Matthew Lepis
CPE 551-WS
5/19/2021
Final Project

Steam Store Database Analysis

Introduction

The objective of this project was to analyze steampowered.com's database of games and their respective user reviews for an understanding of what the best video games on the market have in common. Using skills obtained in CPE-551, I constructed a program using Python 3.8 to do just this. The criteria I used for analysis include the game's developer(s), publisher(s), the various categorical features the game included, and lastly the genres that apply to a given video game.

Dataset

I had originally intended on creating a data crawler to extract the information I wanted from steampowered.com's (otherwise known as Steam) website and develop my own dataset but, I thought that I should enact the objective first, so I found a suitable dataset online. The only issue with this was that the data I processed was not the most current, specifically, it was from May 2019, but again this would be more than serviceable for the scope of the project's objective.

The dataset I chose was a dataset from Kaggle.com, developed by user: Nik Davis, called: "Steam Store Games (Clean dataset)" (Davis, 2019). The dataset creator scraped data from Steam using SteamSpy.com's APIs. Within this dataset, 27,000 different games had their data stored. Originally the dataset had over 20 different features, but I only used four of its textual features: developer, publisher, categories, and genre; and two of its integer features: number of positive reviews, and number of negative reviews.

Before I could correlate the data, I had to clean the dataset for operation and future users' intentions, this included filtering out non-English reviews or more specifically non-ASCII based text information from the dataset features, but overall, the data was already rather clean, as the title of the dataset suggested. Any further preprocessing of the dataset I left to the end-user to decide what was most relevant for them, and that data was collected at runtime, through UI console prompts. The user can filter games out of the dataset based on the following criteria:

1. Total number of user reviews of the game.
2. Total number of accounts that owned the game.
3. How many games should be considered relevant for each given textual data feature group.
    a. This metric allows the user to control the volatility of their results by filtering out outliers from the dataset, like a developer that only has one game being the best developer. If the end-user decided that was not appropriate for their needs it can be filtered out.
    b. Note, for Categories and Genres this number is multiplied by 30, as a lot more games overlap in categorical feature and genres than developers and publishers.
4. What percent of the dataset should be used during correlation.
5. If the sample should be randomly determined or statically chosen.
    a. These last two criteria were mostly a program testing feature, but I thought it would be best to leave it intact because of future project intentions that I will touch on in the future direction section.

Implementation Details

Most of the algorithmic work in this project was string manipulation/interpretation, as well as pandas DataFrame manipulation and python list manipulation; all done through various nested for loops and other such basic methods of iteration and comparison.

Specifically, however, the main idea was to compare the percentage of positive user reviews and attribute them to their respective textual group (developer/publisher/category/genre), by iteration through the DataFrame imported dataset and counting how many games belonged to each textual grouping, then averaging the value as a score to rank them based upon.

For example,

In the program's Developer calculation, let us say Valve Corporation has 3 games on the market that met the end-user's filtering criteria, each has a user review rating of: 93%, 78%, and 84%. Valve's Developer Score would be 85%, this would then be saved, and the program would move onto the next developer. And at the end, the best developers would appear on the developer graph for the end-user's inspection.

Lastly, in addition to the user-defined data filters spoken of above in the dataset section, the user can define two additional parameters

1. The number of groupings to show on the data graphs the program generates
    a. This effectively allows the user to determine how many of the best in a grouping shows in the end figures.
2. The user can determine if the program should show the graphs during runtime or just save them instead.

Future Directions

When I find additional time to work on this project I would like to additional modules for more in-depth analysis. Things like price, release date, platform, average user playtime, come to mind.

Other ideas I had included: developing a data crawler to gather a live dataset or at least a semi-live dataset that would be much more accurate for the current marketplace.

A quite simple addition that I simply overlooked until writing this was to include a feature to filter out a number of games that are to be deemed relevant for each textual feature. As of right now the program accepts only one number to filter how games are to be considered for all four current textual groupings and this is rather limiting for the overall scope, so this is something I would correct.

Lastly, and most ambitiously I would Implement a machine learning model to predict the positive user review percentage of a game with user-definable features or a game developed similar to the principles of a given developer. This feature reminds me of an indie game called Game Dev Tycoon, and I think it would be fun and informative to implement.

