Steam: Analyzing and Predicting Reviews Final Report

Reed Paskvan (rpaskvan@umich.edu)**, Antonio Montalto** (tonymont@umich.edu)**, Jeffery Jones** (jeffejo@umich.edu)

Introduction

Steam is a video game storefront that connects publishers and developers with their audience. The store enables users to leave public reviews, containing a rich mix of subjective and objective criticism of a game. "'Now I can create a niche game and have it reviewed by people who care about those games. When I'm developing, I can scan those reviews to get an aggregate idea of things that players are struggling with and address them'"(Lane, 2018). The language used in these reviews give valuable insight and feedback to the game developer. Reviews can provide information about bugs within the game, such as graphical, audio or logical issues. Important aspects that a game developer couldn't catch in the quality testing process. Often a review can inform of a game's poor direction or gameplay design as Phillip Bak stated, "'At its heart the Steam review is the most raw unfiltered feedback'" (Lane, 2018). As a novel tool of criticism that inspires and informs game developers, analyzing Steam reviews is crucial. Unfortunately, as the number of Steam reviews for a single game grows, sometimes over 2 million as observed with Dota 2, it becomes difficult to summarize every review. Gardner showed how Steam reviews become an issue of scale, "'In AAA, you're all but guaranteed to get your game in front of millions of eyes…You're only ever going to be able to understand a minuscule fraction of the people who played your game.'"(Lane, 2018) Utilizing machine learning techniques should help developers retrieve a richer representation of their game reviews at scale. As a ratio of positive scores, does little to inform the game developer of nuanced public opinion. The field of topic modeling and sentiment analysis are ideal for understanding user reviews on Steam.

Each individual review contributes their outcome to an overall community score that is prominent in Steam's UI. This review score can potentially draw buyers in or deter them. The impact is apparent considering the sheer number of transactions, averaging over one million daily, on the storefront (Steam Market Report Q3 : Strongest Ever Q3 Performance, 2023). As the algorithm pushes higher rated games on the storefront, following a power law in sales. Game developers need to increase a game's public review score, to increase the likelihood of sales. A more robust and nuanced understanding of their game's reviews can help direct their development efforts and improve their product to increase revenue.

The team has direct domain knowledge with the Steam platform. As customers of many of the game developers we explored throughout the project, we've left reviews. Viewing other player feedback over the last decade and internet lexicon evolved has inspired our hypothesis and data questions. One member of our team aspires to be an indie game developer, giving relevance to the results of our research.

Word2Vec for word association, Latent Semantic Analysis for feature extraction, and Latent Dirichlet Allocation for topic modeling are utilized for unsupervised learning methods. Anqi Zhang experimented with similar methods, as noted in the related works, but our models are trained on current scraped data with a broader generalization across all genres. The modeled topics identify important terms that correlate to positive or negative review sentiment.

Logistic Regression, Multinomial Naive Bayes, Support Vector Machines, and Random Forests were applied for supervised learning methods. These models covered three model families, probabilistic, non-probabilistic, and tree-based. The Random Forest performed the best with default parameters, however a strong machine and more intensive testing, Logistic Regression could outperform the other models. Accuracy, precision, and recall were relatively high and robust, however a limited number of features may have lowered the ceiling of all the models. Overall, a model was developed that successfully generalized and predicted the sentiment of Steam reviews. Zhen Suo's work had similar results but our models benefited from additional features that boosted model performance, instead of reducing noise with text preprocessing.

Data Source

The Steam Web API, the official Valve maintained endpoint is our data source. To access the API, urls must be built with the correct parameters for a HTTP request to Valve's servers. The official documentation states that a maximum of 100,000 calls are allowed a day.(Valve Corporation, 2010) During testing, our team discovered that Valve likely projects your current call rate or bucket size during an undisclosed length of time. At the onset of development, a max of 60 calls were made before a "rate limited" response from the server. Testing multiple rates, a safe value of one call every five seconds was utilized. With a storefront of over 90,000 items and some items having over two million reviews, the scale and limitations of our data source greatly influenced the design of the scraper and sampling to collect our dataset.

The first step was the ISteamApps interface and the call to retrieve all public apps on steam, returned as a JSON response.(Valve Corporation, n.d.-a) The game list was nearly 100,000 tuples, each consisting of a unique app id and name. This list contained some empty name fields, which cannot be queried for further details. These empties were filtered out, along with titles including the name "Demo" or "playtest". Demos possess a unique app id, but user reviews may only be left on the parent app page by users. Any HTTP request made for the details or reviews of a demo app, would result in a failure code. With a finite amount of calls per day, any meaningless calls must be prevented. The filtering of demos does remove a potential feature for our dataset. Games with a free playable experience may influence a user's review. The cleaned list of app ids and names were sampled to limit the data set size. Two sample sets were created, a random sample and the most popular 250 games on Steam. Often a sample could not finish processing before reaching the rate limit, a data structure was developed to keep track of progress. The structure could pickle and be loaded into a new class instance and resume the following day.

For each game app id in the sample, one HTTP request was made to ISteamApps with the parameters set for that specific app id. This returned a JSON filled with most of the game's information, that a user would find on a specific steam page, such as developer info, trailers, hardware requirements.(Valve Corporation, n.d.-c) For our review analysis, the variables of genres, release date, price were the most important features. The details call was made 250 times on average for both sample creation strategies. There was no distinction for games released during a certain time period. The initial list of tuples from the previous endpoint did not return a timestamp. A game's release date wouldn't be known until the sequential call for a specific app. Concerned with rate limits, we chose not to exclude a game based time of release. This call provided the number of reviews for that game. Often games would have zero reviews and the data structure marked this app id to avoid reprocessing and further HTTP calls. A non-reviewed game was not included in the joined dataframe.

With each app and its details, a series of store HTTP requests were performed to retrieve that game's reviews. A mostly equal number of positive and negative reviews were requested to create a balanced sample. These reviews were retrieved by the most recent in the stream. Additionally, another sample with reviews sorted by most helpful was requested as a measure against sarcasm. This “helpful” rating is a feature of each review, where Steam users could rate a specific review as particularly helpful or funny.(Valve Corporation, n.d.-b) More helpful reviews were expected to be less sarcastic in content. Unfortunately, the most helpful reviews led to imbalanced classes. Each request could only retrieve a maximum of 100 reviews and to prevent over-representation from a single game this was the self-imposed limit to prevent excessive calls. Balanced classes were created with two calls of 50 most recent positive and negative recommendations.

To ensure our dataset was computationally efficient we explored several different storage and sampling strategies. Games such as *Dota 2* would have over two million reviews, and the upper limit of 100 reviews per game prevented over-saturation in our corpus. To enable faster model training, the corpus was limited to 10,000 reviews. The scraper pulled far more reviews, close to 25,000. Reservoir sampling was developed to ensure our sample was representative of every game in our data structure.

