CS-4641-Steam

Introduction

With over 3 billion gamers worldwide and bringing in more revenue than all other forms of entertainment, video games have become one of the most popular forms of entertainment in the world (Arora, 2024). With more games being created now than ever before, it's become difficult to search for a new favorite game.

Literature Review

Classification analysis on games has been performed a number of times before. Xiaozhou Li (2020) performed exploratory factor analysis specifically on Steam games and tags, and Horppu et al. (2021) created a classification tool to predict the genre of a particular game based on description.

Problem

Steam, the largest PC game platform, has a own recommendation system where they display games similar to one a player owns. However, this system regularly displays popular games that avid users will already know about, and often are not very similar. Additionally, Steam has a categorical tagging system to label games, but it is still too restrictive and too difficult to search with. Knowing these issues, our group aims to create a clustering tool to group games into more niche categories.

Methods

Database

We plan to use a Kaggle database of all Steam games, which includes the game's title, description, Steam tags, price, release date, user rating, and user reviews. We will also use the website "How Long to Beat", which is a database for the length of a game.

Links:

- https://www.kaggle.com/datasets/antonkozyriev/game-recommendations-on-steam/data
- https://howlongtobeat.com/

Data Preprocessing

We implemented multiple proposed pre-processing methods. Firstly, we cleaned the dataset via removing irrelevant data points such as duplicate games, soundtracks, art books, and software. This required combing the dataset algorithmically as well as manually to find keywords that would indicate superfluous data. Furthermore, we removed unimportant dimensions that we did not expect to influence the quality of the recommendations from the prior dataset, such as discounts and augmented prices. The Kaggle dataset we were using was far from perfect for providing us relevant information as we began to analyze it. Therefore, we also added additional quantitative dimensions, such as playtime, to the dataset by integrating data from the website How Long to Beat. This required additional data-scraping through How Long to Beat's API. Additionally, we scrapped Steam itself to find tags and genres for additional features. Finally, as a large proportion of our data turned out to not have sufficient data, we deleted these points such as to not overgeneralize our data, and to cull our data set further as to not have too many values. Values assumed to be invalid or inaccurate were also cleaned and edited to be more accurate, such as the playtime of a game showing to take longer to complete the main story than completing all side-quests due to the user driven nature of How Long to Beat.

We then proceeded to encode the categorical variables as numerical values, and added scalers to certain feature columns to center the data and account for outliers. We also performed dimensionality reduction in order to choose the features that provided the most information. While we had spreadsheets with all tags and genres assigned to each game, the curse of dimensionality proved to be too much even when performing reduction, resulting in us removing this from our dataset.

Algorithms / Models

- **Naive Bayes:** We chose to implement Naive Bayes as it appeared to be a better model for recommendation systems than the ones we had originally proposed. We thought Naive Bayes would end up being faster, easier to implement, and more suited to the number of parameters we had. Naive Bayes is a supervised learning algorithm, meaning the data has already been sorted into groups. In this case, the games have already been classified by the label given to their average ratings on Steam. This made Naive Bayes an ideal model to implement at least for our project midpoint, as it would allow us to assess how viable the project was in the short term and what we would need to change without investing too much too early on.

- **GMM:** With GMM, we can perform model-based clustering in which we can group games together. Then, given a cluster of games that a user likes, we could then attempt to predict what other games that a user may like but does not have.
- **SVM:** The support vector machine can classify where a new data point would be placed, meaning we can give it a hypothetical game and find what games would be closely related to it.