Conclusion

In conclusion, I found this project to be a good test of my programming ability in python, as well as an interesting concept for me to implement and take advantage of as an avid video game fan, as during testing I was able to discover games that I might like that I otherwise had not heard of, perhaps this could be an additional application of my future machine learning model.

References

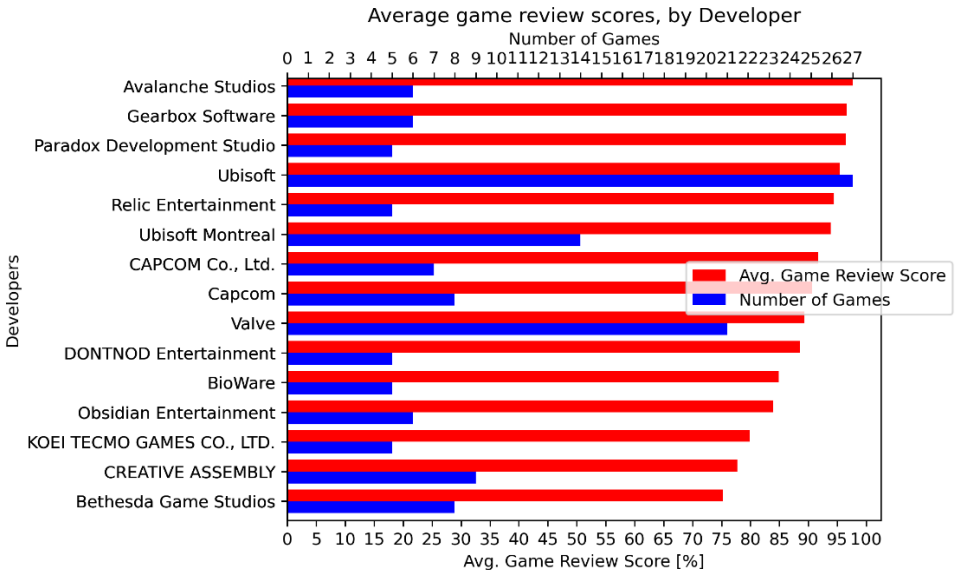
Davis, N. (2019, June 12). *Steam Store Games (Clean dataset)*. Kaggle. https://www.kaggle.com/nikdavis/steam-store-games?select=steam.csv.

