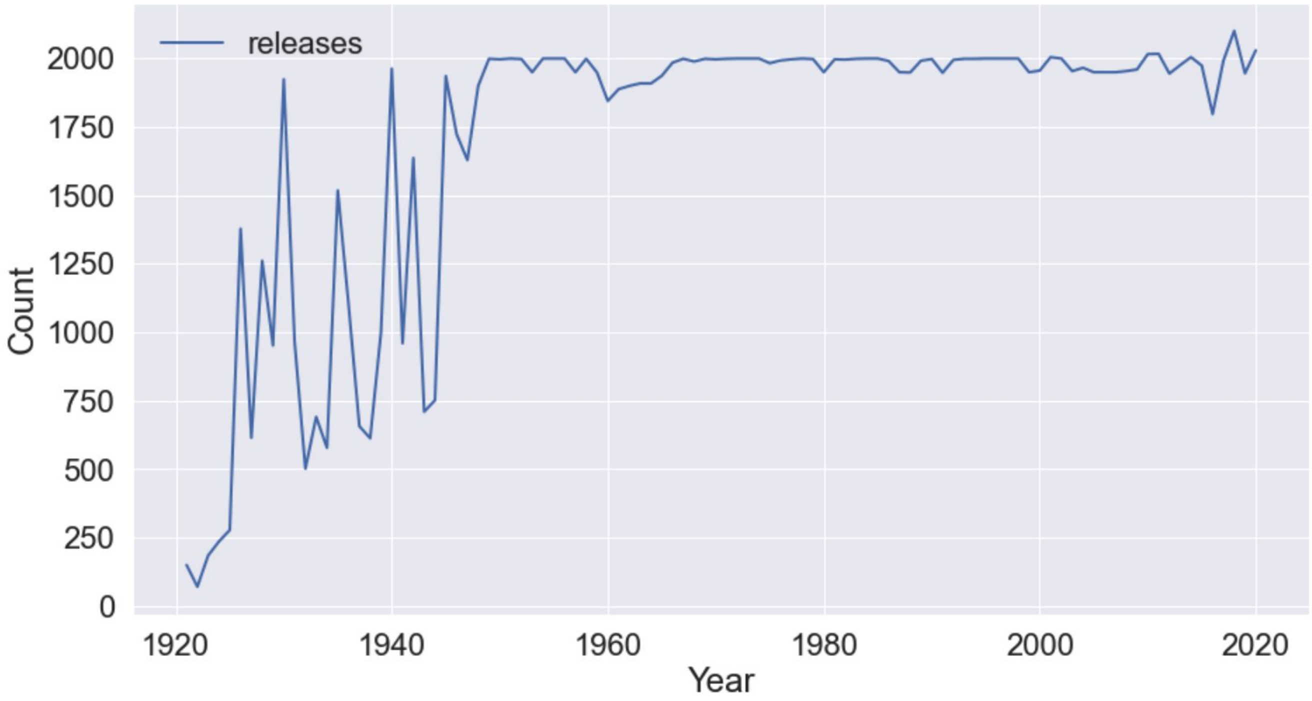
```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence']
fig = px.line(year_data, x='year', y=sound_features)
fig.show()
```



Data analysis - Popularity



Data analysis - Releases per year

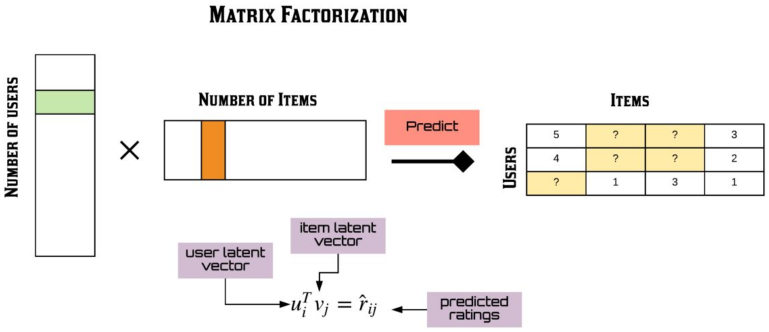


Song characteristics

KEY	VALUE DESCRIPTION
duration_ms	The duration of the track in milliseconds
key	The estimated overall key of the track
mode	Mode indicates the modality of a track. Major is represented by 1 and minor is 0
time_signature	The time signature is a notational convention to specify how many beats are in each bar
acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic
danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity
energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity
instrumentalness	Predicts whether a track contains no vocals
liveness	Detects the presence of an audience in the recording
loudness	The overall loudness of a track in decibels (dB)
speechiness	Speechiness detects the presence of spoken words in a track
valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track
tempo	The overall estimated tempo of a track in beats per minute (BPM)

Recommendation system

- Use combination of matrix factorization and classification to produce song recommendations for a particular user
- Extract latent features for users and songs
- Latent factors in conjunction with our audio features can train a classification model
- The model predicts classes of a high listen count versus a low listen count of song
- Use these predictions will then be used to recommend songs to a particular user.



Details for ML model

Follow-along the GITHUB link below for more details on Ensemble using Random Forest and XGBoost

https://github.com/akashtc/ml/tree/main/spotify_recommendation

Results

At prediction time, if we want to know if a user will listen to a song we will join the user features and the song features of that song and predict. The function 'get_top_songs', takes in a user id as an argument and recommends five songs by returning the five songs with the highest probability of belonging to our class representing a high listen count.

```
get_top_songs('f1ccb26d0d49490016747f6592e6f7b1e53a9e54')
```

	song_id	song	pred
3556	SOTHNRN12A8C143963	Fallin' Apart - The All-American Rejects	0.915466
1524	SOTRQEJ12AF72A45D7	Spies - Coldplay	0.900136
1526	SOWSZBE12AB01830DE	The Legionnaire's Lament - The Decemberists	0.896441
2805	SODKJWI12A8151BD74	From The Ritz To The Rubble - Arctic Monkeys	0.868117
2552	SOCATCA12AB0181E75	Dragon Queen - Yeah Yeah Yeahs	0.838229

Verifying against users listened songs

Besides AUC score, another way we can evaluate the recommendation system is by seeing if recommended songs are similar to what that user has listened to. Below are the users top 5 listened to songs.

```
In [56]: train2_df[train2_df.user_id == 'f1ccb26d0d49490016747f6592e6f7b1e53a9e54'].sort_values(by='listen_count', ascending=F
```

Out[56]:

	song	listen_count	label
338836	Woods - Bon Iver	6	1
267435	Creature Fear - Bon Iver	6	1
358035	Lonelily - Damien Rice	4	1
155277	Blindsided - Bon Iver	4	1
92315	Coconut Skins - Damien Rice	3	1
173111	Blood Bank - Bon Iver	3	1
267208	Flume - Bon Iver	3	1

Lessons learnt

- This type of model requires a ton of data.
- The number of user and item pairs is very large so we are training on a very small subset of the universe of possibilities.
- More data is necessary for a better score.
- Difficult to create a good recommender system with a small amount of data

Limitations

- Although not big enough, the dataset is still very large.
- The size of the data affects the quality of hyperparameter tuning of the model.
- Time and computing power limits our ability to use a grid search method for tuning our hyperparameters.
- This method would most likely have returned a better final AUC score.

Real-world architecture