Steam Games Recommendation System

By: Hantao Lin

Problem

Who hasn't check up the reviews of a product before buying? We are in an era where recommendating system play an important role in customer market. In this project, we are going to build a reommendating system on steam games.

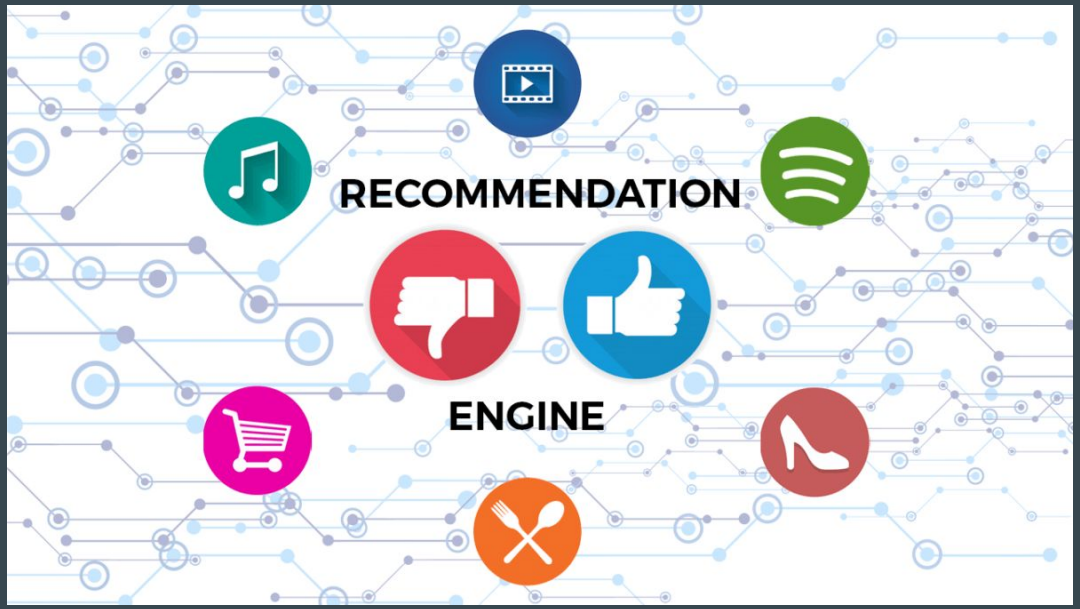

Figure 1. Picture on recommendation

Data Context

There are two datasets used in this project. Both dataset are downloaded from https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data describing Austrilian Steam information.

1. User information dataset
- The dataset contains information about users such as user id, user url, user reviews and etc.
2. Games dataset
- The dataset contains information regarding games such as game price, game id, publisher, genres and etc.