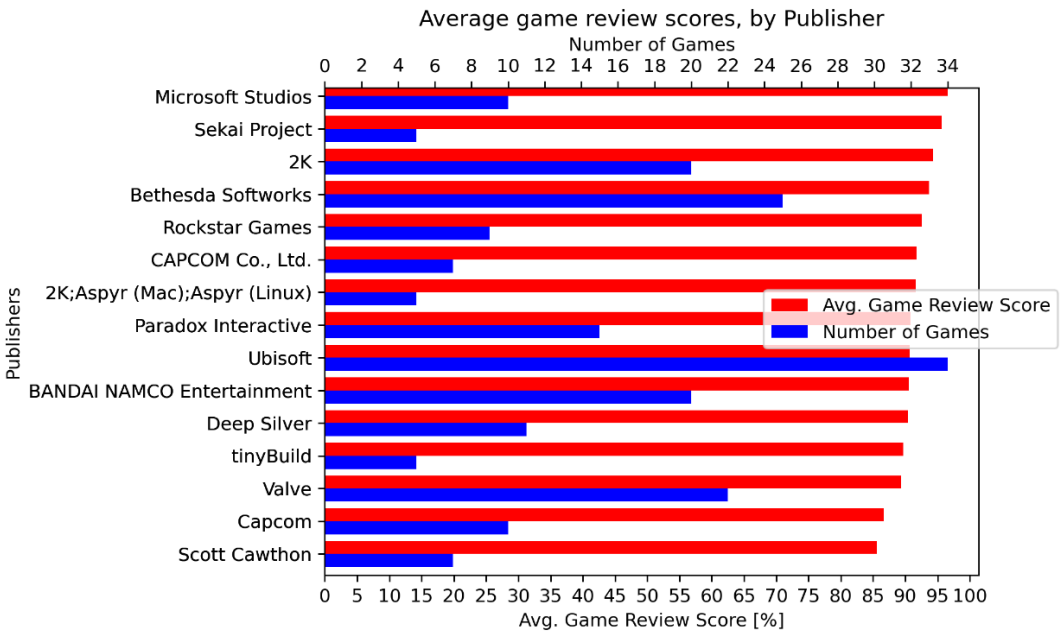
Project Deliverables

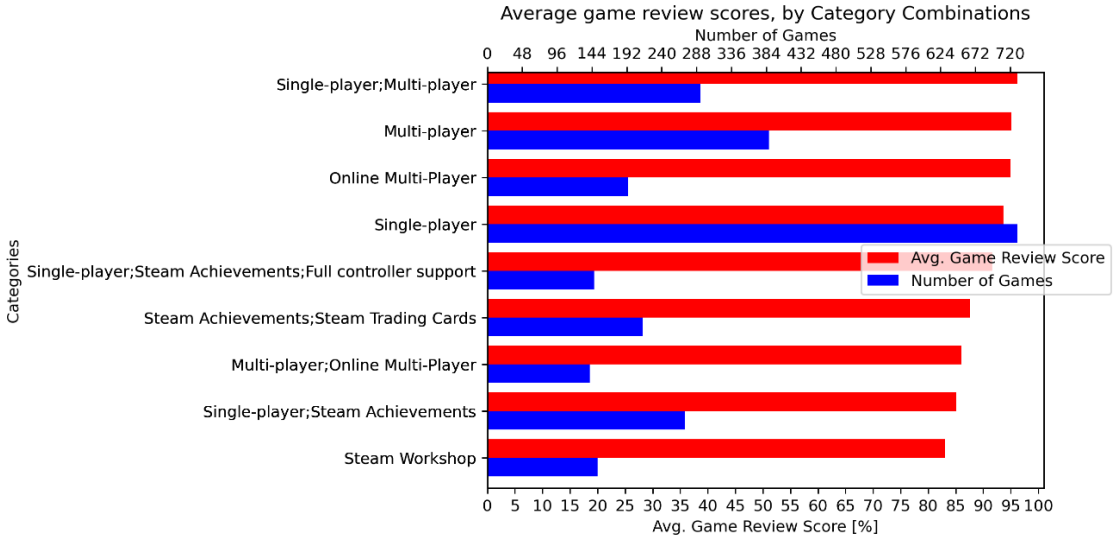
GitHub: https://github.com/crossfirev/SteamDatabaseAnalysis-CPE551

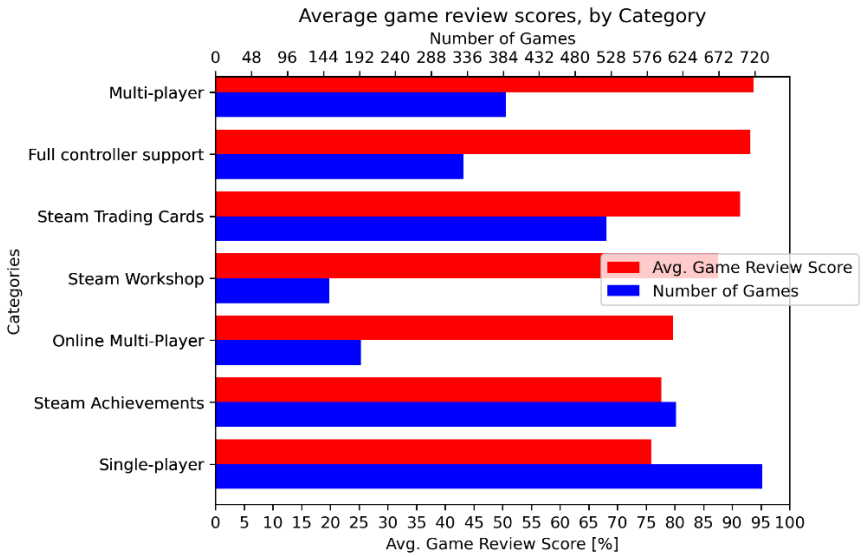
YouTube Demo: https://www.youtube.com/watch?v=pZ9TigDik6w

