Steam Marketplace Customer Sentiment Prediction

Matt Calcaterra
Mohammad Awad

Motivation

Our project aims to analyze data of games listed in the Steam Marketplace in order to develop a model which can predict the sentiment of customers, based on specific variables of a game. The intention is that this model can be used to estimate whether a game looking to come to market will be received positively by customers. Using Steam reviews as a source of data for NLP is a common occurrence on websites such as Kaggle. Our project differs from those in trying to link the sentiment of those analyses with variables which can be determined before the launch of a game, such as genre, developer, and languages. We hope to create a model which will allow developers and publishers to estimate the reception of their game before launching on the Steam Marketplace.

Data Sources

Our project used data from two primary sources: Steamspy API, and Steam Marketplace Review Scraping. The data for both of these sources was collected in a separate notebook from our main project, and stored in a .csv file for access during the project. Due to the size of both of our dataset, this allowed us to save a significant amount of time, and reduce demand on the datasource websites.

Steamspy API

The primary datasource for our project was the steamspy API, which can be called using the following url: https://steamspy.com/api.php

Steamspy API allows for multiple requests, however we were interested in gathering data on all games in the Steam Marketplace so we used the `all` endpoint. This returns data for all games in the Steam Marketplace which have owners data. 1000 entries are returned per page, so we used a loop to call 1000 entries, then proceed to the next page until all games had been returned.

We use the API to collect a dataset of all games in the Steam Marketplace, resulting in a dataframe with 63,968 rows, or 63,968 games. There were also 20 variables contained in the dataset. Our prediction model is based on 5 of these variables, however 18 of these variables were kept for analysis purposes.

The following were the variables in our dataset:

- **Appid -** The games ID
- **Name -** The games name
- **Developer -** Developer of the game
- **Publisher -** Publisher of the game
- **Score_rank -**
- **Positive -** Number of positive reviews indicated by Steam
- **Negative -** Number of negative reviews indicated by Steam
- **Userscore -**
- **Owners -** Range of owners of the game
- **Average_forever -** Average player playtime
- **Average_2weeks -** Average player playtime in the last two weeks
- **Median_forever -** Median player playtime
- **Median_2weeks -** Median player playtime in the last two weeks
- **Price -** Current price of game
- **Initialprice -** Initial price of game
- **Discount -** Current discount
- **Languages -** Languages indicated on Steam listing
- **Genre -** Genres indicated on Steam listing
- **Ccu -** Current Concurrent Users
- **Tags -** Tags given to the game by players

Since we want to predict how a new game coming to market will be received by the player, our model is built off of variables which will be known before a game releases. These variables include:

- Developer
- Publisher
- Price
- Languages
- Genre

Since the data was significantly large, the dataset was saved as a .csv for easier access during the project. This also eliminated the need to call the API repeatedly, which takes a lot of time.

Steam Marketplace Scraping

In order to acquire reviews for each game in the steam marketplace we scraped the reviews from the steam marketplace using the following url format:
`https://store.steampowered.com/appreviews/<appid>?json=1`
We have opted for the data to be returned in json format, which was then converted into a pandas dataframe. This was called for each appid in the Steamspy API dataframe, and 100 reviews were attempted to be collected for each game. If the game had less than 100 reviews, all the reviews were returned.

Since only 100 reviews were able to be viewed at a time, we used a moving cursor in case more than 100 reviews were wanted to be retrieved. The function to collect this data was based off code by Medium used Andrew Muller:
https://andrew-muller.medium.com/scraping-steam-user-reviews-9a43f9e38c92

The primary variables of interest from this data source consisted mostly of `review` and `appid`. Other variables were kept in the dataset for future analysis, however our project was only interested in the individual reviews for the game and the appid which they corresponded to.

Once the review endpoint had been called for 100 reviews for each appid, our resulting dataframe contained 1,336,828 rows. Due to this size (and time required to gather the data), the data was saved as a compressed .csv file for easier data access in the rest of the project.

Data Manipulation Methods

Our data manipulation followed 5 main steps: Natural Language Processing, Null Value Evaluation, Feature Engineering, Model Tuning and Selection, Analysis.

Natural Language Processing (Sentiment Analysis)

The first step in our data manipulation was the extraction of sentiment from each steam review. We used the dataset of steam user reviews we had previously assembled. Null values were dropped from the review dataframe to ensure that each row passed through NLP has a non-null review.