Data Wrangling

Because I do not need all the features in both data, I had to extract what is necessary. From dataset 1 I extracted user id, user recommendation, item id, and reviews. Then I left joined the two datasets together. After cleaning out the null values and unnecessary columns, the final dataset contains 15 columns with 49704 entries.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49704 entries, 0 to 59304
Data columns (total 15 columns):
user_id         49704 non-null object
item_id         49704 non-null object
recommend       49704 non-null int64
review          49704 non-null object
publisher       49704 non-null object
genres          49704 non-null object
app_name        49704 non-null object
title           49704 non-null object
url             49704 non-null object
release_date    49704 non-null object
tags            49704 non-null object
specs           49704 non-null object
price           49704 non-null float64
early_access    49704 non-null object
developer       49704 non-null object
dtypes: float64(1), int64(1), object(13)
memory usage: 6.1+ MB
```

Figure 2. Dataset info

Storytelling

At this part of the project, I am going to do an exploratory data analysis by plotting different graphs.

Figure 3. Exploratory Analysis

Question: How generous are people giving out recommendation on games?

About 89.4% people gave games “would recommend” and 10.6% games received “would not recommend. It seems like australian people are very happy about most of the game sold on steam.

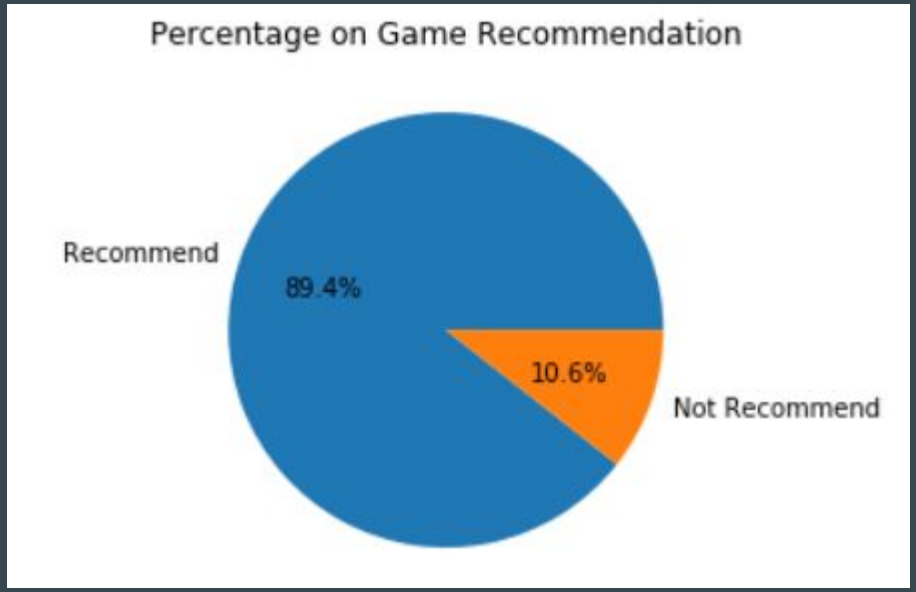

Figure 4. Percentage on Game Recommendation

Question: What's the percentage of games launch early access?

Only 5.7% of all games on steam launch early access. For those that didn't launch early access, I assume the publisher either has a great team and is very confident at their product or they are just carefree.

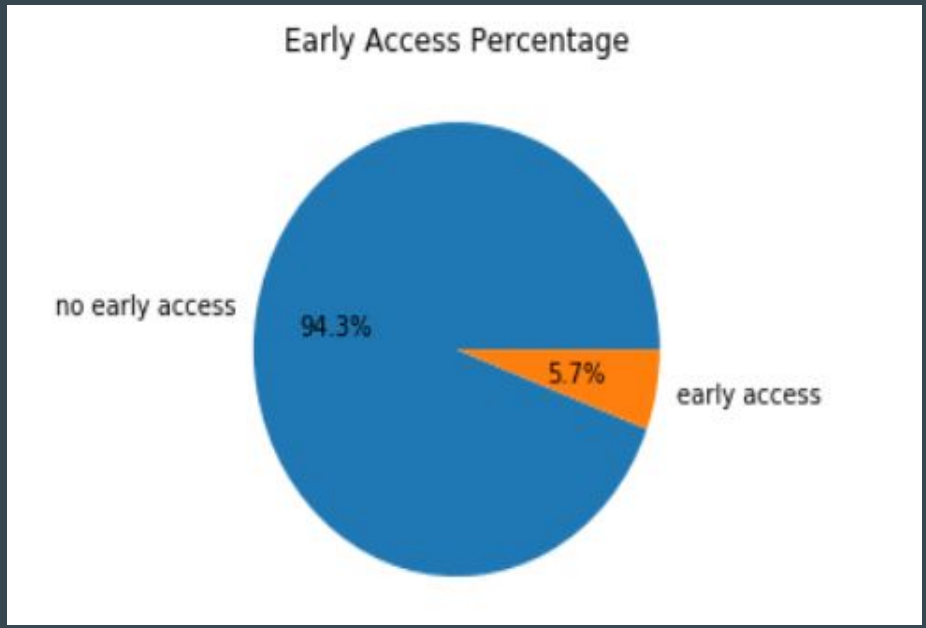

Figure 5. Early Access Percentage

Question: What's the top 10 most played game ?

It's no surprise to see Counter Strike as the most popular game. What surprised me is that Gary's Mod ranked #3 on the list. That just proved I am not an australian.

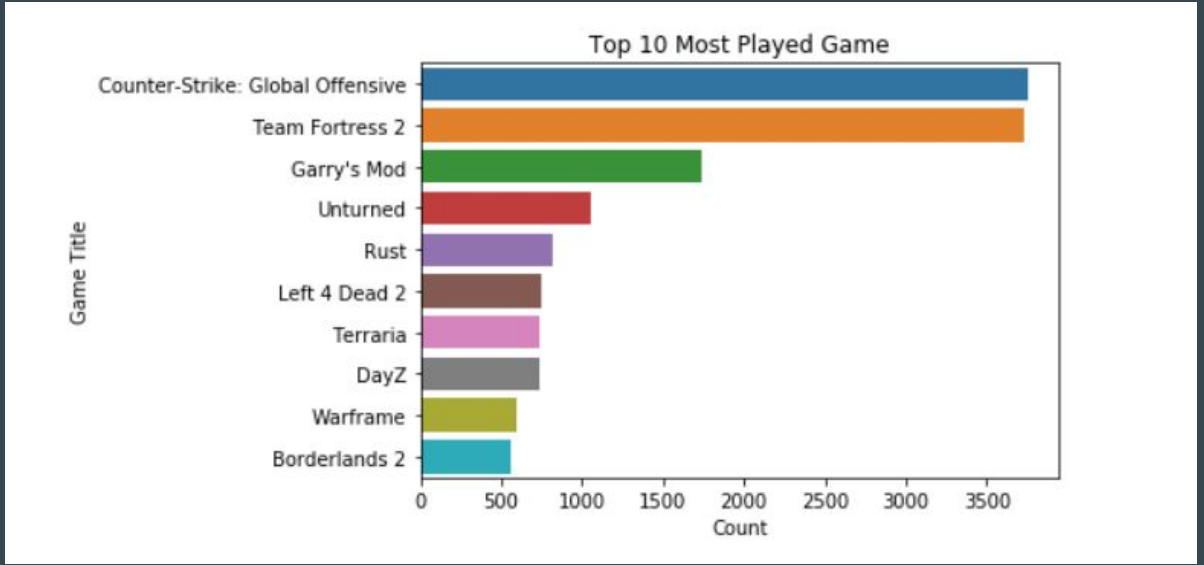

Figure 6. Top 10 Popular Games

Problem: How's most game priced ?

About 80% of the game priced under $20. It's rare to see games exceed $20. It is because that australian people do not purchase games over $20?