Related Work

Bais, Odek and Ou studied sentiment analysis on Steam reviews. Expanding upon an existing study, they tagged parts of speech to determine semantic orientation by creating multiple two word phrases, satisfying a pattern of adjectives, adverbs and nouns. This would identify phrases such as ,”great game” or “absolutely terrifying” (Bais et al., n.d., p.3). Phrases would be utilized for the calculation of pointwise mutual information between phrases and the training lexicon. Our project improves upon their work, where html pages were scrapped, our team instead utilizes the official steam web api to retrieve the most helpful reviews. This will reduce the amount of sarcasm in the training data.

Wagh utilizes a pre-made dataset of merging reviews and game details, to train traditional machine learning models. A particularly interesting note is a computed feature, the character length of each review.(Wagh, 2020) Wagh searches for a pattern between the review length and outcome of a review, utilizing common embedding strategies such as word counts and term frequency-inverse document frequency, TF-IDF. Our project expands on this strategy with token stemming from data cleaning to improve generalization. Additionally, our data is de-labeled, to avoid bias against certain game titles.

Anqi Zhang chose to segment 21 games into three separate datasets of purely numerical features, keyword features, and a combined set. Both TF-IDF and Latent Dirichlet Allocation are utilized to vectorize review text. Zhang evaluates these different embeddings, modeled with several naives bayes classifiers and a multiple layer perceptron regression.(Zhang, 2022) The overall goal of Zhang’s paper is to analyze the importance of educational games on Steam. Our analysis differs with live scraped game data, with no filtering as to the genre of game. The models should generalize for a more nominal steam review.

Zhen Zuo utilized naive bayes and decision tree classifiers for sentiment analysis. The data was collected from SteamDB and SteamSpy, third party sites, to pull extra features such as daily player counts with an open source scraper. Notable is their extensive data-preprocessing stage.(Zuo, 2018) Zuo removed special characters, transformed text to lowercase, removed stop words, stemmed, and filtered out most frequent and infrequent words. Additionally, Zuo removed short reviews to prevent models from learning noise. Our data efforts instead pull directly from the official Steam Web API as we explore the difference of language between different genres.

Feature Engineering

The first step in the feature engineering process was to load raw data into a visualization notebook. Each feature was explored using basic visualization techniques (box plots, histograms, word clouds, etc.). This helped observe class imbalances, data types, and to understand the shape of our data. The statistics of our data was as expected, with a balanced outcome variable, and feature distributions heavily skewed to the right. We included outliers, as they could provide valuable insight into each data point. Once initial exploration was completed, the data was then loaded into a preprocessing notebook.

Each key variable, included as a model feature for both supervised and unsupervised methods, were cleaned. Columns that represented similar data, or were unnecessary for the completion of the models were dropped. For example, the feature “free”, a boolean value representing if the game was free or not, was dropped because the “price” feature held zeros indicating a game was free. Features represented as boolean values were converted to 1 and 0 to represent “True” or “False”, and all columns that dealt in values represented in minutes were converted to hours for interpretability.

The variables “price”, “review”, and “genre” needed more complex steps to transform the data into usable features. The price column was initially loaded in as a string denoting the currency and price paid. Around 9,200 of the 10,000 rows were either Canadian or American dollars. For simplicity and time, all other types of currency were dropped from the dataset. The currency symbols were removed from the string, and those values were converted into floats. For the genre column, we were given a list of dictionaries containing the genre ID, and the title of the genre. A small helper function was created to pull out the genre title from each row, and one-hot encodings were created for each genre that was identified in the dataset. While these columns were not ultimately used in training the supervised models, the unsupervised methods relied on looking at per-genre word choices. Finally, the “review” column was prepared for semantic analysis. The raw data was represented as a string containing the review left by the user (See Appendix A for Feature Table). To keep the preprocessing as simple as possible, and to maintain flexibility further down the pipeline, the only steps taken to clean this column were removing stop words using the nltk library.

Supervised Learning

Methods

For supervised models, we loaded in preprocessed data that was completed in previous steps of the pipeline. All features besides text data, were ready to be included in the model training. While stopwords had been removed, a few extra considerations were made before the data could be split for training and testing. First, we opted to not tokenize the data after removing stopwords, as a high rate of jargon and gaming-specific terms or phrases could lead to a high proportion of out-of-vocabulary words. Second, choosing one of two main vectorizing options: TF-IDF and CountVectorization. Initial exploratory model analysis showed that TF-IDF vectorization outperformed CountVectorization, expectedly, later evaluations of models were all completed using TF-IDF vectorization. After confirming the vectorization of the text, extra features were appended to the vectors, and vectorization hyperparameters were adjusted. A cell was used to run through various combinations of n-gram length, max document frequency, and minimum document frequency. In the end, we found that using (1,2) ngrams, with 0.002

minimum document frequency, and 0.99 maximum document frequency performed the best. Around two thirds of the reviews were below 25 words, with around half being below twelve words. Because of the brevity of the reviews, the removal of too many words due to overall frequency could have been preventing the models from learning information.

Four models were chosen for the supervised learning section: Logistic Regression, Multinomial Naive Bayes, Random Forest, and SVM. Due to the variability and uncertainty of text data, we chose not to use an instance-based model, and opted for two probabilistic models (Logistic Regression and Multinomial Naive Bayes). For increased robustness we included a tree-based (Random Forest) and non-probabilistic (SVM) model. Initial runs of the models used 80/20 train test split, leading to the highest performing model of Random Forest. Model families were compared by performing 5 fold cross-validation on each model.

<u>Evaluation</u>

Each model was evaluated using accuracy, precision, recall, and F1 score. We felt comfortable using accuracy as a key evaluation metric, as our dataset was fairly balanced for the outcome variable. Precision, recall, and F1 score allowed us to identify areas in which the model excelled or failed. The inclusion of the precision and recall metrics ensured that the models were not biased towards a specific outcome label, and had an appropriate balance in performance. The F1 score indicated the harmonic performance of these two metrics, and was tested on both balanced and imbalanced classes for our models. It was found that the F1 score stayed relatively stable across these datasets. Below is a table showing the average results of the best model for each family:

Table 1: Scores based on model and evaluation metric

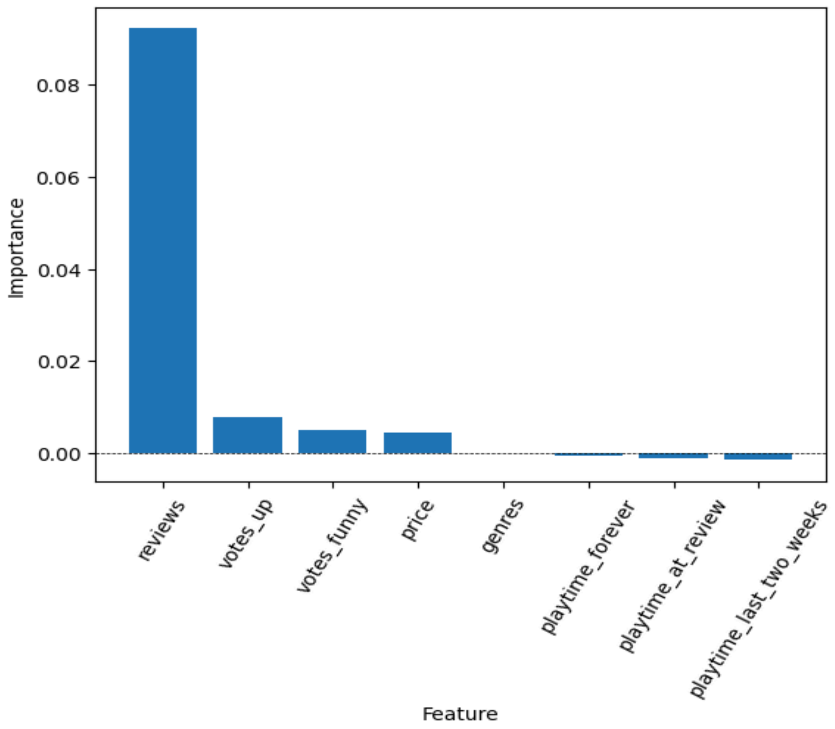
	Accuracy		Precision		Recall		F1	
type	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Model								
RandomForest	0.803	0.027	0.779	0.033	0.791	0.048	0.784	0.037
SVM	0.795	0.02	0.81	0.05	0.725	0.018	0.764	0.021
MultinomialNB	0.559	0.053	0.532	0.05	0.596	0.156	0.543	0.069
LogisticReg	0.607	0.028	0.596	0.058	0.477	0.087	0.522	0.047

It should be noted in table 1, Logistic Regression performed poorly, as the default max iterations were too low for the solver to converge. When max iterations were increased, this model was the best performing. It's been kept in its default state in order to preserve true comparison between all of the models in their default state. It did not outperform the other models to a degree that warranted the increased run time it added to continue evaluating Logistic Regression as the primary chosen model. With max iterations set to 1500, Logistic Regression had an average accuracy of 0.8187 while default Random Forest had an average accuracy of 0.8026.

For feature importance, all included features were taken out one at a time to evaluate the impact they had on the Random Forest Classifier's accuracy. Eight features, including text reviews, were used to train the model. The "genre" feature was represented as one-hot encodings. To develop a baseline, all of the one-hot encoded columns were excluded from the training data. Expectedly, text reviews hold the largest feature importance, reducing the model performance by ~9.2% when removed. The second highest feature, "votes_up", which represents the amount of upvotes a review received, only reduced model performance by

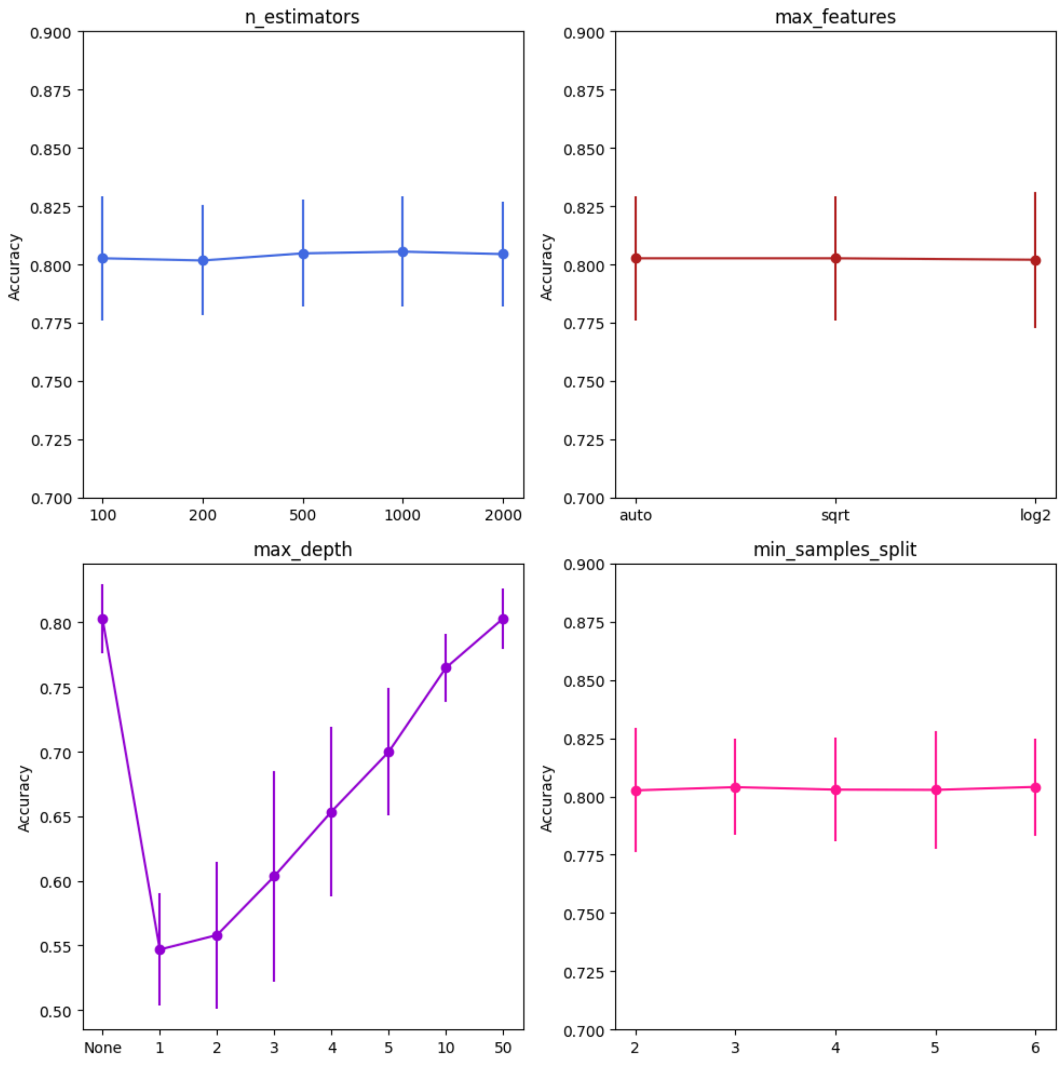
~0.7% when removed. It was expected the reviews would hold the most importance for model performance, the magnitude of this importance and the magnitude of the other positive features were smaller than anticipated. The “genre” feature held close to zero importance. This was unexpected, but may be due to many genres overlapping (i.e. an action game can also be a simulator and adventure game). Lastly, three features led to a decrease in model performance. All three of these features dealt with the amount of hours players had played the game at different points in time. This was the most unexpected outcome, as early exploration of model performance, the addition of these features boosted most of the models’ evaluation metrics. This may be unique to the performance of the Random Forest Classifier, or could be an issue unearthed by using k-folds cross validation that would need to be further investigated in future iterations. Nevertheless, each could be considered negligible amounts of added error, and in other models would most likely show reductions in error.

