MUSIC
RECOMMENDER SYSTEM

Data-driven insights for personalized song recommendations

PROBLEM DEFINITION

People live fast-paced lives with **minimal time** to search the overcrowded music marketplace for new music.

The streaming platform business model depends on **user engagement**, but it's challenging to maintain user attention.

Capturing and retaining user interest is especially difficult given the **diverse individual preferences** from user to user.

PROBLEM TO SOLVE

We need to identify new songs for users; offer highly individualized recommendations that **keep users interested and active.**

Can we **draw relationships between users and items** based on listening behaviors of other users?

Can we **accurately predict** whether a given user will enjoy a new song they've not yet interacted with?

LISTENER BEHAVIOR

We analyzed **users, artists, releases, song titles, years** and **play counts**. Play counts were our target variable.

To evaluate user preference, we aimed to predict listener interactions **from 0 - 10 plays per user**.

Users listened to songs that were **released between 1922 to 2010** in the provided dataset.

CHALLENGES
AND KEY CONSIDERATIONS

Data exploration uncovered issues with **missing data**.

Imputation strategies introduce some level of **uncertainty**.

Scant features available to draw comparisons between users and songs.

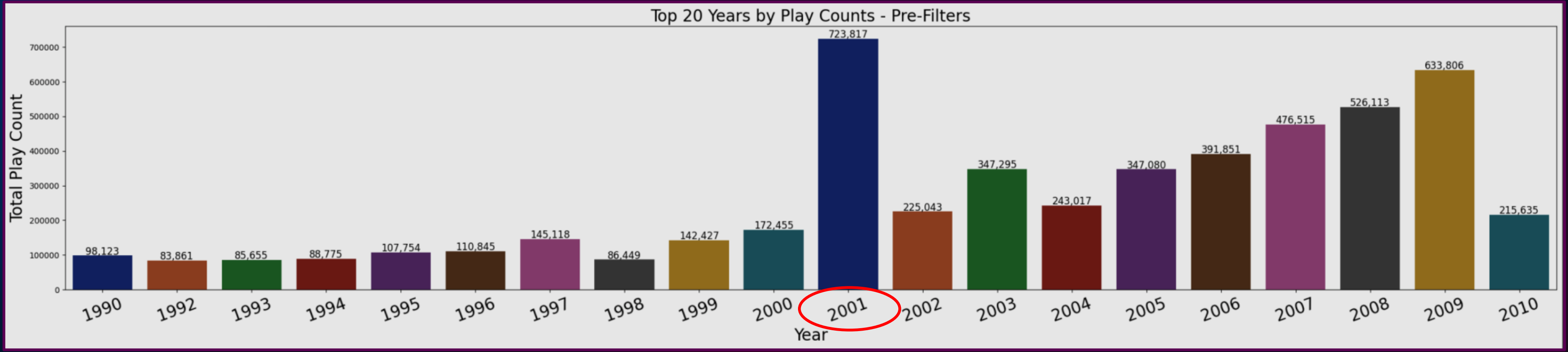
Features like **genre, song tags**, etc. could've assisted with precision.

Instating play count caps was necessary due to the **size of the initial dataset**.

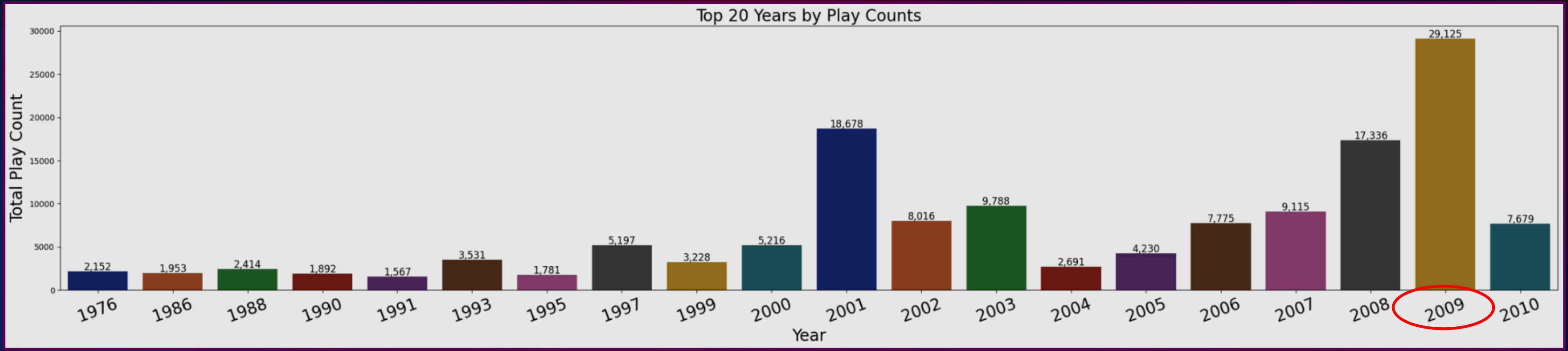
Trade-off insight for **computational efficiency** and **scalability**.