The first step in our sentiment analysis was to preprocess our text. We created a function which tokenized a review, removed the stop word, lemmatized, and then combined it back into a string. This function was then mapped to our dataframe to be applied to each review. We then created a second function which used NLTK SentimentIntensityAnalyzer to return sentiment scores for a review. This function was then applied to our dataframe on the preprocessed reviews.

It was important to preprocess our text, as each review followed differing formats, and creating the most consistent formatting of reviews would increase the accuracy of our sentiment analysis. Each review was tokenized using NLTK in order to allow our language model to recognize and edit each word as a token. Stop words were removed as they provide no meaningful information, and removing them increases the speed and accuracy of our sentiment analysis. Each token was then lemmatized, allowing us to create more consistency across reviews, by placing each word in the same form. The preprocessed text was then passed to the sentiment analyzer, which returned a positive, neutral, negative, and compound sentiment score for each review.

Due to the number of reviews, and intensity of the sentiment analysis, this process took a very long time to run (8 hours). As a result, we stored the sentiment in a csv, with only the appid for each review, and the sentiment analysis. This .csv was used for the rest of the analysis. Additionally, the sentiment analysis in the .csv was stored in a dictionary. For use in the remainder of our project we expanded each element of the dictionary into its own column, the data was grouped by appid taking the mean for each score, and then merged with our game data dataframe on the appid variable.

Addressing Null Values

The next step in our data manipulation was to address the null values in the dataset. Since our dataset is so large, we opted to reduce our dataframe based on the number of reviews first before handling any null values.

We first assessed the `positive` and `negative` variables in our dataset to determine if any games had null values. There were no null values for these variables, so they were added together to create a new variable called `n_reviews` which represented the number of reviews for the game. This variable was then used to limit our dataframe to games with 100 or more reviews. This was chosen due to the fact that games with a small number of reviews may skew our results. After this process our dataframe was reduced to 14,741 rows.

After reducing our dataframe we assessed the null values which remained. There were 9 variables which had null values. These variables and number of nulls were:

- Name - 1
- Developer - 50
- Publisher - 31
- Score_rank - 14624
- Price - 2
- Initialprice - 1
- Discount - 1
- Languages - 6
- Genre - 69

Name

Since there was only one null value for name we opted to search the game in the steam store using the appid. After finding the game corresponding to the appid, we made the assumption that this game was left null due to the use of an emoji in the game name. We imputed the name of the game based on our findings. For future analysis, it may prove better to drop these rows to integrate better into a pipeline. Additionally, this variable will not be used for our model so dropping the column as a whole may be an option as well.

Develop and Publisher

The developer and publisher columns of our dataset were handled in the same fashion. The missing values were imputed with the value “Unknown”. This allowed us to keep the values for the other variables in the dataset, while grouping them together. These values may have been null if they were independent projects, which did not come as the results of a developing or publishing company.

Score_rank

The score rank variables was dropped from the dataset. This appeared to be an outdated metric which was returned by the API. Since most of the entries in the dataset were null, we dropped the column entirely

Price, Discount, and Initialprice

Similar to developer and publisher, price, discount, and initialprice were handled together. We made the assumption that if a game was missing data for its price, the game was free. To confirm we looked at the genre listed for these games to determine if Free to Play was listed. Since the genre column for these games were also null, we set the price, discount, and initialprice to 0.

Languages

Since there were only 6 rows which had null values for languages, we chose to drop these rows from our dataset. It could have been possible to impute these games with the most common languages, however based on the data there was no way to reasonably determine which languages should be listed for the game.

Genre

There were 69 games which had no genre listed. We imputed these nulls with “No Genre”. Since genre may not be a necessary component to a store listing, we wanted to make sure that our data indicated which games had no genre listed.

Feature Engineering

After cleaning the null values from our dataset, we wanted to create some new features in our dataset in order to get a better understanding of the data. The `n_reviews` column had been previously created. This was then used to create three new columns. Percentage_positive was created by dividing the number of positive reviews by the total number of reviews for that game. A similar process was used to create percentage_negative. Percentage_positive was then divided by the percentage of negative reviews to create a new variable review_ratio.

We also created a new variable called n_languages, which indicated the number of languages listed for each game. Upon analysis, it appeared that some games had 29 languages included on their Steam listing. We made the assumption that the producers of these games were free to indicate which languages they wanted on the listing, without being limited by the actual

languages of the game. Because of this, we chose no to encode each language, but rather keep the number of languages to determine if including more languages on the listing increased sentiment. We also looked at the games which had a single language listed, and what language was listed. English was significantly the most popular language, as a result we also created a new column `english` which indicated whether each game had english listed as a language.