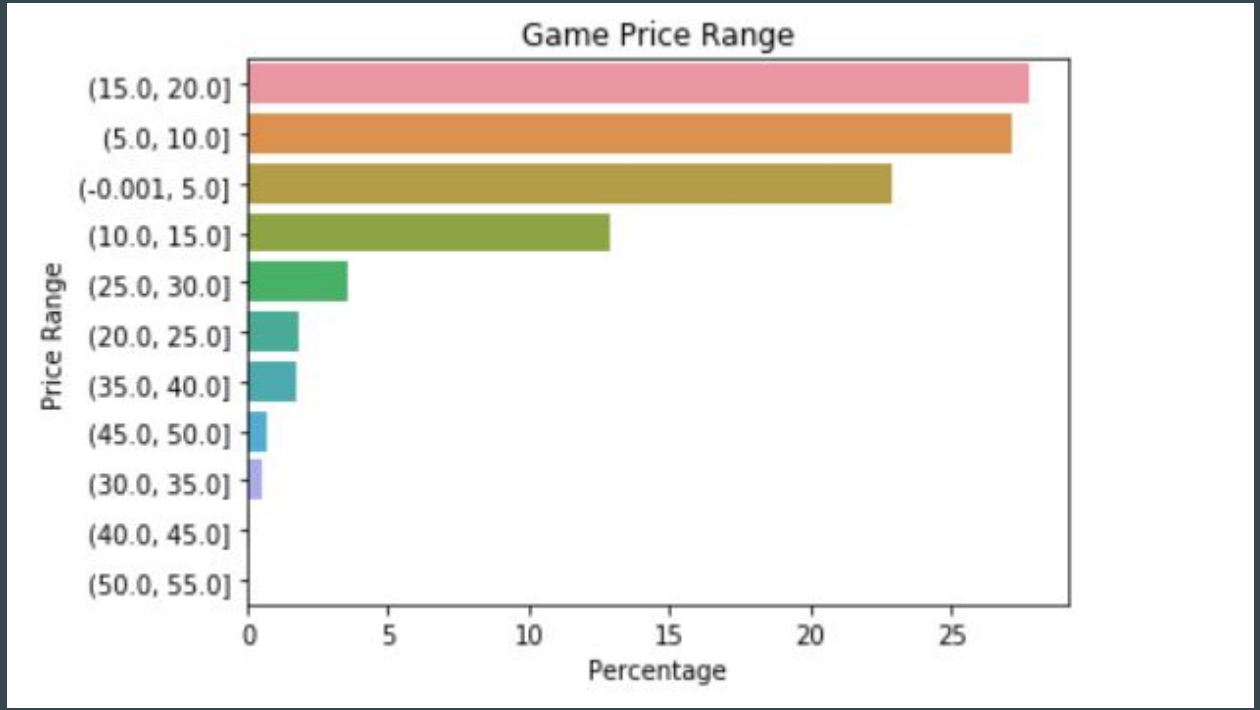

Figure 7. Game Price Range

Problem: What's the most popular genre ?

The most popular genre is "Action" games. That makes sense because the most played game is Counter Strike. Australian people really don't play much games outside of action, strategy, and RPG games.

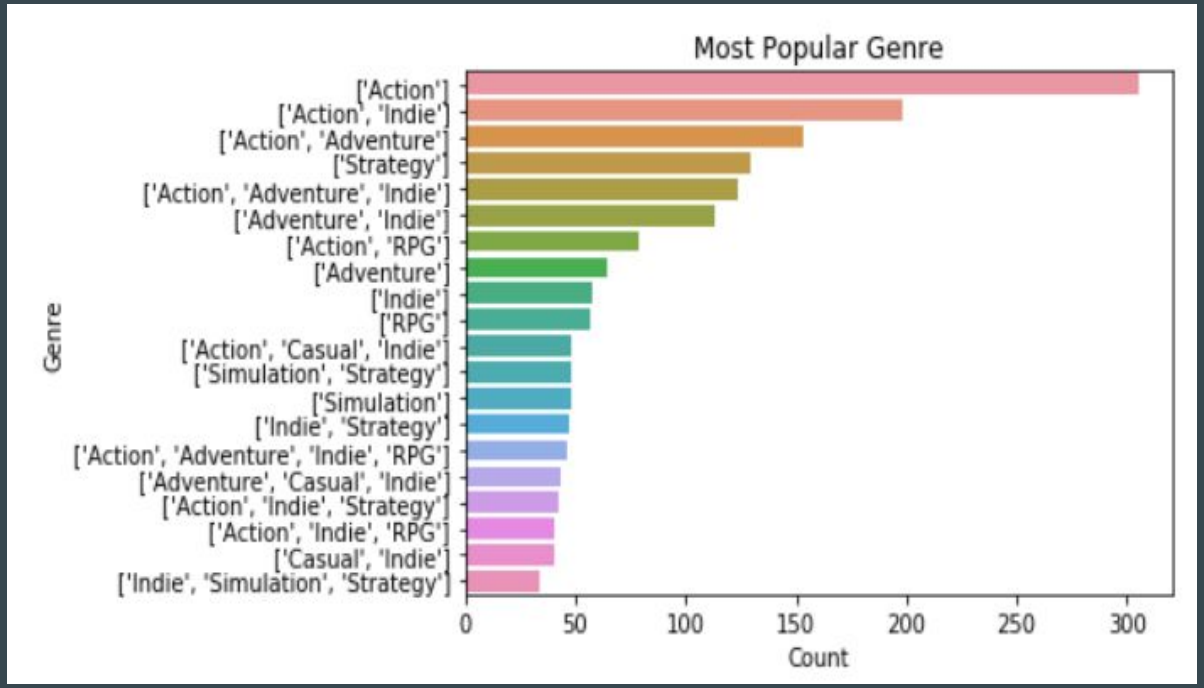

Figure 8. Most Popular Genres

Problem: Top Developers?

It's totally not surprised to see Valve as the top developer since Steam is their company. I have never heard of the other game company except Ubisoft. This is interesting.

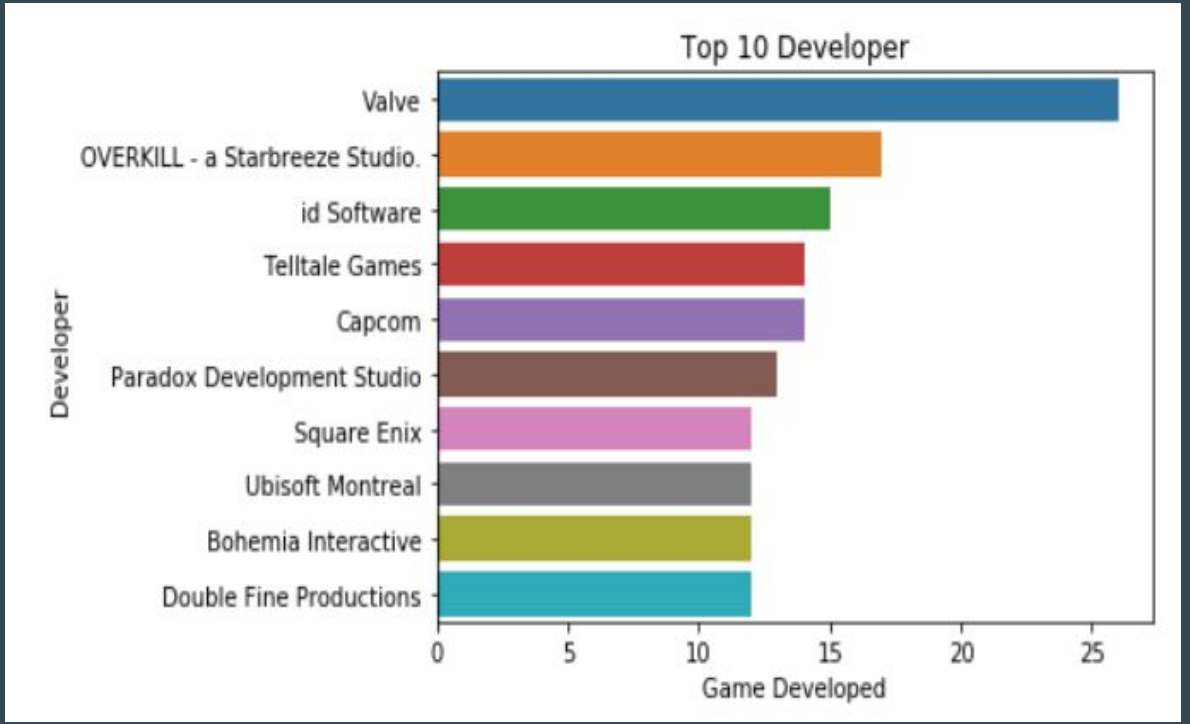

Figure 9. Top 10 Developers

Statistical Analysis

From storytelling I gained many interesting information. In this part of the process, I am going to further explore some of the information presented and perform A/B testing.

Figure 10. Statistical Analysis

Do [‘Action’] games cost more than [‘Action’, ‘Indie’] games?

Null Hypothesis:

Average price of [‘Action’] games ≥ Average price of [‘Action’, ‘Indie’] games.

Alternative Hypothesis:

Average price of [‘Action’] games < Average price of [‘Action’, ‘Indie’] games.

Alpha:

5%