Results and Discussions

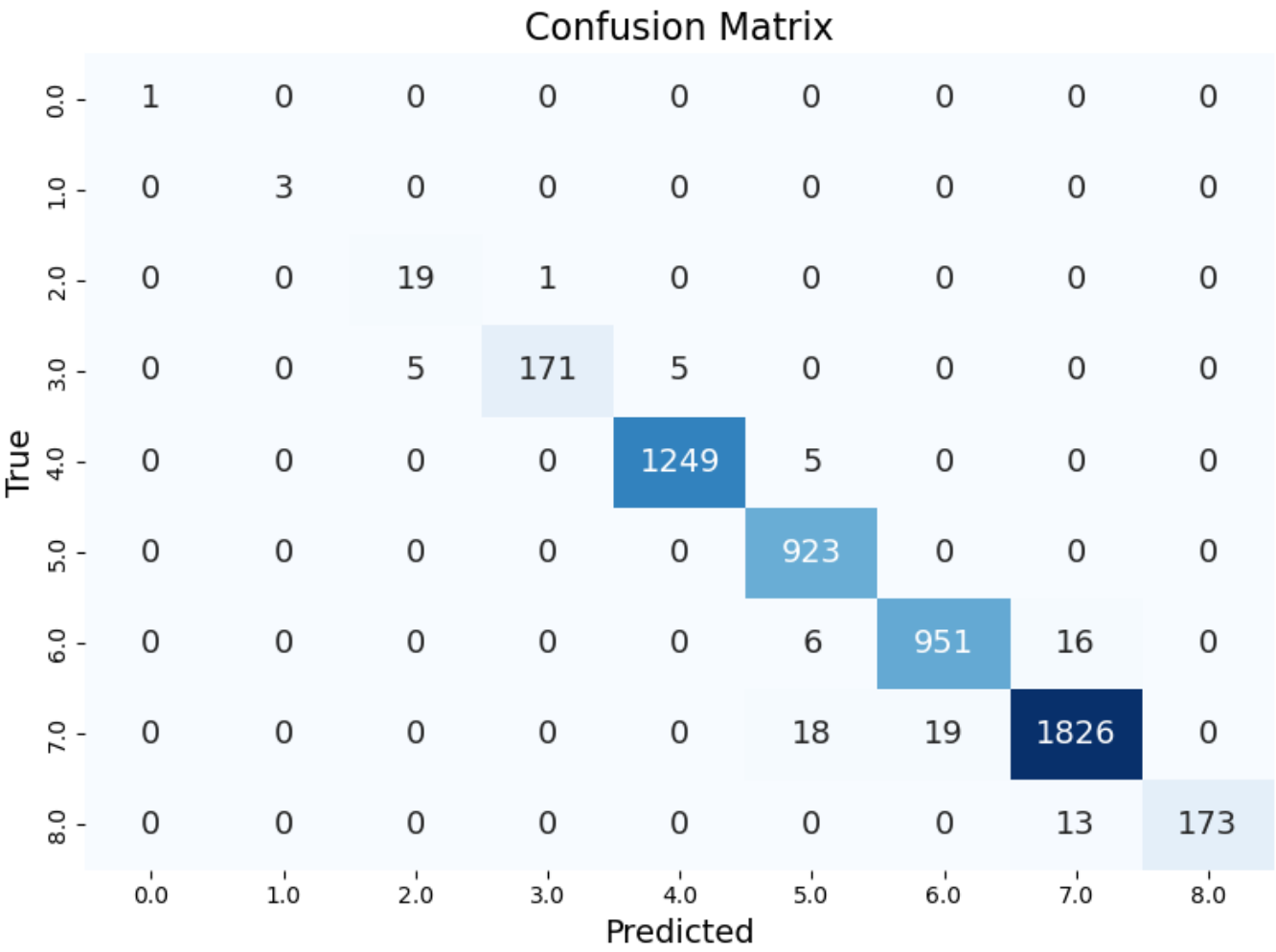
Chosen Quantitative Metrics

These analyses will yield the following metrics:

- **Accuracy** would measure how often our predicted clusters and actual clusters compare, allowing up to see how our algorithms fare
- **Cohen's Kappa** would measure the alignment between predicted clusters and actual game categories, allowing us to see how effective our clustering algorithm is compared to the already existing classification system used by Steam.
- The **V-measure score** will tell us how good our clustering process is, and complete clusters will allow for niche expansions on Steam's current categories.