The next step of our analysis was to encode the genre for each game. The genre variables were a list of genres listed for the game. We created dummies for these columns in order to look at the distribution of genres. We found that some genres had very few games listed under them, so we dropped at genres which had less than 10 games. This process was later incorporated into our ML preprocessing pipeline.

The final step of this process was to create two new variables “major_producer” and “major_developer”. These are binary variables which indicate if each game is created by a major publisher or developer. We set the threshold for major publishers as those that have published more than 50 games, and major developers as those who have developed more than 25 games.

Model Tuning and Selection

The first step of our model creation process was to select the features and labels from our dataset. Since we want to predict the sentiment of customers for a game which is planned to be released, we chose features which would be available before the release of a game. These included:

- major_developer
- Major_publisher
- developer
- publisher
- genre
- price
- n_languages
- english

The desired dependent variable of prediction for our analysis was compound_sent. This variable ranges on a scale from -1 to 1 with -1 being negative sentiment and 1 being positive sentiment.

Due to the potential for genres to be different between the test/train split, we encoded our genres before splitting the data. The genres were encoded in the same fashion as earlier where MultiLabelBinarizer() was used to encode the genres.

We started by creating a test train split of our dataset, with 80% of the data being assigned to the training set, and 20% of the data being assigned to the test set.

A preprocessing pipeline was established where we used StandardScalar() to scale our numeric variables.OneHotEncoder() was used to encode developer and publisher.

Three regressors were then chosen for our models. We chose RandomForestRegressor, LinearRegression, and SVR as these were three common models which would fit our needs.

After our models had been selected we created a grid of parameters, and used a GridSearchCV to identify the best parameters for each regressor. These parameters were then applied to each model, and the mean square error was calculated for each. Additionally, the three models were also combined into a VotingRegressor and the mean square error was calculated for that as well.

Analysis

Our analysis of the Steam games dataset utilized accurate data integration and manipulation to prepare for our predictive modeling. We examined various attributes, particularly focusing on reviews, sentiment scores, genre distribution, and language support.

Review Analysis: Figure 1 | Figure 2
We began by calculating the landscape of player feedback. New columns - 'review_ratio', 'percentage_positive', and 'percentage_negative' - were calculated by normalizing the counts of positive and negative reviews against the total number of reviews. The review ratio was derived from the proportion of positive and negative reviews. Our findings revealed a predominance of positive feedback with the frequency of positive reviews clustering between 80% to 100% range, and the average positive review score of approximately 78.5%. Negative reviews were found to be less frequent, with a peak close to 0%, and averaging approximately 21.4%.

Normalization of Data: Figure 3 | Figure 4
To normalize the data, we took the log of review ratios and number of reviews. We then generated histograms to visualize the distribution. This was completed by using the NumPy log function on 'review_ratio' and 'n_reviews' columns. The logarithmic transformation presented a normal distribution for review ratios, confirming the data's suitability for further analysis. The distribution of the number of reviews highlighted that most games attract a small number of reviews, with only a few exceptions receiving a large amount of reviews. Which is a common trend in consumer products.

Sentiment Correlation: Figure 5 | Figure 6
Through hexbin plots, we identified the relationship between sentiment and percentage of reviews. This was completed by using the Seaborn joint plot, setting the x to 'percentage_postive' and y to 'pos_sent' and inversely for the negative plot. We confirmed a correlation between the volume of positive reviews and high positive sentiment score. A similar, inverse correlation was found between negative reviews, indicating that items with fewer reviews typically had lower negative sentiment scores.

Genre Trends: Figure 7
Our exploration into genre trends showed a large preference for Indie games, surpassing other genres by a significant margin. This was done by expanding the ‘genre’ column and creating value counts for the expanded column and plotting using the counts as the x and genre count index as the y. This may be due to a diverse range of content falling under this category.

Sentiment Score Distribution: Figure 8
We further analyzed the sentiment scores, aligning them with the observed review distributions. By plotting a histogram of the ‘compound_sent’ column we noticed that the majority of sentiment scores were positive, mirroring the trend noted in the positive review analysis and validating the sentiment analysis’s accuracy in reflecting the dataset.

