Music Recommendation System

Final Presentation by
Vanessa Pinto

Summary of Problem

- Our lives are more fast paced and efficient.
- Less time to explore new artists and bands makes finding new music to listen to a challenge.
- Competition for top music streamer is strong, a multi-billion dollar market with a customer base of 523.9+ million global subscribers
- Important for streaming platforms like Spotify to keep their users engaged and make their experience better.

https://www.businessofapps.com/data/music-streaming-market/

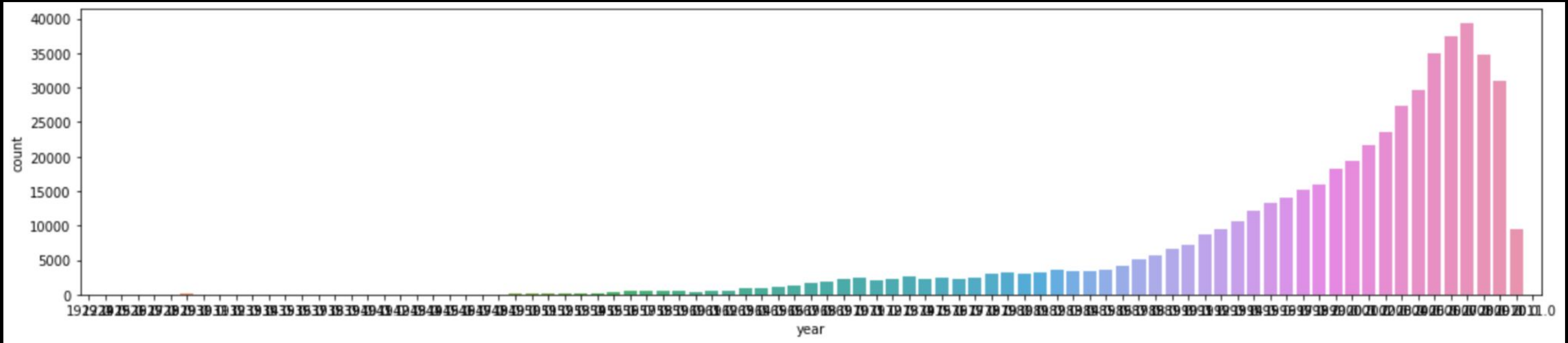
Problem to Solve

- **Spotify needs to consistently recommend relevant new songs and artists for their customers to explore**
- How can we use recommendation system models to achieve this goal?
- Can this be a measure to also maintain or even increase the number of monthly paying customers for the platform?

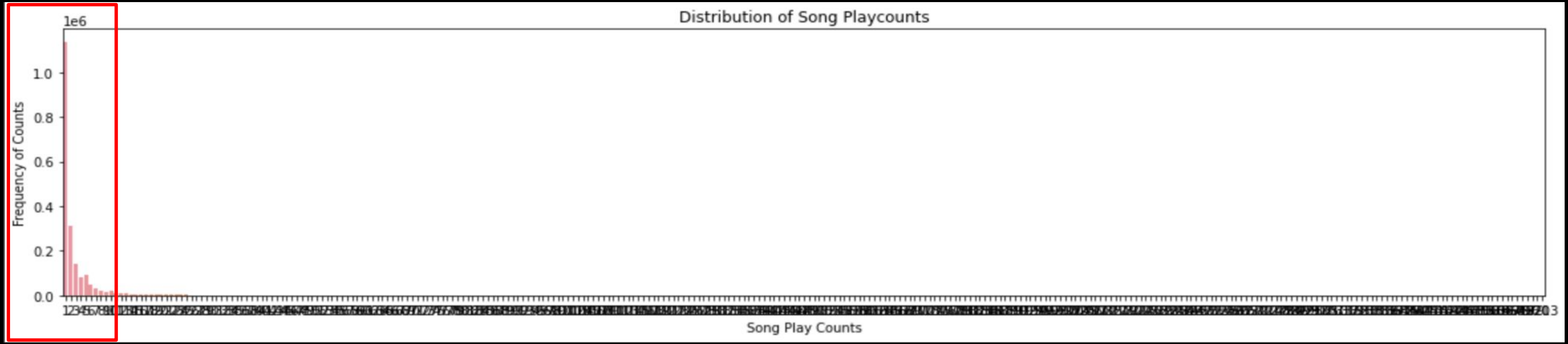
Key Takeaways

- Original dataset:
  - 76,353 unique users
  - 10,000 unique songs
  - Measure of rating: play count of song
- Final dataset:
  - 2466 unique users
  - 610 songs
  - Measure of rating: play count of song
- The average person in this dataset listens to a particular song 3 or fewer times.
  - Those that listen to a song more than this average are outliers and implies that they really like the song
- The majority of songs listened to are from the year 2000 and forward.
- As we will see in the next few slides, the model showed that we learn the most about users based on other similar users rather than from the songs a particular user listens to.