```
# Use boostrap to find the distributions and calculate CI
diff_list=[]
act_ind_meanlist=[]
act_mean_list=[]
for i in range(10000):
    act_ind_samp=np.random.choice(act_ind['price'],len(act_ind['price']))
    act_samp=np.random.choice(act['price'],len(act['price']))
    act_ind_mean=np.mean(act_ind_samp)
    act_ind_meanlist.append(act_ind_mean)
    act_mean=np.mean(act_samp)
    act_mean_list.append(act_mean)
    diff=act_mean-act_ind_mean
    diff_list.append(diff)
percentile=np.percentile(diff_list,[5,95])
print('95% Confidence Interval: '+str(percentile))
```

```
95% Confidence Interval: [5.79101534 6.50657473]
```

Figure 11. Code of Bootstrap for game genres A/B testing

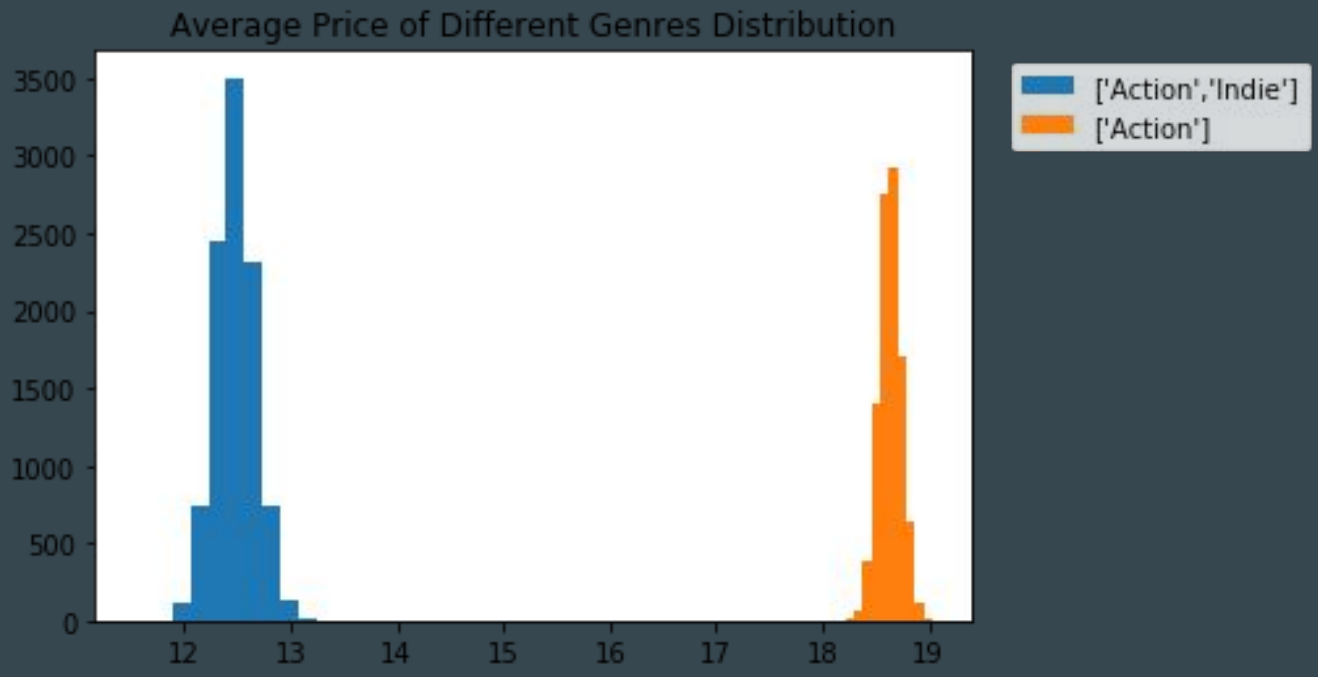

Figure 12. Average price of Different Genres Distributions

Conclusion

From the distribution and the confidence interval, it is lucid that the [‘Action’] games have higher average price than [‘Action’, ‘Indie’] games.

Figure 13. Inide, Action vs Action

Does multiplayer games cost less than single player games?

Null Hypothesis:

Average price of single player games ≥ Average price of multiplayer games

Alternative Hypothesis:

Average price of single player games < Average price of multiplayer games

Alpha:

5%