Figure 1: Feature Importance



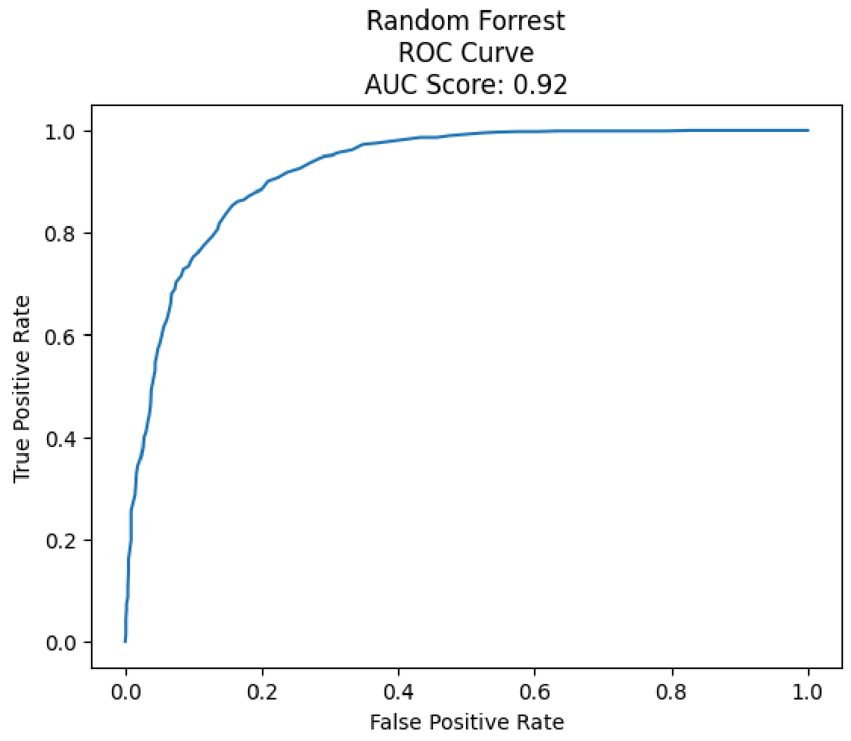
To perform sensitivity analysis on our Random Forest Classifier, a form of grid search was executed where in each iteration one hyperparameter was adjusted over a 5 fold cross validation set, and the accuracy scores were averaged. From this, it’s shown that many common hyperparameters of the Random Forest Classifier don’t have a strong impact on the model’s performance for accuracy. The only exception being “max_depth”. This hyperparameter controls the maximum depth of the trees in the forest, and when the trees are too shallow they are unable to properly learn the characteristics of the outcome variables, leading to an accuracy of around 54%, which is hardly better than random chance. However, as the depth of the trees increases, the accuracy rises again. While it may be a bit unexpected that the other hyperparameters have such little effect on the accuracy of the model, it was anticipated that “max_depth” would have the largest impact. We observed a moderately sized standard deviation for each iteration, possibly indicating some areas of our data are easier for the model to understand, certain train/test splits resulted in improved model performance. Overall, this model is not very sensitive to hyperparameter changes, and most likely will be able to generalize well to new data in similar situations without much, if any, parameter tuning.

Figure 2: Hyperparameter Sensitivity



We also observed relatively high precision and recall rates from the Random Forest Classifier, as shown in the figure below. This model could reach an AUC score of 0.92. From the beginning, this model demonstrated a good balance between the false positive rate and the false negative rate indicating its ability to correctly identify the positive and negative classes of the outcome variable at a high rate. It should be noted that other models (such as the max_iter boosted Logistic Regression model) also achieved high AUC scores, even when tested on unbalanced classes. This exemplifies the positive impact the additional features had, as well as the inherent ability for our models to accurately classify the outcome variable with minimal parameter tuning. This balance of precision and recall is preferred, as there isn't a necessity to prioritize one metric over the other in this case. Both the negative and positive classes are equally important to identify.

Figure 3: ROC/AUC curve of Random Forest Classifier



<u>Failure Analysis</u>

When creating models for the purpose of sentiment analysis, there are a few areas that could cause inaccuracies in the models. One area that may have been harming our models is the contextual sarcasm that exists in this particular area of review scores. Field specific jargon, ironic sentiment, or simply nonsense are all present to some degree in the mass of reviews. While it's unclear what portion of our specific sample of reviews contain these issues, it's important to recognize the issues they can present. We added extra features in hopes they would behave as proxies to these issues, such as if people found the reviews helpful or funny. While our models were able to perform fairly well, there are more ways to adjust for these issues, like using a subjective lexicon to help add feature information to our vectorized text. In addition, processes like stemming and word removal may impact the performance of the models. With many out-of-vocabulary words present, stemming may hurt performance in the long run. Being selective about which words should or should not be present in the dataset, and avoiding stemming may be the only way to remedy this issue. Finally, there is such a wide range of potential data to sample from in an area of interest like this, and the models were trained on a small subset of this information. One way we could ensure robustness in the future would be to include even more data, or to devise a way to more precisely sample from the total population so that the amount of reviews in each genre, company, and review type are entirely proportionally representative.

Unsupervised Learning Methods

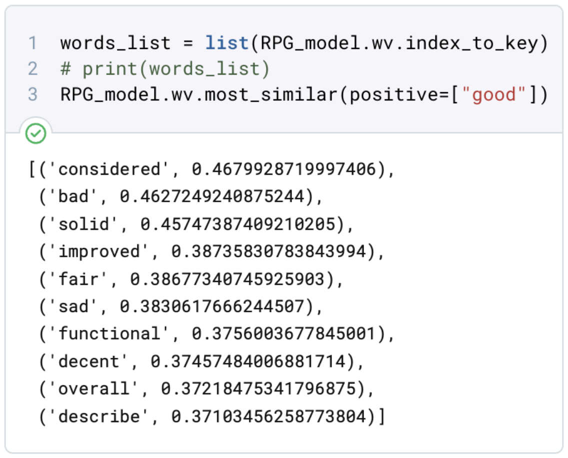
The unsupervised learning methods, focused on the reviews themselves and excluded price, playtime, or other variables. The most common stop words were removed as they would add noise to our analysis. This project utilized Word2Vec, Latent Semantic Analysis (LSA), and Latent Dirichlet Allocation. Word2Vec could easily estimate word associations, helping determine what words could lead to a review being “good” or “bad”. Gensim’s word2vec most_similar method helps find the most commonly used words when describing a game as “good” or “bad”. LSA is effective for capturing the relationship between words in a document, which could compare the information between reviews that are positive and negative recommendations. For LSA, a TFIDF object, along with a random forest classifier calculated the accuracy score. The chi-squared test extracted which terms were most relevant for distinguishing between positive and negative reviews. Lastly, LDA discovered topics from the words in the reviews. LDA Topics were created from gensim and calculated the coherence score to evaluate the model based on the number of topics chosen. Finally LDA visualized the topics within our data to observe any overlap. After the initial runs of the models, we decided to create word2vec and LSA models for each of the genres in the dataset in order to explore the differences in the reviews between genres and observe if smaller samples would show more variety.