Language Support: Figure 9
Finally, we analyzed the language support across games. By plotting a histogram of the ‘n_languages’ column we noticed that the majority of games support a limited number of languages, with a significant decrease in games offering multilingual support. This emphasizes the resource-intensive nature of localization and its rarity in the dataset.

Top Games by Positive Reviews: Figure 10
An additional graph provided insights into player reception at the individual game level. This gave us an insight into the possibilities and potential reasoning behind why they were received well to the player base. Noticing any potential trends with the currently highest rated games.

The final feature selection for our predictive model is informed by these insights, including major_developer , Major_publisher, developer, publisher, genre, price, n_languages and english. These features are anticipated to provide a strong foundation for accurate predictions regarding game success on the Steam platform.

Based on the four models we chose for testing, the VotingRegressor had the lowest RMSE at 0.0325. The best regressor which was not the VotingRegressor was our Support Vector Machine model which had a RMSE of 0.033. Since our model is predicting compound sentiment, which is on a scale from -1 to 1, we find that a RMSE of 0.0325 is a low enough deviation from average to effectively predict the sentiment of games.

Further analysis should be conducted to determine whether customer review sentiment is a good metric for predicting game sales and performance. Pulling data about game sales, and the amount of money generated from games sales should be done and combined with the current datasets to investigate trends and findings regarding the correlation between review sentiment and sales performance.