```
# Use boostrap to find the distributions and calculate CI
diff_list=[]
multi_meanlist=[]
single_mean_list=[]
for i in range(10000):
    multi_samp=np.random.choice(multi['price'],len(multi['price']))
    single_samp=np.random.choice(single['price'],len(single['price']))
    multi_mean=np.mean(multi_samp)
    multi_meanlist.append(multi_mean)
    single_mean=np.mean(single_samp)
    single_mean_list.append(single_mean)
    diff=single_mean-multi_mean
    diff_list.append(diff)
percentile=np.percentile(diff_list,[5,95])
print('95% Confidence Interval: '+str(percentile))
```

```
95% Confidence Interval: [-2.93274549 -2.60298355]
```

Figure 14. Code for bootstrap for game specs

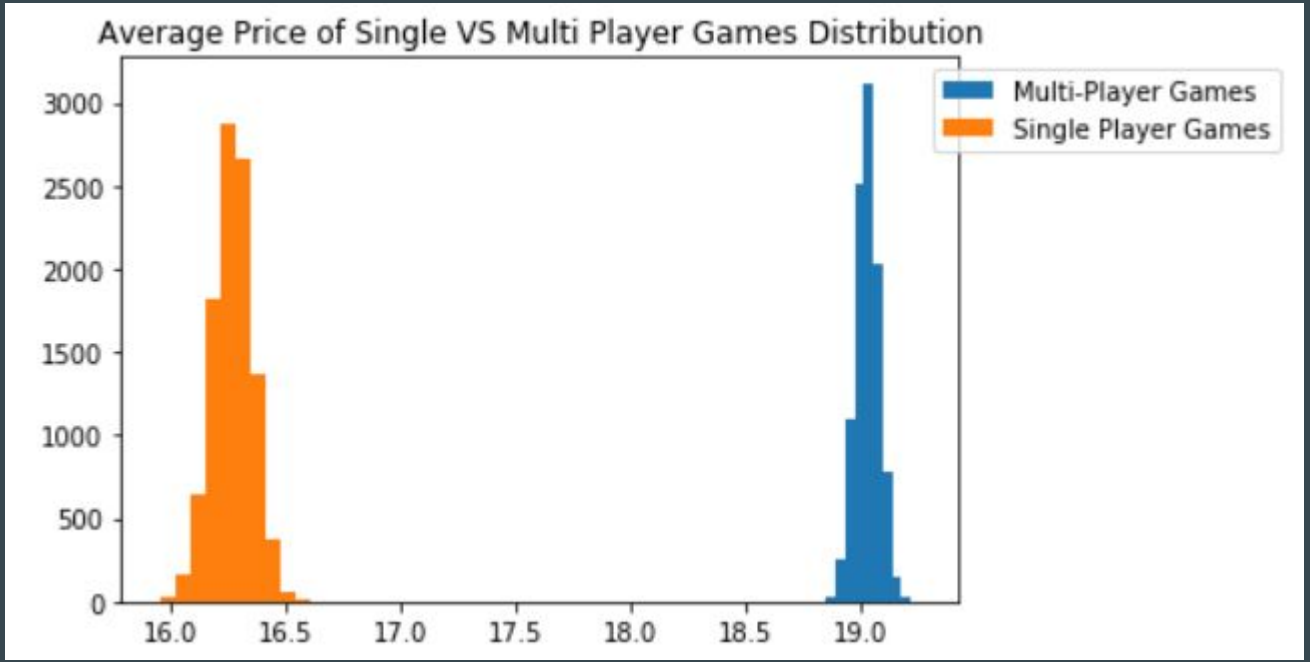

Figure 15. Average Price of Single VS Multi Player Games Distributions

Conclusion

From the distribution and the confidence interval, I conclude that single player games do not cost more than multiplayer game. In fact, multiplayer games cost more than single player games.

Figure 16. Multiplayer vs Single Player

Which game has better reputation in terms of recommendation? (CS: Go VS Team Fortress 2)

Null Hypothesis:

Percentage of recommendation from CS:GO ≥ Percentage of recommendation from Team Fortress 2

Alternative Hypothesis:

Percentage of recommendation from CS:GO < Percentage of recommendation from Team Fortress 2

Alpha:

5%