Distribution of Songs by Year



Distribution of Song Play Counts

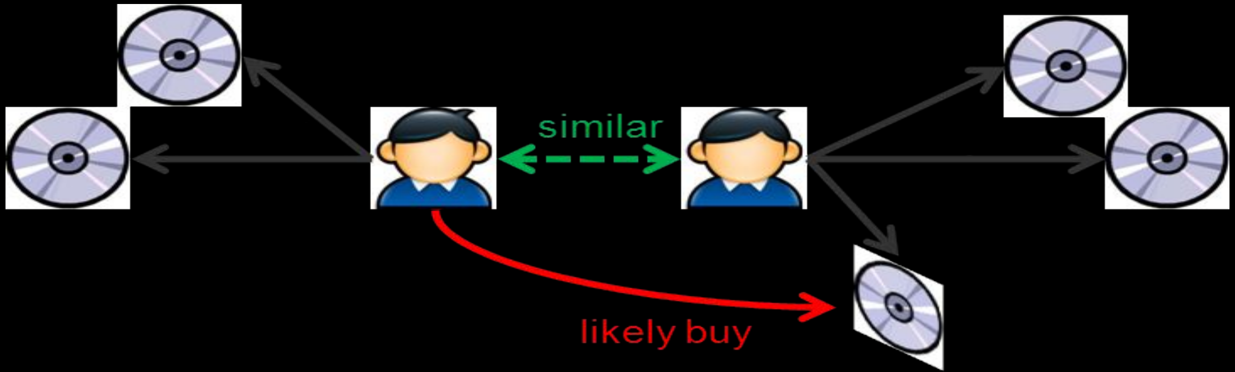


Proposed Solution Design

- **Similarity-based collaborative filtering recommendation systems** allow us to use users and items to find similarities and make recommendations
- These models search for neighbors based on similarity of product preferences and recommend products that those neighbors will buy
- **User-User** models find similarities between users and **Item-Item** models find similarities between items and make recommendations to users.

Proposed Solution Design (cont.)

- **Evaluation Metrics for Model Performance:**
  - RMSE (comparing train set to test set)
  - Precision (frequency of recommended songs that are relevant)
  - Recall (frequency of relevant songs that are recommended)
  - F1 score (combo of precision and recall)
- These four metrics are compared across the different tested models to find the best performing model



Final Model Solution

- Several recommendation system models were **tested and tuned** throughout this project to find the best performing one.
- Based on the comparison of the models and the evaluation metrics used, the **optimized user-user similarity based recommendation system** seems to be working the most accurately.
- It seems that the performance of this model in this case study shows that **songs are more likely recommended to users based on other similar users** rather than similar songs they have listened to or a combo of that.

Next Steps

- Consistently recommending relevant songs to their users.
- Tuning of recommendation system models (user-user similarity based model).
- Provide better recommendations to retain users (the more data, the more robust the insight).
- Brand new users - free trial and rank-based recommendation system
- Use popular songs as model for new songs
- New features to maintain engagement

Cost/Benefits & Risks/Challenges

- ❖ Free trials costs vs. long-term benefits
- ❖ Computational efficiency
- ❖ Maintaining top spot in music streaming and new customers.
- ❖ Increase in competition
- ❖ Novelty in music streaming

Executive Summary

- The music industry is an ever **growing industry with increasing demand**.
- Music streaming in particular seems to be significantly growing every year
- In order to keep **customers engaged** and remain the most subscribed platform, Spotify must consistently deliver quality and relevant music for exploration.
- Collaborative filtering recommendation systems seem to work very well in this domain, in particular **user-user similarity based models**.
- These user-user based models suggest that **similar users enjoy similar songs** (or items in general)

https://www.businessofapps.com/data/music-streaming-market/

Appendix

Comparison of Techniques & Performance

1. With all of this in mind, the optimized user-user similarity based recommendations system model gave the best performance, with the optimized item-item model as a close second in performance.
2. The optimized user-user collaborative filtering recommendation system has given the best performance in terms of:
    a. its recall score since this model resulted in the highest recall score across models
    b. its F1 score since this model was among the highest in this metric
    c. and lastly, in overall performance in predicting a playcount (compared to the true playcount), which seems to align with its high recall score as well

	Baseline User-User	**Optimized User-User**	**Baseline Item-Item**	**Optimized Item-Item**	**Baseline SVD**	**Optimized SVD**
RMSE **Presicion@k** **Recall@k** **F1 Score@k**	1.72 0.43 0.82 0.56	**1.65** **0.44** **0.82** **0.57**	1.63 0.37 0.68 0.48	1.60 0.45 0.72 0.55	1.60 0.44 0.73 0.55	1.56 0.45 0.76 0.57
Avg Prediction Performance (predicted/true playcount)	81%	**92.5%**	61%	91%	80%	77%

```
[ ] sim_user_user_optimized.predict(6958, 1671, r_ui = 2, verbose = True)

user: 6958       item: 1671       r_ui = 2.00   est = 1.39   {'actual_k': 18, 'was_impossible': False}
Prediction(uid=6958, iid=1671, r_ui=2, est=1.392681875259934, details={'actual_k': 18, 'was_impossible': False})
```

```
[ ] sim_user_user_optimized.predict(6958, 1056, r_ui = 2, verbose = True)

user: 6958       item: 1056       r_ui = 2.00   est = 2.04   {'actual_k': 26, 'was_impossible': False}
Prediction(uid=6958, iid=1056, r_ui=2, est=2.039487513745809, details={'actual_k': 26, 'was_impossible': False})
```

```
[ ] sim_user_user_optimized.predict(6958, 1050, r_ui = 5, verbose = True)

user: 6958       item: 1050       r_ui = 5.00   est = 5.22   {'actual_k': 11, 'was_impossible': False}
Prediction(uid=6958, iid=1050, r_ui=5, est=5.220743899120661, details={'actual_k': 11, 'was_impossible': False})
```

```
sim_user_user_optimized.predict(6958, 1787, r_ui = 2, verbose = True)

user: 6958       item: 1787       r_ui = 2.00   est = 1.88   {'actual_k': 30, 'was_impossible': False}
Prediction(uid=6958, iid=1787, r_ui=2, est=1.879918781556753, details={'actual_k': 30, 'was_impossible': False})
```

```
[ ] sim_user_user_optimized.predict(6958, 3232, verbose = True)

user: 6958       item: 3232       r_ui = None   est = 1.97   {'actual_k': 9, 'was_impossible': False}
Prediction(uid=6958, iid=3232, r_ui=None, est=1.9736456772894173, details={'actual_k': 9, 'was_impossible': False})
```

Thank You!