Unsupervised Evaluation

With our word2vec models, words that indicated good reviews were “enjoy”, “fun”, “funny”, “solid”, and “fair”; while words that indicated bad reviews were “honest”, “downright”, “sucks”, and “negative”. These were not helpful findings, as these are expected words for their respective classes but, there are some interesting results when the models were run by genre. While a lot of the indicating words for “good” and “bad” reviews were similar between genres, it was interesting to see the differences between genres. Such as: “ugly” being associated with “bad” for Simulation games; “sad” being associated with “good” reviews for RPGs, but “bad” reviews for Action and Multiplayer games; and “hardest” being associated with both “good” and “bad” reviews for Action games, but only being associated with “bad” reviews for Indie games. Interestingly, we also noticed that “handling” and “aiming” were associated with “good” reviews for Action and Indie games but not mentioned in our models for other genres. This really gives a good indication of what people are looking for when playing certain genres of games.

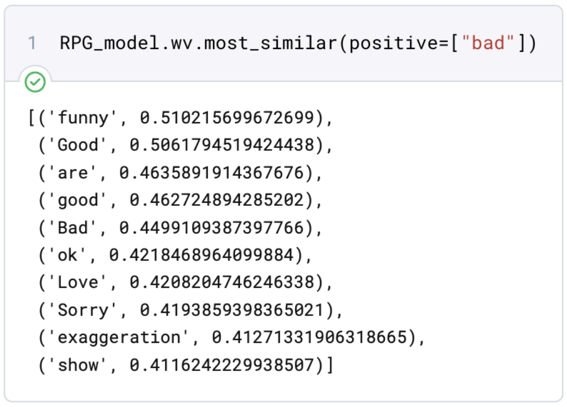
Figure 4: Word2Vec Most Similar for RPG Reviews and Simulation Reviews

RPG Reviews Most Similar to "good" and "bad"



```
words_list = list(RPG_model.wv.index_to_key)
# print(words_list)
RPG_model.wv.most_similar(positive=["good"])
```

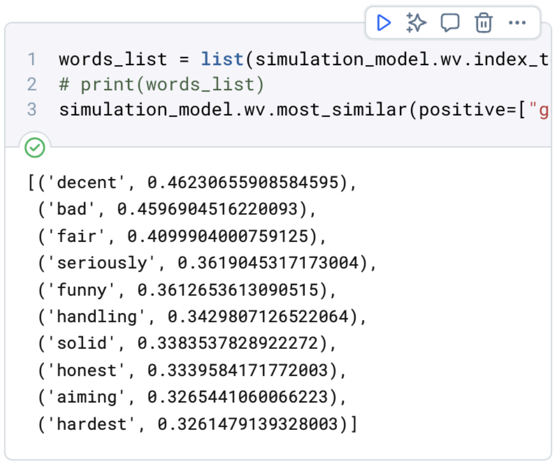
```
[('considered', 0.4679928719997406),
 ('bad', 0.4627249240875244),
 ('solid', 0.45747387409210205),
 ('improved', 0.38735830783843994),
 ('fair', 0.38677340745925903),
 ('sad', 0.3830617666244507),
 ('functional', 0.3756003677845001),
 ('decent', 0.37457484006881714),
 ('overall', 0.37218475341796875),
 ('describe', 0.37103456258773804)]
```



```
RPG_model.wv.most_similar(positive=["bad"])
```

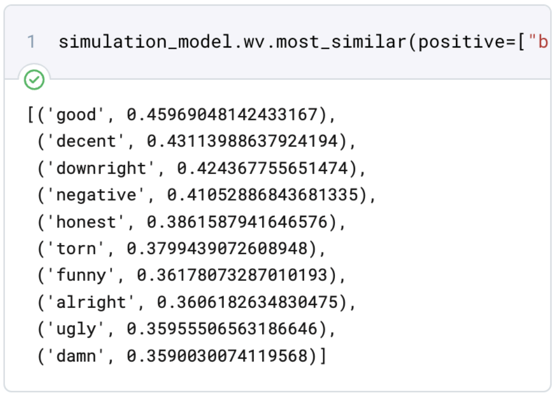
```
[('funny', 0.510215699672699),
 ('Good', 0.5061794519424438),
 ('are', 0.4635891914367676),
 ('good', 0.462724894285202),
 ('Bad', 0.4499109387397766),
 ('ok', 0.4218468964099884),
 ('Love', 0.4208204746246338),
 ('Sorry', 0.4193859398365021),
 ('exaggeration', 0.41271331906318665),
 ('show', 0.4116242229938507)]
```

Simulation Reviews Most Similar to "good" and "bad"



```
words_list = list(simulation_model.wv.index_t
# print(words_list)
simulation_model.wv.most_similar(positive=["g
```

```
[('decent', 0.46230655908584595),
 ('bad', 0.4596904516220093),
 ('fair', 0.4099904000759125),
 ('seriously', 0.3619045317173004),
 ('funny', 0.3612653613090515),
 ('handling', 0.3429807126522064),
 ('solid', 0.3383537828922272),
 ('honest', 0.3339584171772003),
 ('aiming', 0.3265441060066223),
 ('hardest', 0.3261479139328003)]
```

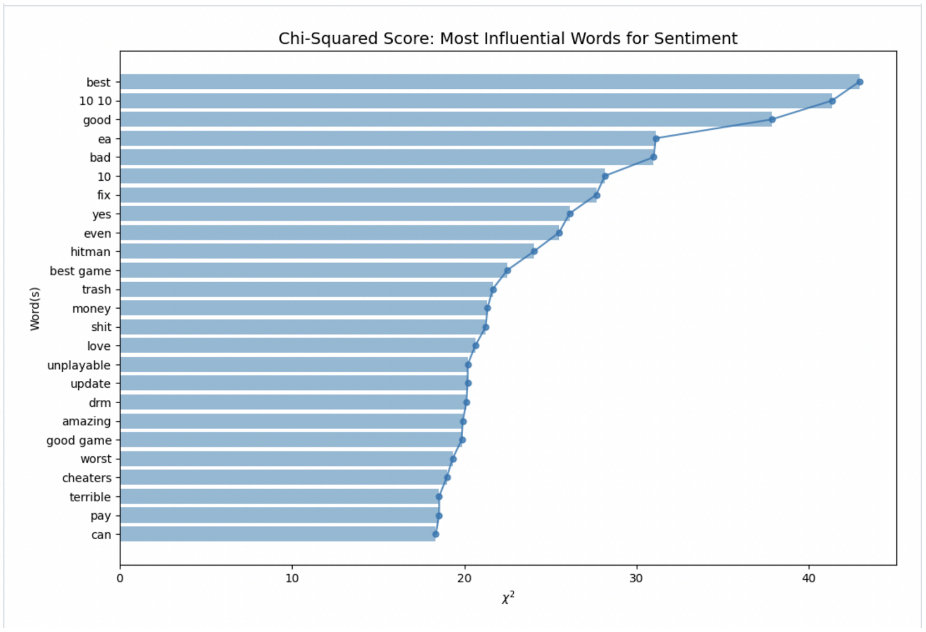


```
simulation_model.wv.most_similar(positive=["b
```

```
[('good', 0.45969048142433167),
 ('decent', 0.43113988637924194),
 ('downright', 0.424367755651474),
 ('negative', 0.41052886843681335),
 ('honest', 0.3861587941646576),
 ('torn', 0.3799439072608948),
 ('funny', 0.36178073287010193),
 ('alright', 0.3606182634830475),
 ('ugly', 0.35955506563186646),
 ('damn', 0.3590030074119568)]
```