```
# Use boostrap to find the distributions and calculate CI
diff_list=[]
cs_percent_list=[]
tf_percent_list=[]
for i in range(10000):
    cs_samp=np.random.choice(cs['recommend'],len(cs['recommend']))
    tf_samp=np.random.choice(tf['recommend'],len(tf['recommend']))
    cs_percent=round((cs_samp.sum())/len(cs_samp)*100,2)
    cs_percent_list.append(cs_percent)
    tf_percent=round((tf_samp.sum())/len(tf_samp)*100,2)
    tf_percent_list.append(tf_percent)
    diff=cs_percent-tf_percent
    diff_list.append(diff)
percentile=np.percentile(diff_list,[5,95])
print('95% Confidence Interval: '+str(percentile))
```

```
95% Confidence Interval: [-4.98   -3.2795]
```

Figure 17. Code for bootstrap for game recommendation rate

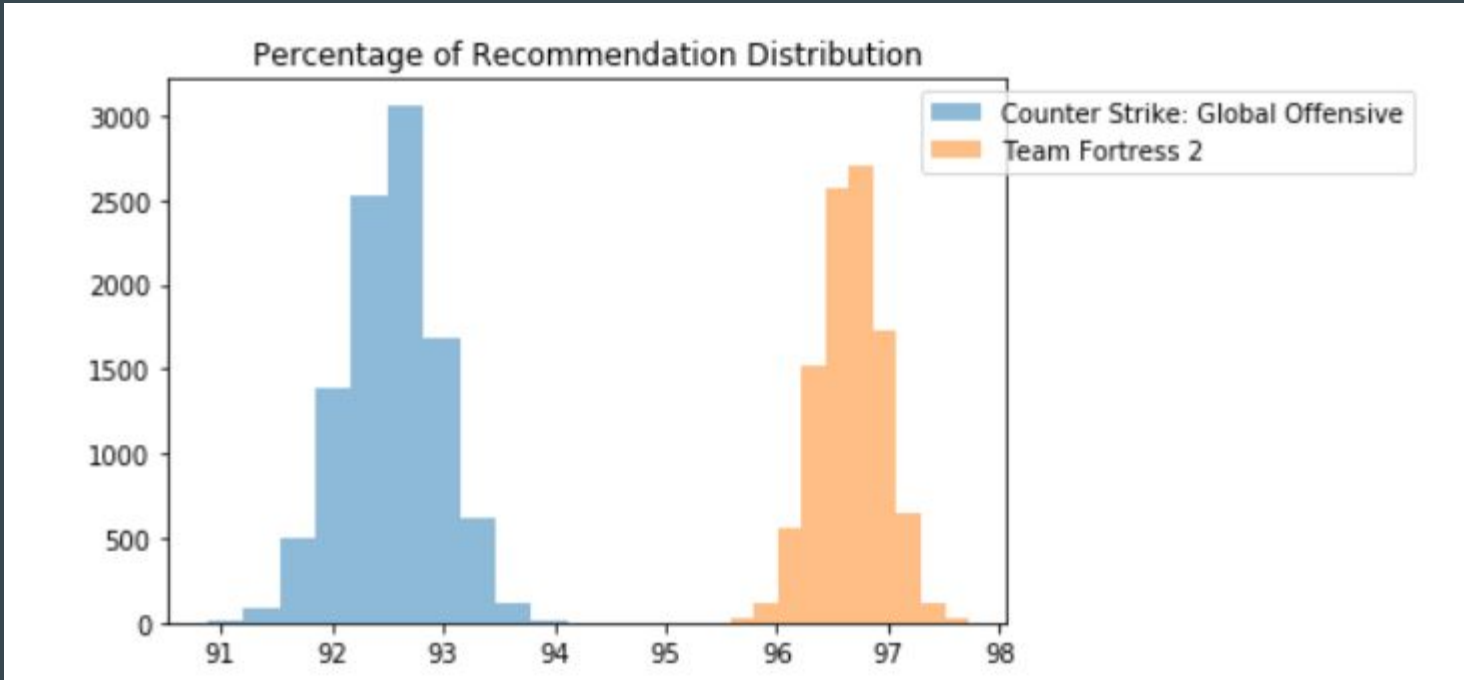

Figure 18. Percentage of Recommendation Distributions

Conclusion

Based on the evidences presented, CS:Go didn't have a greater recommendation rate than Team Fortress 2. In fact, Team Fortress 2 has greater recommendation rate than CS:Go.

Figure 19. TF2 VS CSGO

Building Recommend System

In this part of the project, I am going to build an item-item collaborative filtering recommender system because the number of users exceed the number of games. With the choice, it will saves me massive time.