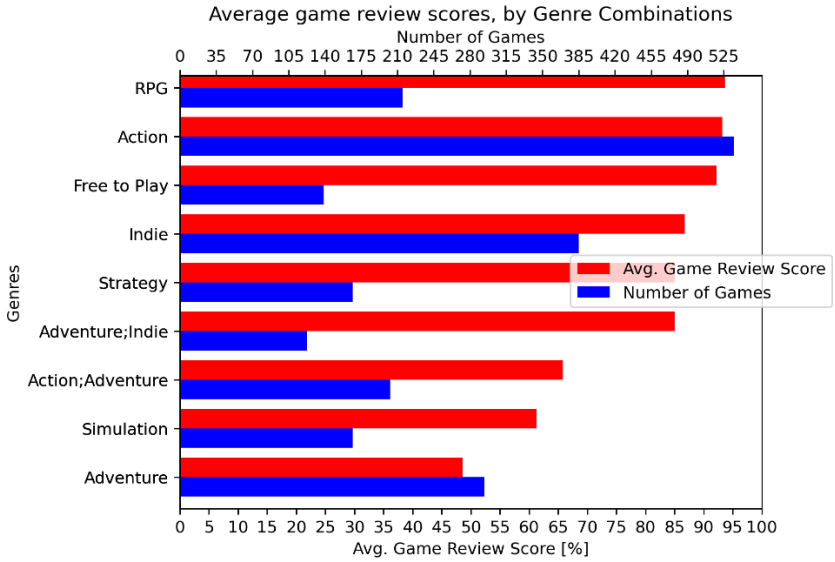
Example Figures

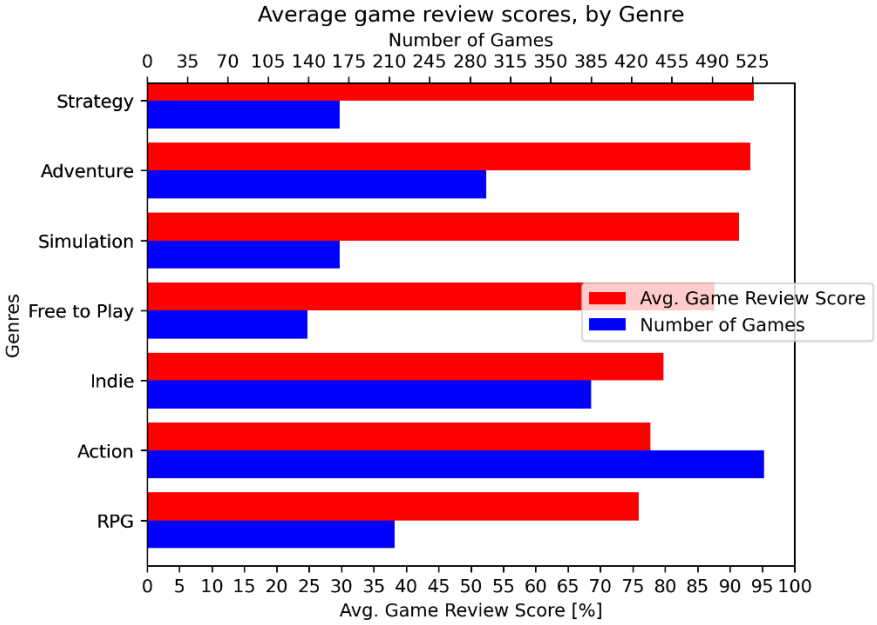







Average game review scores, by Category
Number of Games
0 48 96 144 192 240 288 336 384 432 480 528 576 624 672 720
Multi-player
Full controller support
Steam Trading Cards
Steam Workshop
Online Multi-Player
Steam Achievements
Single-player
Categories
Avg. Game Review Score
Number of Games
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Avg. Game Review Score [%]


Average game review scores, by Genre Combinations
Number of Games
0 35 70 105 140 175 210 245 280 315 350 385 420 455 490 525
RPG
Action
Free to Play
Indie
Strategy
Adventure;Indie
Action;Adventure
Simulation
Adventure
Genres
Avg. Game Review Score
Number of Games
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Avg. Game Review Score [%]


Average game review scores, by Genre
Number of Games
0 35 70 105 140 175 210 245 280 315 350 385 420 455 490 525
Strategy
Adventure
Simulation
Free to Play
Indie
Action
RPG
Genres
Avg. Game Review Score
Number of Games
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Avg. Game Review Score [%]