TOP YEAR BY POPULARITY

The top 3 years prior to instating cutoff filters included **2001, 2009 and 2008**.

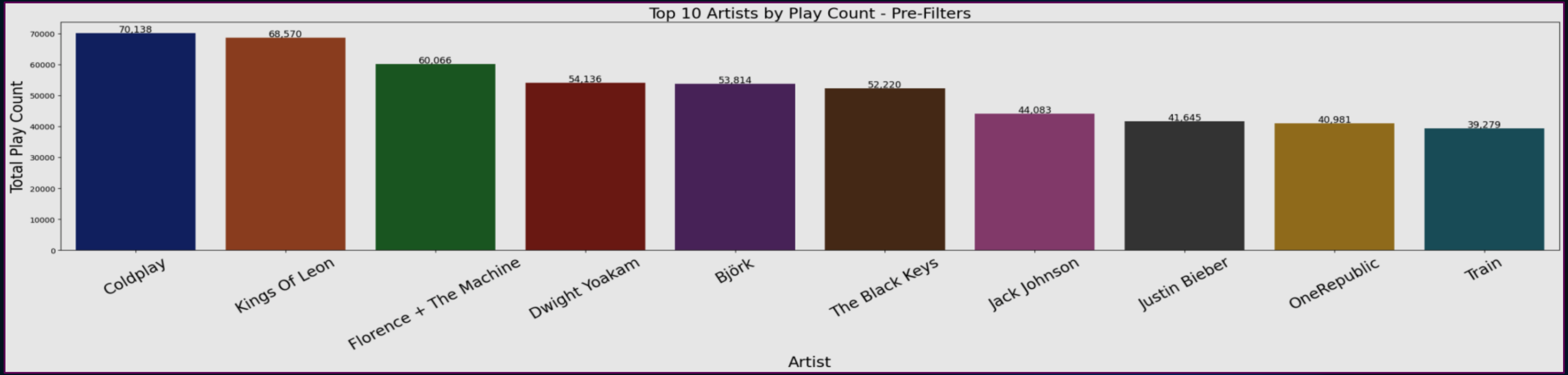


The top 3 years in our final dataframe after cutoffs included **2009, 2008, and 2001**.