I am going to use five different algorithms:

1. KNNBasic
2. KNNWithMeans
3. KNNWithZScore
4. SVD
5. OneSlope

Performance of Each Algorithms

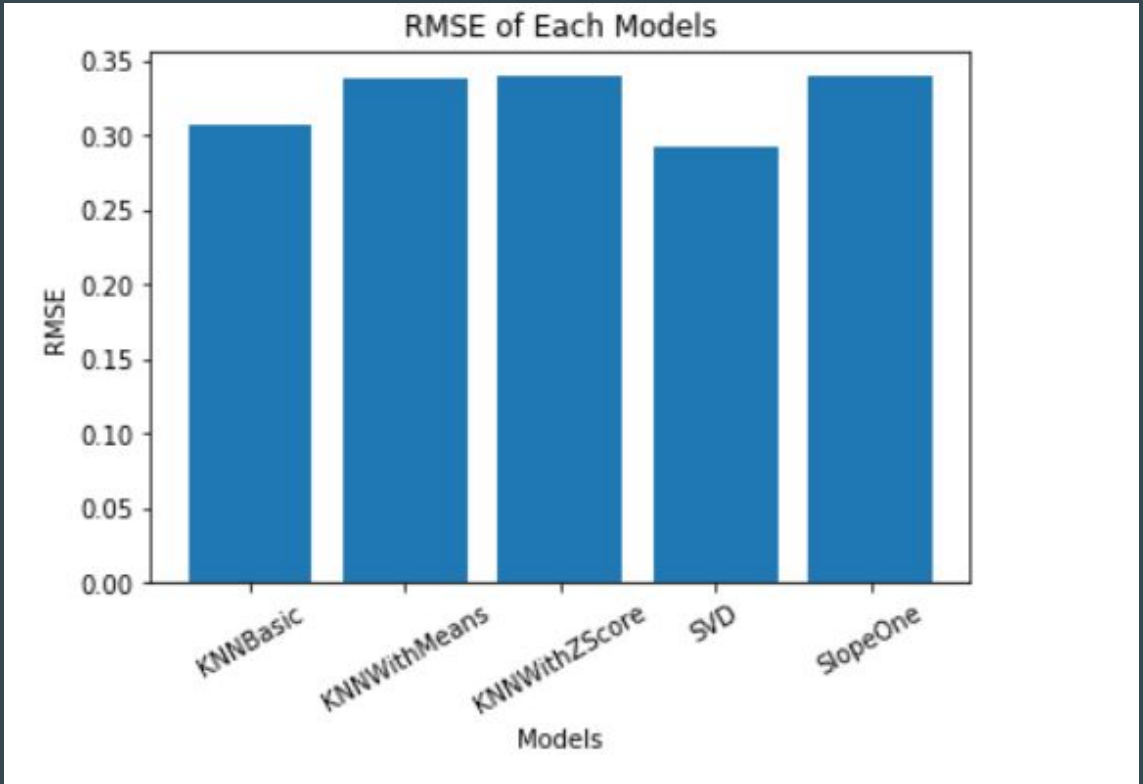

Figure 20. RMSE of Each Algorithm

Result

It is clear from the graph that based on the RMSE of all the models, SVD has the lowest RMSE score which makes it the best model to use for building an item-item collaborative recommender system.

Final Thought

According to https://neoteric.eu/ , “59% of shoppers who have experienced [recommender system] say it has a big influence on their purchase decisions”. This is exactly why a recommender system is beneficial to company like Amazon, Steam, and many other company that has a shopping website online.

There are many more different algorithms that can be used such as hybrid recommender system, content-based recommender system and etc. I will definitely try all these algorithms on different datasets if I have the chance.

Dataset Contributors

Self-attentive sequential recommendation

Wang-Cheng Kang, Julian McAuley

ICDM, 2018

pdf

Item recommendation on monotonic behavior chains

Mengting Wan, Julian McAuley

RecSys, 2018

pdf

Generating and personalizing bundle recommendations on Steam

Apurva Pathak, Kshitiz Gupta, Julian McAuley

SIGIR, 2017

pdf