For the LSA models, the TFIDF object and random forest classifier were able to calculate an accuracy score of 81%, with a precision score of 80% for positive and 83% for negative, along with a recall of 75% for positive reviews and 86% for negative reviews. We are satisfied enough with this, given that reviews can be written by anyone on Steam and could have been affected by things such as review bombs and sarcasm. Calculating these scores by genre, the precision and recall scores were higher for all genres, with the majority of them being 93% or higher and the lowest being 87%. This makes sense given that the words within a genre will be more specific to them and a common word for one genre will likely throw off the accuracy for the entire dataset, since it will not be used much in reviews for other genres. Along with these tests, we also used Chi-Squared tests to visualize the most important words that distinguish between the recommended and not recommended reviews.

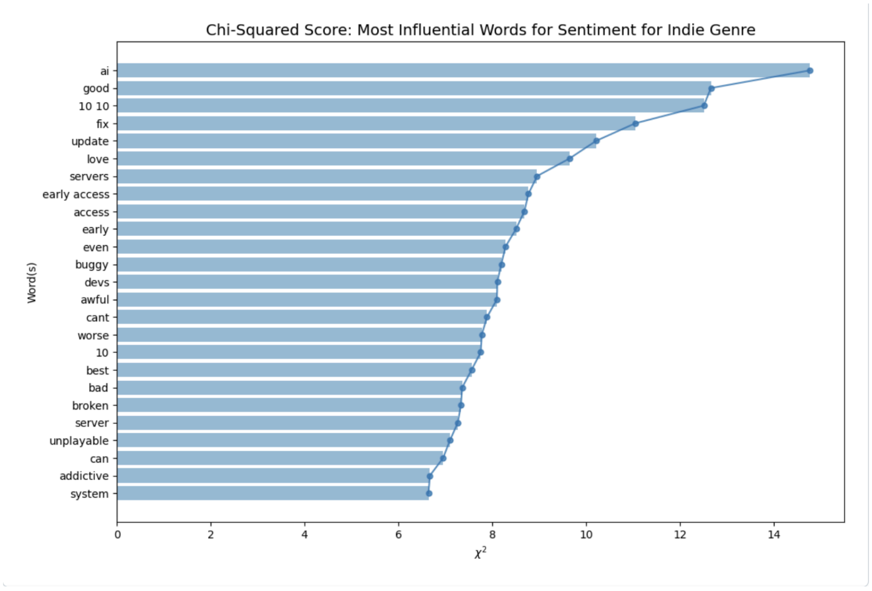
Figure 5: Chi-Squared Score: Most Influential Words for Sentiment All Reviews



Like the word2vec models, the initial model utilizing the entire dataset was not as insightful. Words like “best”, “good”, “10”, and “bad” were going to differentiate reviews. We think this is the case when analyzing data from so many different games because words like these are common factors that could apply to any game. It is more beneficial to analyze the models created for the reviews of each genre.

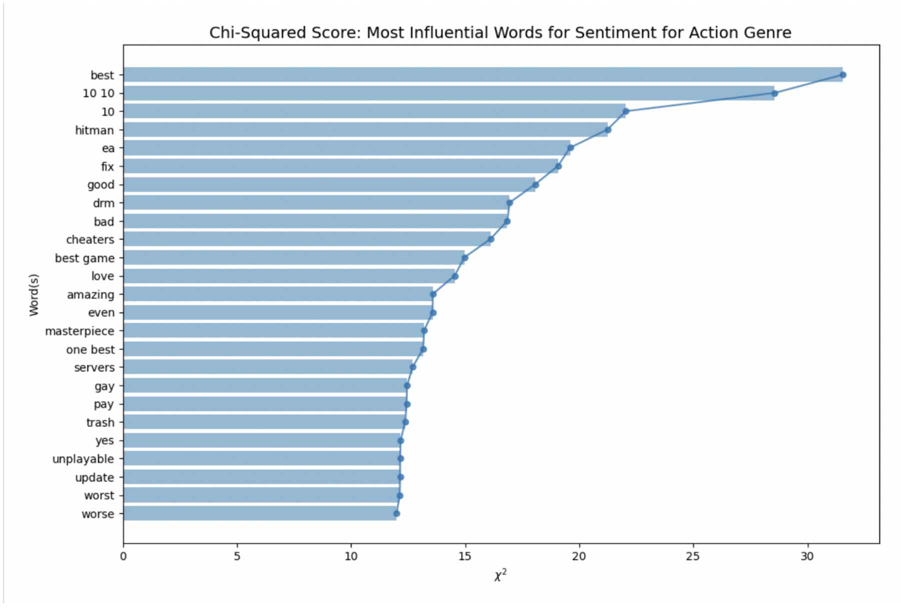
From our Chi-Squared tests by genre, we observed that the words “unplayable”, “fix”, “servers”, “crashes”, and “update”, “pay”, “money”, and “ea” were common amongst all genres. These words were mostly observed in our model above from the entire data, so it’s not surprising to see them in a majority of genres individually. EA (Electronic Arts), being a major video game company, is not surprising to be found in these models since many reviewers like to mention the developer or publisher by name. While the rest of these words are common complaints in video game reviews, it makes sense why they are here. Interestingly, Indie games had one of the more unique results from this model with “early access”, “buggy”, “devs”, “broken”, and “ai” being amongst the most influential words for sentiment. All of these make sense for Indie games, as they are often created by smaller teams with less resources and are more often released into early access. However, it’s surprising that these were only found to be most influential in Indie games when they are fairly common amongst all genres.