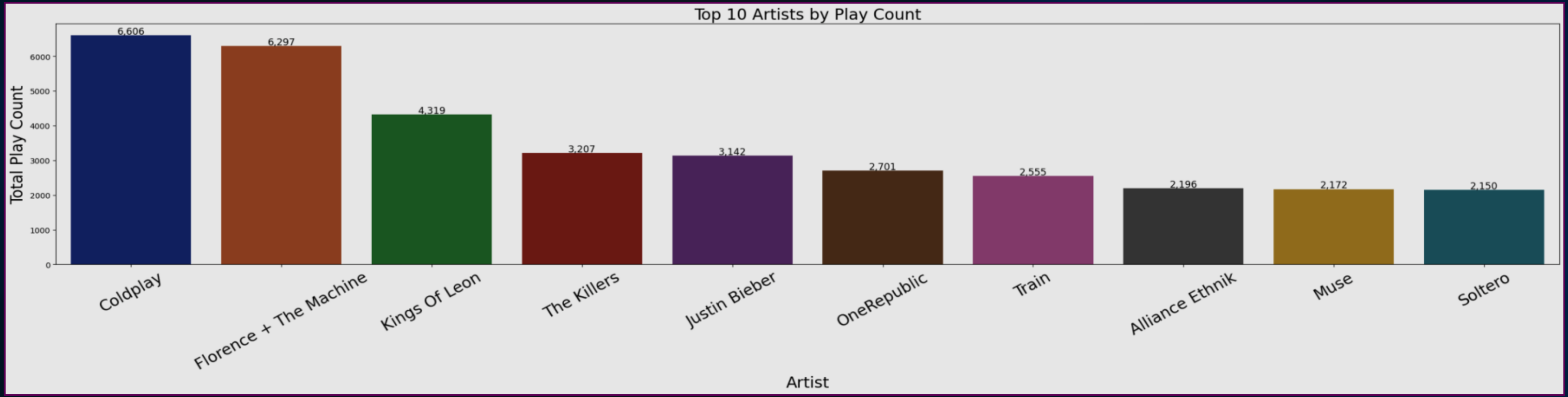


TOP **ARTIST**
BY POPULARITY

The top 3 artists prior to instating thresholds included Coldplay, Kings of Leon, and Florence + The Machine.



The top 3 artists in our final dataframe were in a slightly different order in terms of popularity.



MODEL EVALUATION

*__RMSE__ is high across all models, but we must consider the context—our interaction scale goes up to 10 play counts.

User User Similarity-Based Collaborative Filtering had the **highest precision (accuracy) post-tuning**, but it did not perform well on predictions.

Our Item Item model performed worse on predictions even though **recall was strong**.

SVD had the **highest recall and F1**, and it performed well on predictions.

Cluster-based model had promising estimation accuracy, but we see an **increase in RMSE** post-tuning that would require further attention.

USER USER	RMSE	PRECISION	RECALL	F1 SCORE
BASELINE	2.26	0.49	0.73	0.59
TUNED	2.12	0.50	0.73	0.59

ITEM ITEM	RMSE	PRECISION	RECALL	F1 SCORE
BASELINE	2.16	0.41	0.70	0.52
TUNED	2.08	0.48	0.78	0.59

SVD	RMSE	PRECISION	RECALL	F1 SCORE
BASELINE	2.07	0.49	0.78	0.59
TUNED	1.99	0.48	0.81	0.61

CLUSTER	RMSE	PRECISION	RECALL	F1 SCORE
BASELINE	2.12	0.47	0.65	0.54
TUNED	2.19	0.47	0.62	0.53

FINAL **MODEL**

SOLUTION DESIGN

- Of the models we tested, **Matrix Factorization using Singular Value Decomposition (SVD)** performed the best.
- We saw improvement in all assessed performance metrics for SVD except for precision, which could be **further optimized**.
- SVD performs well at capturing **latent features**, and it handles data sparsity effectively. Combining with a CB approach will yield the best results.

SOLUTION EXECUTION
AND RECOMMENDATIONS

- Enrich **dataset quality** by incorporating additional features.
- Collaborate with engineers to align on **system architecture** and **scalability**.
- Consider implementation of real-time **user feedback** on recommendations.
- Continue experimenting with **different model configurations** (SVD with content-based filtering).
- Conduct **A/B testing** to measure performance of difference iterations.
- Monitor **KPIs** and adapt strategy to ensure system drives tangible business outcomes.

EXECUTIVE SUMMARY

Matrix factorization using SVD can predict user interactions with songs within ~1 play count from actual.

We can **further optimize** the performance on recommendations through a hybrid approach including CF and CB strengths.

Continue with **A/B testing** to compare different versions of the system and ID which variations most effectively drive user engagement.

APPENDIX

PERFORMANCE HIGHLIGHTS
AND CODE EXAMPLES

```
[179] # Making prediction for user using a song where we know the interaction count
      svd.predict(3237, 8252, r_ui=10, verbose=True)

      user: 3237       item: 8252       r_ui = 10.00   est = 8.81   {'was_impossible': False}
      Prediction(uid=3237, iid=8252, r_ui=10, est=8.813730675494554, details={'was_impossible': False})

[180] # Making prediction for song this user has not interacted with previously
      svd.predict(3237, 5375, r_ui=5, verbose=True)

      user: 3237       item: 5375       r_ui = 5.00   est = 4.94   {'was_impossible': False}
      Prediction(uid=3237, iid=5375, r_ui=5, est=4.9381308084779, details={'was_impossible': False})

[181] svd.predict(3237, 512, r_ui=None, verbose=True)

      user: 3237       item: 512        r_ui = None   est = 4.53   {'was_impossible': False}
      Prediction(uid=3237, iid=512, r_ui=None, est=4.526863563058735, details={'was_impossible': False})

[182] svd.predict(62759, 7416, r_ui=10, verbose=True)

      user: 62759      item: 7416       r_ui = 10.00   est = 9.79   {'was_impossible': False}
      Prediction(uid=62759, iid=7416, r_ui=10, est=9.794446676661478, details={'was_impossible': False})

[183] svd.predict(62759, 6270, r_ui=5, verbose=True)

      user: 62759      item: 6270       r_ui = 5.00   est = 4.10   {'was_impossible': False}
      Prediction(uid=62759, iid=6270, r_ui=5, est=4.097429505801609, details={'was_impossible': False})

      svd.predict(62759, 512, r_ui=None, verbose=True)

      user: 62759      item: 512        r_ui = None   est = 3.24   {'was_impossible': False}
      Prediction(uid=62759, iid=512, r_ui=None, est=3.242479150644983, details={'was_impossible': False})

[185] svd.predict(6958, 1671, r_ui=2, verbose=True)

      user: 6958       item: 1671       r_ui = 2.00   est = 2.45   {'was_impossible': False}
      Prediction(uid=6958, iid=1671, r_ui=2, est=2.4545803831165265, details={'was_impossible': False})
```