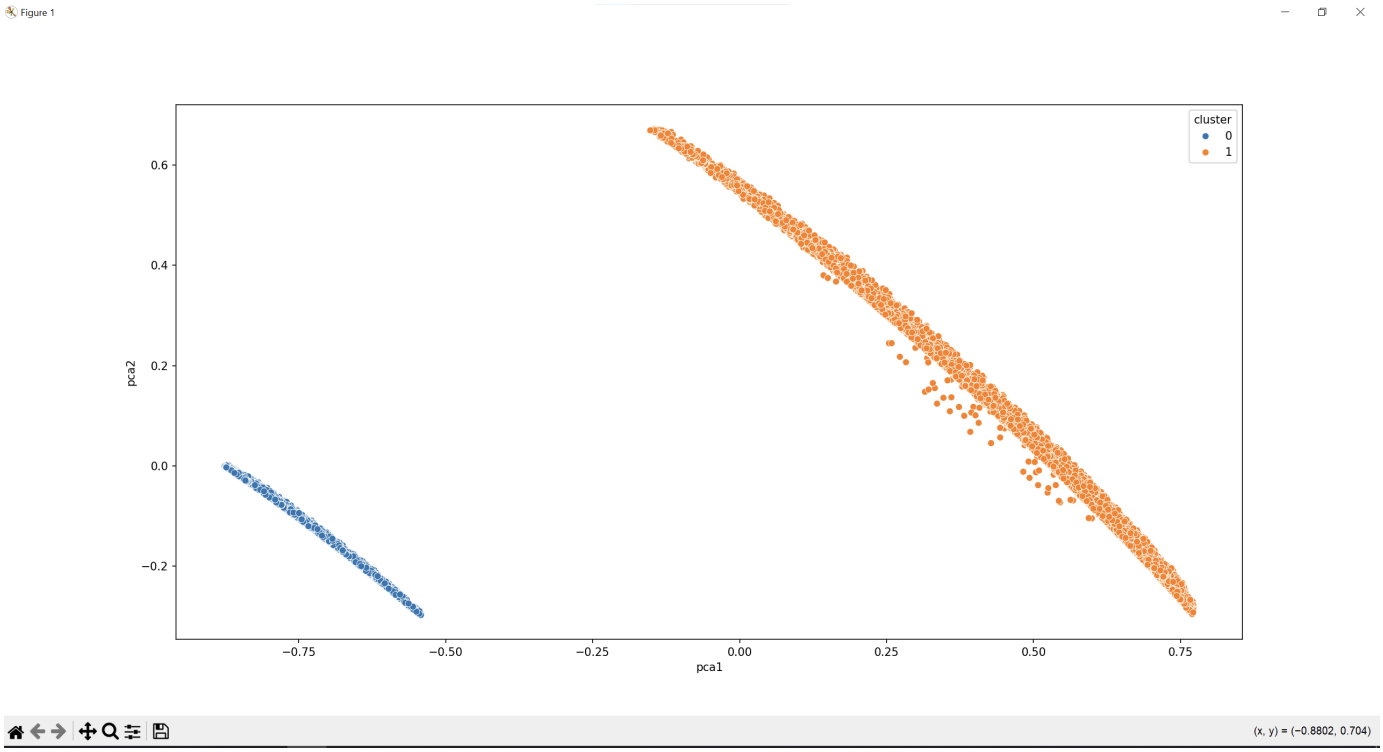
We expect to find clusters of games that will allow us to group existing games and theoretical games that are searched. We'll also identify what classifications are made to games that may not have proper classification prior.

Naive Bayes:



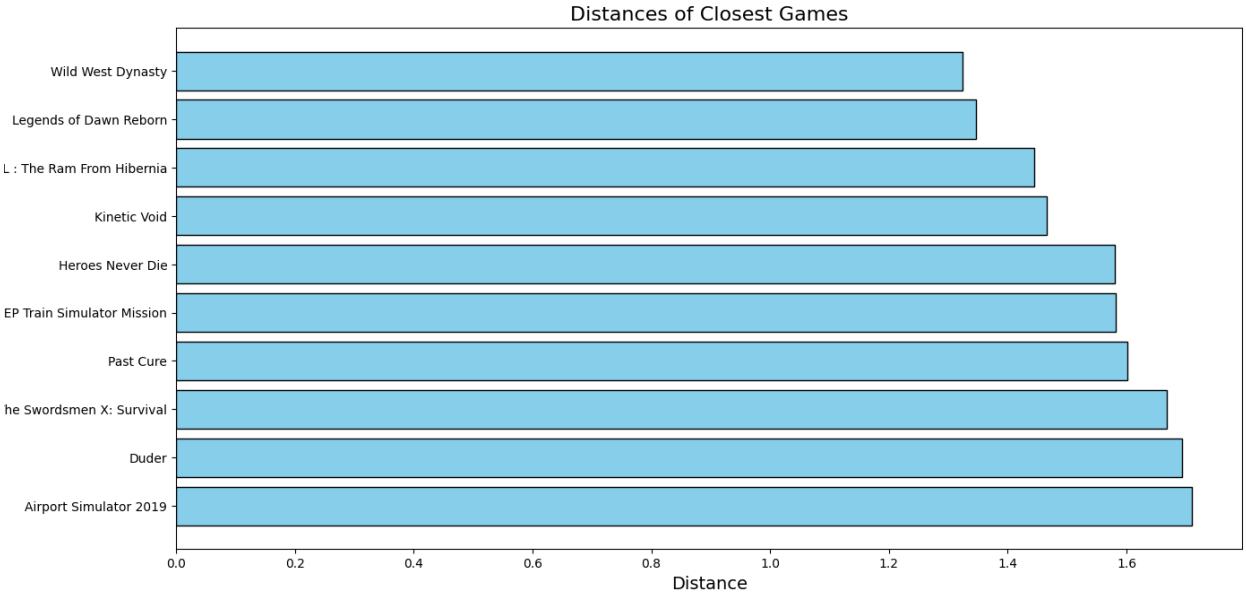
- **Quantitative Metrics:**
  - **Accuracy:** 98.37%
  - **Cohen Kappa:** 97.87%
  - **V-Measure:** 94.55%
- **Analysis of Algorithm / Model:** Our implementation worked fairly well, ending up with high scores in our quantitative measures indicating our model works well. We ended up with an accuracy of 87.87% for our training data and 88.62% for our testing data, indicating that there is no concern of underfitting or overfitting while working accurately. The high Cohen Kappa score serves as another good indication of the accuracy of our model as it agrees with the testing data set. Finally, our V-Measure Score indicates that our data is accurately clustering data points together.