Figure 6: Chi-Squared Score: Most Influential Words for Sentiment Indie Reviews



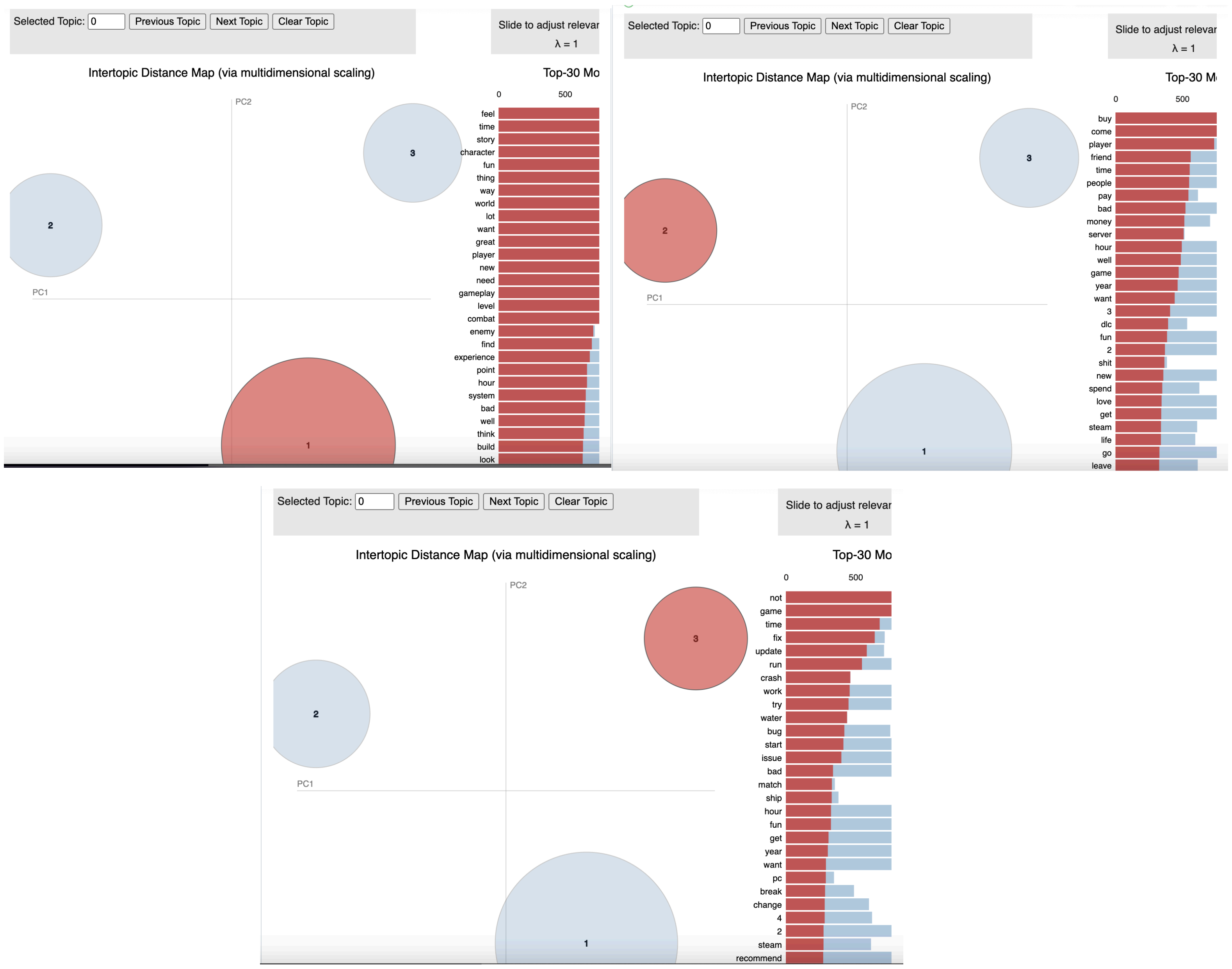
The Action genre was another Chi-Squared model that had distinctive words such as “hitman”, “drm”, and “cheaters”. However, unlike Indie games, these words were prevalent in the Chi-Squared model for all data, regardless of genre. This shows the influence of the Action genre in our dataset as a whole. While this is likely due to the Action genre being very popular amongst gamers, it would be interesting to see if we scraped data from Steam again, if that would still be the case.

Figure 7: Chi-Squared Score: Most Influential Words for Sentiment Action Reviews



For the LDA models, the coherence scores determined that the optimal number of topics to generate is three. After running this model with a different number of topics to confirm, we determined that tree topics lead to no overlap, which is ideal for determining distinct topics.

Figure 8: PyLDAVis Intertopic Distance Map



The three topics that we generated from our LDA model describe 3 different types of reviewers. Topic 1 describes a reviewer with an overall positive view, notated by the words “fun” and “great”. They value the gameplay and combat of a game. This type of reviewer likes to be immersed in the feel, story, character, world, and experience that a game offers. Topic 2 details a reviewer who appears to have mixed feelings. The words “buy”, “pay”, and “money” could go either way, but “bad” and “want” have a negative connotation. This type of reviewer does have fun with friends and other people, but there are definitely things that could improve their experience. Topic 3 describes a reviewer with a more negative view that places emphasis on the crashes, bugs, and issues that they have encountered while playing the game. This isn't to say that the review is all bad, as "fun" is still an influential word. This type of reviewer is just more critical of the game and wants to make recommendations on what could be fixed or updated. These topics help us make sense of the most influential terms amongst all of our reviews, putting more of a meaning behind them. We believe these three topics cover a wide array of opinions, not only amongst sentiment, but also what parts of a game people value the most.

Sensitivity Analysis

Through the exploration of all of our unsupervised models, we observed that they are all sensitive to changes in parameters, features, and data inputted into them. All of the models were run with different stages of preprocessing done to the data and each time the results were estranged drastically. While these models only directly used the review columns of our data set, making changes to the parameters of that data set indirectly impacted the results. Even something as simple as letter case and special characters will impact the models, since they are returning words directly from the data. Separating the data into genres of games had an impact on the models; so if we further separated those into different groups, or even further separated the genres into different price ranges for example, we would see substantial differences in our models' results.

Supervised Learning Discussion

One of the biggest surprises was that sarcasm and jargon did not pose as big of an obstacle as anticipated. Added features did boost performance and our models were able to achieve relatively high performance in our evaluation metrics, however it's unclear if the overall ceiling was lowered due to the content of the reviews we sampled. If given more time for further research into the field of sentiment analysis, we could have implemented subjective lexicons, or sampled even more data to further investigate this. Testing many different combinations of features and models could've led us to an even deeper understanding about these methods, and how to further optimize the supervised learning models. Another surprise regarding features was the impact of each feature on our highest performing model. The features that represented user playtime in each game were shown to, on average, lower the accuracy of our best model. In earlier tests, the addition of these features boosted accuracy in most models. However, using k-folds cross validation seemed to expose the particular train/test split as being an exception to the overall impact that these features had. If we had more time and could sample different combinations of data, we could have investigated this issue further.

In terms of model performance, it was a bit surprising that the Random Forest classifier performed the best. We expected Logistic Regression to outperform the other models by a noticeable amount. This may have been the case if we had the resources to continue testing and optimizing model performance, but to increase the accuracy of the Logistic Regression model we needed a stronger machine to reduce the overall time. A big challenge of running the model creation and evaluation steps was the overall computation time. The high performing Logistic Regression models had a long training time and a lot of resources to run, resulting in choosing to compare default model parameters, and selecting the Random Forest model as the primary model to continue evaluating. If we had a stronger machine and more time to explore, Logistic Regression may be improved by a significant amount. Testing the upper limits of this model may have also aided in exploring the impacts that the features and preprocessing steps had on the accuracy of our models.