- We see **healthy predictions** from our SVD model.
- Even though RMSE was high, the actual performance **exceeds expectations.**
- SVD predictions are **within ~1** play count of actual on the users we evaluated during testing.

PERFORMANCE HIGHLIGHTS

AND CODE EXAMPLES

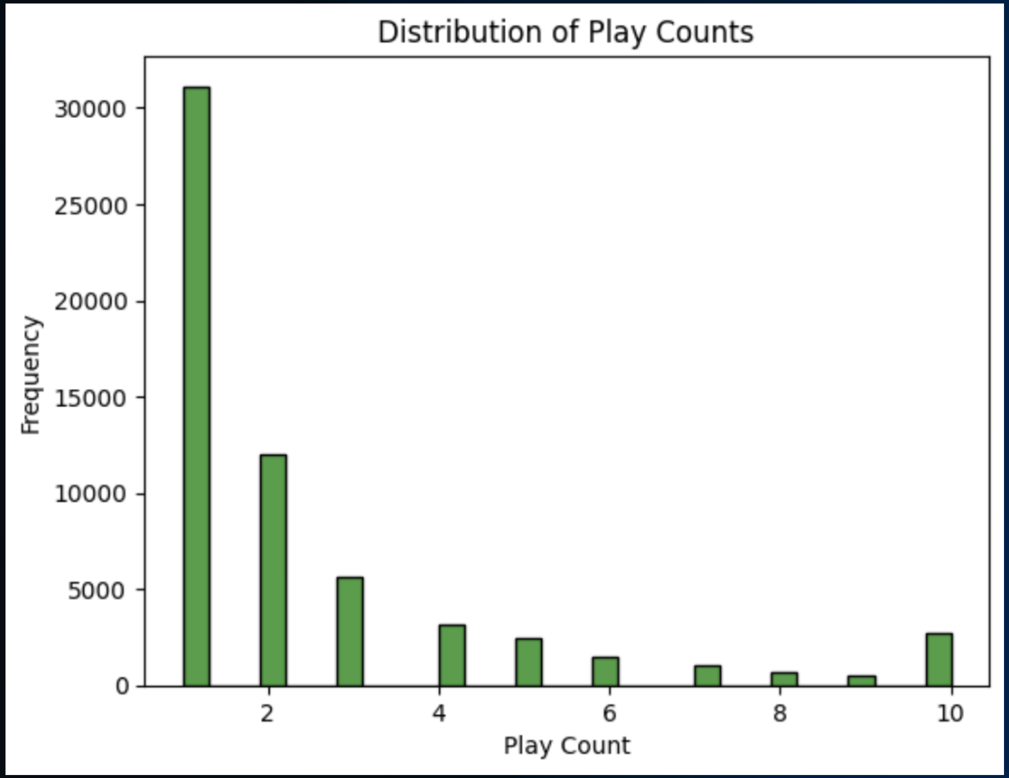
```
[241] # Make the recommendation for the song with title 'Learn To Fly' by Foo Fighters
      title_to_recommend = 'Learn To Fly'
      recommended_songs = recommendations(title_to_recommend, similar_songs)
      recommended_songs

      [245, 138, 207, 173, 178, 177, 176, 175, 174, 172]
      ['Everlong',
       'The Pretender',
       'Nothing Better (Album)',
       'Natural Anthem (Album)',
       'Human',
       'Until The End Of Time',
       "You Know I'm No Good",
       'Smile Like You Mean It',
       'The Ballad of Michael Valentine',
       'Fast As I Can']
```