GMM:



- **Quantitative Metrics:**
  - **Accuracy:** 98.43%
  - **Cohen Kappa:** 96.67%
  - **V-Measure:** 88.43%
- **Analysis of Algorithm / Model:** Our implementation of this model did not work well. We decided to set the number of clusters to two, with the two clusters being if we should recommend the game or not given a certain model. For evaluating the model based on quantitative metrics, we created a model based on our training data and one based on our testing data. We then used both models to predict the labels based on the testing data and compared the two based on our quantitative metrics. This had mixed results; while our V-Measure score indicates that our data is accurately clustering data points together, and our accuracy seems to indicate agreement between the two models, the Cohen Kappa score heavily changed depending on the training and testing data and how they were generated. This was likely due to improper handling of the various parameters and a lack of actual ground truths for classification and appropriate comparison.

SVM:



- **Quantitative Metrics:**
  - **Accuracy:** 100%
  - **Cohen Kappa:** 98.78%
  - **Average Distance:** 5.7188
- **Analysis:** Our implementation of SVM was both good and bad. Based on the decided upon features, the SVM implementation works well in predicting the quality of a game as well as finding similar games on both sides of the hyperplane. It implements hyperparameter tuning to help mitigate overfitting and ensure better generalization. The data preprocessing is done well in many areas, properly handling missing values, converting all labels into numerical values and scaling features. However, there are a few issues with our data preprocessing. Our choice of features is limited and flawed, not being able to provide an accurate enough predictor as could be possible with a greater range of features. While we properly handle the case of missing values, dropping data is possibly unnecessary as we could alternatively handle it through imputation. Finding a more dynamic feature set would greatly help this implementation of SVM. In addition to this, our quantitative metrics suffer from the problems with the feature set, and our implementation is extremely overfitted, resulting in near-perfect metrics across the board. Data leakage seems to be a problem in this implementation as well. The next steps in our SVM implementation would be to reconsider and greatly expand the feature set chosen, as that is where the majority of our issues reside.

Comparison

While Naive Bayes performed exceedingly well, our implementation of GMM and SVM left a lot to be desired. We were unable to get these models to accurately classify games such that good

recommendations could be provided given a player's previously played cluster of games. While both Naive Bayes and GMM are fast methods of attempting to classify data, SVM can be better at classifying the large datasets given the large amount of information we had to work with. We were unable to tune the parameters to be able to accomplish this, however. Similarly, Naive Bayes runs into an issue of assuming feature independence when the features of a game are not, though can be, so.

References

[1] X. Li, "Towards factor-oriented understanding of video game genres using exploratory factor analysis on Steam Game Tags," 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), vol. 3, pp. 207–213, Dec. 2020. doi:10.1109/pic50277.2020.9350753

[2] I. Horppu et al., Automatic Classification of Games using Support Vector Machine, 2021. doi:10.48550/arXiv.2105.05674

[3] K. Arora, "Council post: The Gaming Industry: A behemoth with unprecedented global reach," Forbes, https://www.forbes.com/councils/forbesagencycouncil/2023/11/17/the-gaming-industry-a-behemoth-with-unprecedented-global-reach/ (accessed Oct. 4, 2024).

Extra

Gantt Chart

https://docs.google.com/spreadsheets/d/1COhf-dEsvwDQb1MIrLNSjtyQFBnLniWaHcc7xhpuErE/edit?gid=1567635451#gid=1567635451

Contribution Table

Name	Proposal Contributions
Joseph Seo	GMM Implementation and write up
Jade Spooner	Preprocessing, Naive Bayes Implementation
Nick Unger	Presentation / Write Up
Jordan Kim	SVM Implementation and write up