Unsupervised Learning Discussion

Overall, the Unsupervised Learning models demonstrated what words are associated with “good” and “bad” reviews, the most influential words for sentiment, and what types of reviewers there are in our data set. While we have noticed that these results can change drastically based on the data used and will likely change over time, this is a good summary of the data we gathered at this time. These models allow room for future analysis, with a solid foundation to extend, as well as a rich collection of results for future collection.

What surprised us about the results of the unsupervised learning models were how sensitive they are to the changes in the data. We suspected the results to be similar when running the models with different amounts of data and different parameters, but the results changed drastically every time we adjusted the model. We were also surprised by how lackluster the results of the models were when running them on the entire dataset. The results were adjectives that were very common for video game reviews, but upon separating data into genres, we found more revealing words.

The biggest challenge that we encountered with these models was interpreting the data after changes were made and trying to decide on the parameters and data that would lead to the most effective results. It definitely was helpful that we were running models multiple times between all of these changes because it helped us learn what could impact the results and what we needed to look out for, not only while working on this project, but in future endeavors with these types of models. In the end, we were able to overcome these challenges by evaluating our models at all stages and deciding what model gave us the most insightful results.

While the topics we determined from our LDA model were insightful, given more time and resources, running our LDA model on all genres to determine the topics for them and not just the data set as a whole. It also would have been interesting to see how the results differed if we split our data up further into different price ranges, languages, and even play time of the reviewers. More specific results could be found if the models were run on a single game, but it would not have been representative of the wider steam review population. Given enough time, more games could be scrapped to encompass all genres.

Ethical Considerations

Steam users leave reviews with the expectation that they are public, but likely did not consent to its availability through public API endpoints. The platform’s terms and conditions are extensive and most users did not read or comprehend the extent their data could be accessed. User consent was ignored scraping this data. Most users are not aware their thoughts and criticisms could be accessed from outside parties without agreement. With a lack of transparency, Steam users forfeited their data ownership over their reviews to solely be governed by Valve. To address this issue, we de-identified the records returned by the API. Dropping the user id, makes re-identification difficult due to the lack of personal details in review text. It doesn’t fully remedy the lack of consent, as other features such as pointed reviews, hours played could lead to re-identification.

Statement of Work

	Antonio Montalto	Jeffery Jones	Reed Paskvan
Draft Proposal	All		
Stand Up #1	Support	Support	Lead
Stand Up #2	Lead	Support	Support
Steam Scraping	Support	Support	Lead
Pre-Processing	Support	Lead	Support
Data Viz Exploration	Support	Lead	Support
Unsupervised Learning	Lead	Support	Support
Supervised Learning	Support	Lead	Support
Final Report	All		

References

Bais, R., Odek, P., & Ou, S. (n.d.). Sentiment Classification on Steam Reviews [Thesis]. Retrieved March 3, 2024, from https://cs229.stanford.edu/proj2017/final-reports/5244171.pdf

Lane, R. (2018, March 13). What developers think of Steam reviews. Rock Paper Shotgun. https://www.rockpapershotgun.com/what-developers-think-of-steam-reviews

Li, Susan. "Latent Semantic Analysis & Sentiment Classification with Python." Medium, 6 Dec. 2018, https://towardsdatascience.com/latent-semantic-analysis-sentiment-classification-with-python-5f657346f6a3

Steam Market Report Q3 : Strongest Ever Q3 Performance. (2023, October 10). Video Game Insights; VGInsights. https://vginsights.com/insights/article/steam-market-report-q3-2023-strongest-ever-q3-performance

Wagh, Swapnil. (2020, October 9). Sentiment-Analysis-for-Steam-Reviews. GitHub. https://github.com/SwapnilWagh06/Sentiment-Analysis-for-Steam-Reviews/blob/main/Sentiment%20Analysis%20for%20Steam%20video%20Game%20using%20NLP.ipynb

Valve Corporation. (n.d.-a). ISteamApps Interface. Partner.steamgames.com. Retrieved March 3, 2024, from https://partner.steamgames.com/doc/webapi/ISteamApps

Valve Corporation. (n.d.-b). User Reviews - Get List. Partner.steamgames.com. Retrieved March 3, 2024, from https://partner.steamgames.com/doc/store/getreviews

Valve Corporation. (n.d.-c). Web API Overview. Partner.steamgames.com. Retrieved March 3, 2024, from https://partner.steamgames.com/doc/webapi_overview

Valve Corporation. (2010, July). Steam Web API Terms of Use. Steamcommunity.com. https://steamcommunity.com/dev/apiterms

Zhang, Anqi, "Sentiment without Sentiment Analysis: Using the Recommendation Outcome of Steam Game Reviews as Sentiment Predictor" (2022). Electronic Theses and Dissertations. 490. https://openprairie.sdstate.edu/etd2/490

Zuo, Z. (2018). Sentiment Analysis of Steam Review Datasets using Naive Bayes and Decision Tree Classifier [Thesis]. https://core.ac.uk/download/pdf/159108993.pdf

Appendix A

Feature	Description	Data Type
price	The purchase value of the game	float
review	The user written recommendation for a game	string
playtime_forever	The total hours played by the reviewer for this game	float
playtime_last_two_weeks	The total hours played by the reviewer in the last two weeks for this game	float
playtime_at_review	The total hours played by the reviewer at the time of review submission	float
voted_up	1 - Reviewer gave the game a positive recommendation; 0 - Reviewer gave the game a negative recommendation	Binary integer
votes_up	Number of helpful ratings given by the user community for a review	integer
votes_funny	Number of funny ratings given by the user community for a review	integer
Action	Game genre is labeled as Action by developer, 1 - yes, 0 - no	Binary integer
Adventure	Game genre is labeled as Adventure by developer, 1 - yes, 0 - no	Binary integer
Casual	Game genre is labeled as Casual by developer, 1 - yes, 0 - no	Binary integer
Early Access	Game is released in Early Access, typically in Alpha or Beta, being actively developed for an eventual full release	Binary integer
Free to Play	Game is free, no upfront purchase to play/install	Binary integer
Indie	Game genre is labeled as Indie by developer, indicative of a smaller development team, 1 - yes, 0 - no	Binary integer
Massively Multiplayer	Game genre is labeled as Massively	Binary integer

	Multiplayer by developer, 1 - yes, 0 - no	
RPG	Game genre is labeled as RPG by developer, 1 - yes, 0 - no	Binary integer
Racing	Game genre is labeled as Racing by developer, 1 - yes, 0 - no	Binary integer
Simulation	Game genre is labeled as Simulation by developer, 1 - yes, 0 - no	Binary integer
Sports	Game genre is labeled as Sports by developer, 1 - yes, 0 - no	Binary integer
Strategy	Game genre is labeled as Strategy by developer, 1 - yes, 0 - no	Binary integer