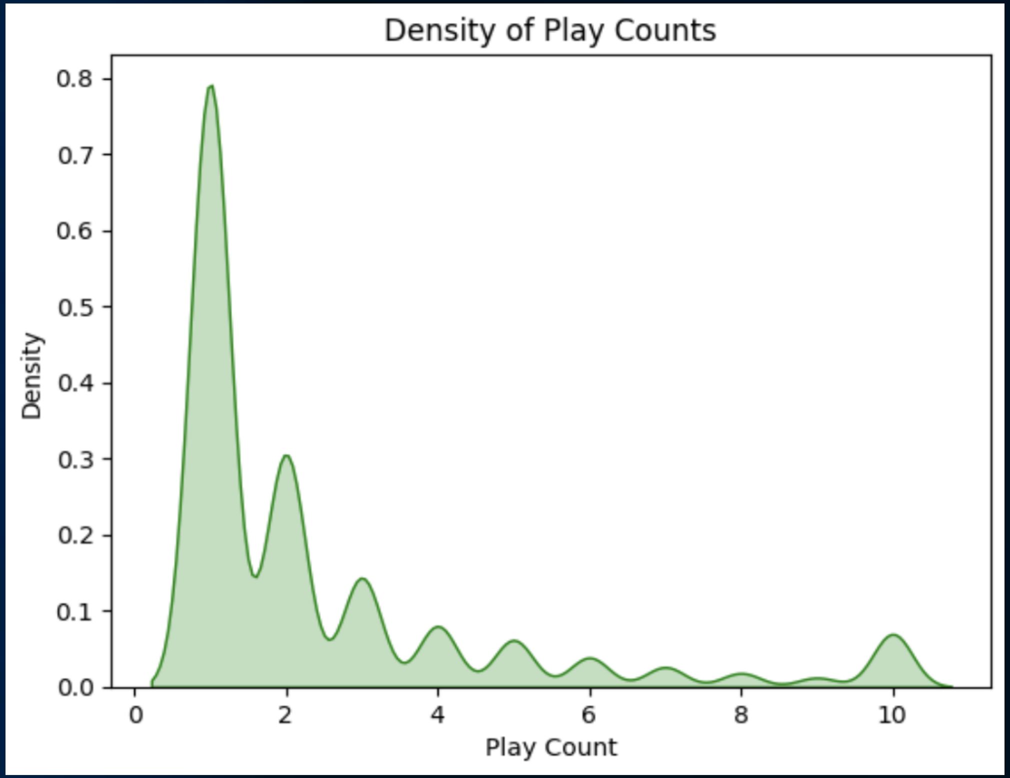
```
[244] title_to_recommend = "Mockingbird" # Eminem
      recommended_songs = recommendations(title_to_recommend, similar_songs)
      recommended_songs

      [58, 139, 132, 102, 83, 79, 149, 105, 88, 234]
      ["If I Ain't Got You",
       'Take A Bow',
       "Hailie's Song",
       'Superman',
       "Day 'N' Nite",
       "'Till I Collapse",
       "Nothin' On You [feat. Bruno Mars] (Album Version)",
       'Te Amo',
       "Hips Don't Lie (featuring Wyclef Jean)",
       'Rock Star']
```

```
title_to_recommend = "Secrets" # One Republic
recommended_songs = recommendations(title_to_recommend, similar_songs)
recommended_songs

[121, 0, 173, 178, 177, 176, 175, 174, 172, 180]
['All The Right Moves',
 'Harder Better Faster Stronger',
 'Natural Anthem (Album)',
 'Human',
 'Until The End Of Time',
 "You Know I'm No Good",
 'Smile Like You Mean It',
 'The Ballad of Michael Valentine',
 'Fast As I Can',
 'Read My Mind']
```

```
title_to_recommend = "The Scientist" # Coldplay
recommended_songs = recommendations(title_to_recommend, similar_songs)
recommended_songs

[23, 17, 33, 2, 19, 27, 21, 7, 28, 181]
['Shiver',
 'Sparks',
 'Fix You',
 'Clocks',
 'In My Place',
 "Don't Panic",
 'Yellow',
 'Brothers & Sisters',
 'Speed Of Sound',
 'LDN']
```

PERFORMANCE HIGHLIGHTS
AND CODE EXAMPLES





THANK YOU