Visualizations

Figure 1

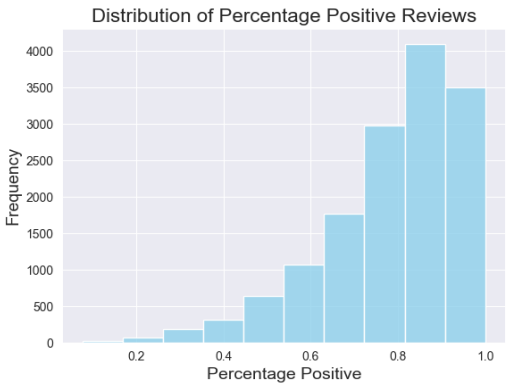


Figure 2

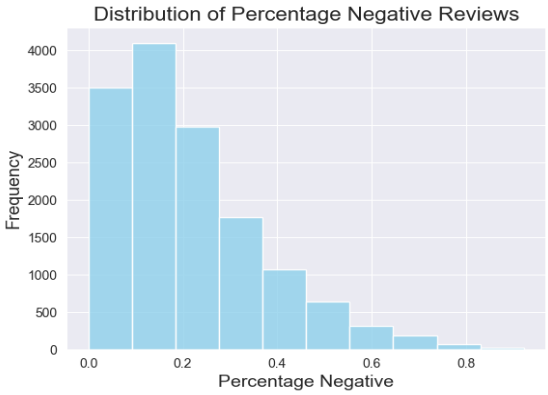


Figure 3

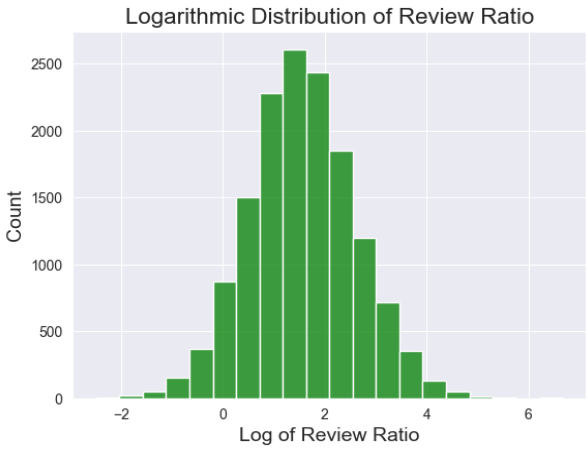


Figure 4

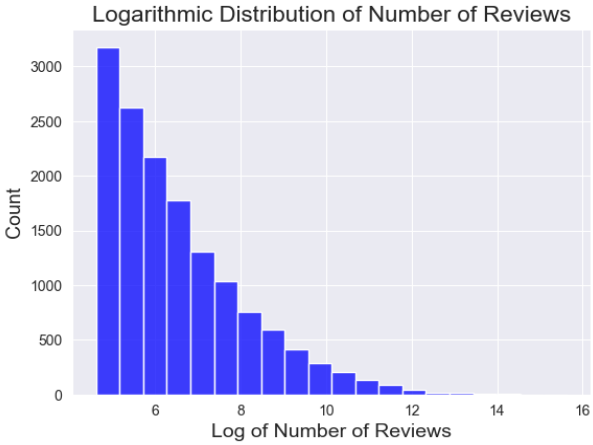


Figure 5

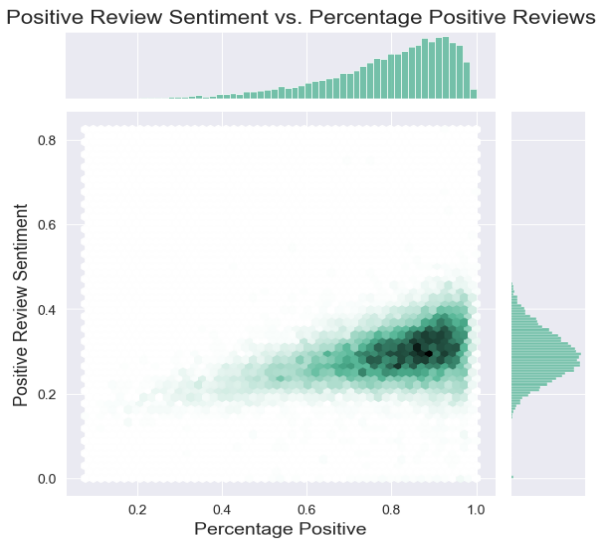


Figure 6

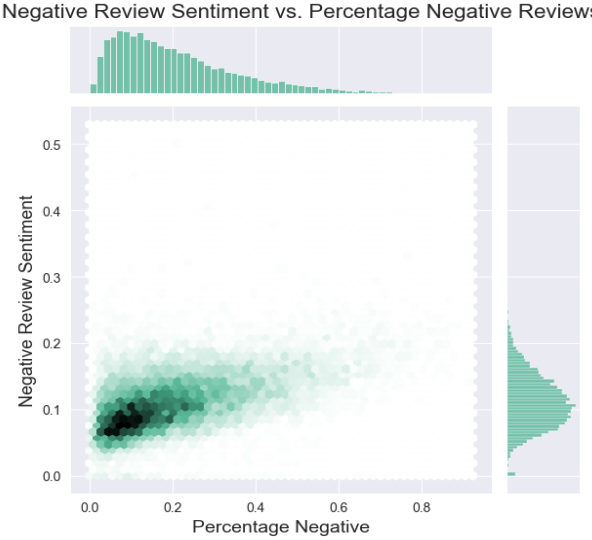


Figure 7

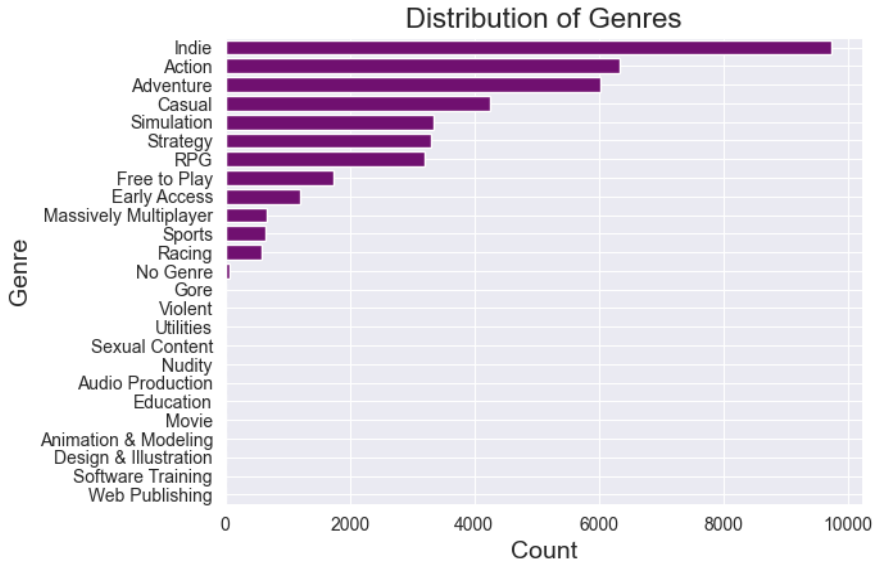


Figure 8

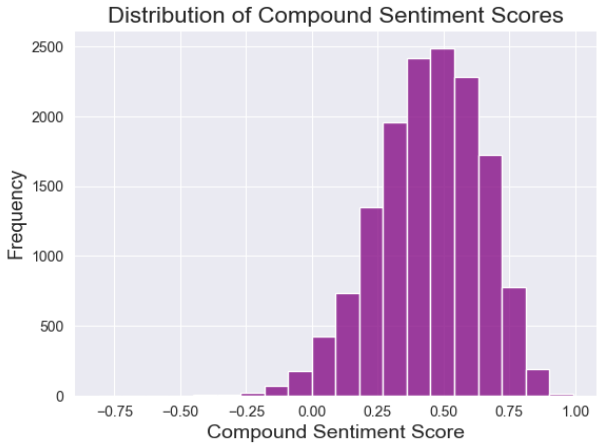


Figure 9

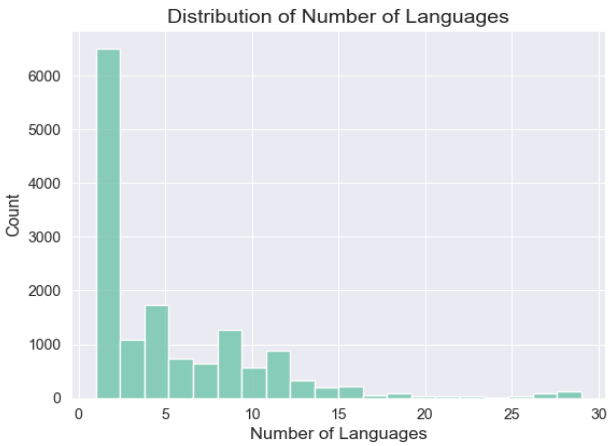
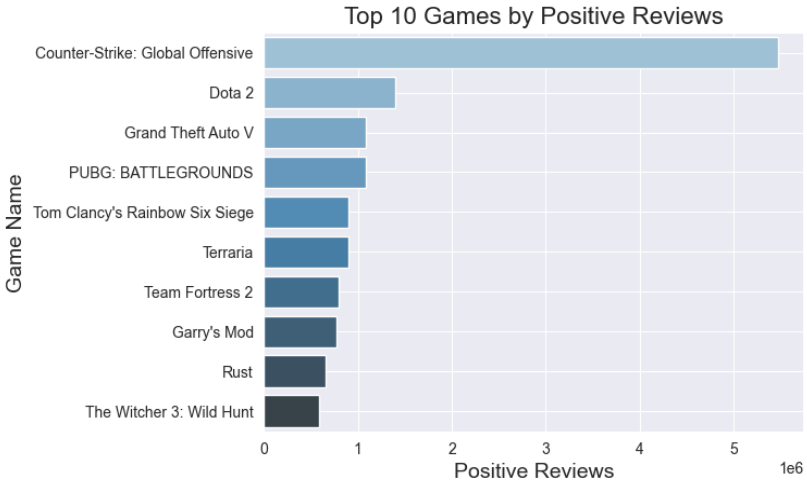


Figure 10



Statement of Work

After a discussion of what work was needed to be completed, the responsibilities were divided among partners. The division of work was as follows:

Matt Calcaterra	**Mohamma Awad**
Data Collection	Exploratory Data Analysis
Feature Engineering	Initial Model Creation
Model Refinement and Selection	Data Cleaning
Data Methods and Manipulation Report	Analysis and Visualization Report

The collaboration of partners went smoothly, with each part being reviewed by the other partner at varying stages of the project. The divided approach allowed for flexibility of working on the project, which proved to be beneficial as there was often time and availability conflict. We also meet semi-regularly on virtual calls, in order to discuss our progress, where we wanted to go next, and answer any questions we may have had.

For collaboration in the future, we would like to keep the same approach of dividing the work in stages, however a few improvements could be made to make the process smoother overall. The first improvement for future collaboration would be to lay out a more detailed roadmap prior to starting work on the project. This would include timelines and deadlines, which would help us stay on track, and plan out our individual parts on the project better. Additionally, adding in more regular meetings would help to increase understanding of how the project is developing, and increase the collaboration on the project.

Acknowledgments

Github Copilot was used to assist with the code throughout this assignment.

Additionally, Data collection code was drew inspiration from Medium user Andrew Muller, and his post:
https://andrew-muller.medium.com/scraping-steam-user-reviews-9a43f9e38c92

Further work should be conducted in order to investigate whether review sentiment is a good predictor of the sales performance for games in the Steam Marketplace.