Game Review Analysis*

Pope
Department of Computer Science and Cybersecurity
University of Central Missouri
Warrensburg, USA
ijp74190@ucmo.edu

Cross
Department of Computer Science and Cybersecurity
University of Central Missouri
Warrensburg, USA
adc03000@ucmo.edu

***Abstract*—This project paper will outline the concepts, execution, and results of testing multiple machine learning algorithms in a real-world scenario. Specifically, it will delve into which algorithm that has been taught about in class would best fit user review prediction in the video games industry. Using a database of games from the popular online video game marketplace Steam made this project possible. Selecting metrics like user playtime, external review media scores, and price at a game's release allowed for use of the particular algorithms of choice. While direct manipulation of the given data was required, a final accuracy score was calculated with each algorithm, determining which would work best in this given use case. Data acquisition was the first step of the project, finding a reliable, legible, and usable database that was current and had enough instances to make testing possible. Data preparation was required for the next step of the project, as the database that was used had an excessive amount of unnecessary information tied with it. Pruning the data tables of information that had no relevance to the project made for a far more manageable table of information to use. After this step was taken, the next goal was execution of the chosen algorithms. Subsequent reflection, alterations during execution, and criteria changes were made during this step of the process, as some algorithms posed more requirements than others. Eventually with all the algorithms having been implemented and run with the given data set, the reflection of provided data was necessary. Acquiring visualizations of accuracy for each algorithm made comparison between each method very simple and easy to understand. From there, comparisons were identified and a compilation of algorithms that did better than others was constructed. Information of which algorithm would perform better to predict user reviews for video games is finally produced. With this information, the final result of which algorithm is best suited for this use case was identified.**

***Index Terms*—naive Bayes, knn, regression, decision tree, svm, Steam, reviews**

I. Introduction

A. Overview

The video game marketplace has been an ever-changing scene for decades, especially when comparing recent years to the history of video games as a whole. Along with the sale and marketing for video games having changed, so too has the culture around gaming, specifically reviewing video games. Like any form of media entertainment, people take it upon themselves to rate video games on a multitude of different scoring bases. Whether it's stars, a percentage score, or a simple number, many different outlets and marketplaces put heavy emphasis on reviews of games as a whole.

Much like purchasing anything else online, it's commonplace to look at reviews of the given item to see if it's worth buying. On average, items with a higher review score tend to sell much better than those that don't have as high a review. This is no surprise, and this trend is no different in the video game industry. From an outside perspective, looking at a list of games and information about the game, it's hard to tell if its given attributes actually make it a good game. Reviews are mostly subjective, and depend on how well a given game caters to an audience, fills a certain niche, or meets the demands of a genre. Even so, taking measurable metrics from users and determining if they like a game on said metrics could prove to be incredibly useful. This project aims to test this theory, taking multiple pieces of data on users' interaction with video games, and determining if they would leave a positive review.

This project specifically aims to answer the question of which machine learning algorithm would work best in this use case. Dealing with a variety of factors, varying information, and a large dataset are essentials to the algorithm of choice. The algorithms being tested in particular include Naive Bayes, SVM, Logistic Regression, K-Nearest Neighbor, Decision Tree, and Linear Regression. Some of these were expected to work better than others, and in execution, each algorithm showed its strengths and weaknesses. This 'proof of concept' experimentation could help to drive a field of research for the video game industry on user reviews, calculating and accurately predicting reviews, and how to cater to audiences to achieve such positive reviews.

B. Importance of Research

This research could prove to be an important gateway for game developers to analyze what players like in games as a generalized metric. Understanding what a player base does and does not like can be a very simple way to modify an existing game, alter development of a game, or even produce a new title altogether. This kind of research can also compare different kinds of games, different years of development, or different genres to see which aspects of specific criteria were successful given their restrictions. Games mold and change over time, certain genres are successful one year, yet stagnant another. Using a 'review expectation' algorithm on a game can help developers predict how well the game might perform commercially.

Not only this, but it also helps developers understand what staples of games are successful regardless of changing criteria. Certain combinations of genre, game type, game length, etc. can culminate in a perfect blend that players really enjoy. Using this research method can help outline just what these trends are, how to cater to them, and how to release reliably good-reviewing games. This could prove to be a solution to an ever-growing problem in the gaming industry, where titles are announced and released with very high critique and not a lot of positive reviews. Understanding what the audience prefers en masse is a simple way to ensure that whatever audience is engaging with the game will be more inclined to leave a positive review.

With the project as a whole, the data gathered is dependent on what you want the method to produce. The desired outcome for this project is a generalized metric for the sake of testing this methodology; given user playtime, user reviews, and the Metacritic score for the game, a third-party website that scores a plethora of media. With this data, we aim to predict if a given game will have a positive review ratio, or not. A positive review ratio in the scope of this project is a final user review score of 85% or higher, with any other number lower than that denoting a negative review score. This might seem like a high number, but through testing it proved difficult to find a substantial difference in outcomes with any positive review ratio lower than 85%. This could prove to be an area of research itself in the nature of leaving positive reviews, if leaving reviews at all. For the sake of the project, however, the focus is on user review score prediction.

C. Impact of Further Research

This area of research could prove beneficial not only to professional development companies, but to indie developers as well. Indie development refers to a small group of people, or even a single person, working on a game. This is in contrast to AAA studios, where hundreds of developers are working on a game. Providing unique benefits for both parties, with further success for AAA studios and a higher chance of recognition and commercial success for indie developers, research into this specific field could prove to become a keystone implementation in the development cycle. This is also not an incredibly hardware-dependent or software-specific project that can be executed, meaning its availability and accessibility can be widespread amongst the game development industry.

The research into positive review ratio prediction can introduce many positives in the industry as a whole. From the range of uses, how beneficial it can be in any part of the development cycle, and how easy it is to execute, it could prove to be a staple in game development in the near future. This project is only the first step in producing a stable approach to accurate review prediction, with the primary focus being on which machine learning algorithm could prove to be the best fit for the job. Through testing in execution with an accurate database, using realistically attainable metrics, and without any extreme hardware, this project will outline what algorithm is the best fit to predict if video games receive positive reviews.

II. Motivations

The motivation behind this project was the mutual interest of machine learning algorithms and concepts, as well as a shared interest in the video game industry as a whole. Not only did it take a unique approach to a real world problem, it also opened a whole new view into the game development industry as a whole. Taking user review data is obviously a useful form of feedback for developers, helping to mold and modify an existing game to make the gaming experience all the better. This extended research into review prediction and a possible mainstream view into review prediction could lead to huge changes and improvements in the gaming industry.

This potential to make widespread changes, and possibly improvements, to the development world was a large motivation for the project as well. Able to predict user reviews is undoubtedly a useful tool to be used. Making this tool accurate and commercially viable to large and small developers alike could prove to change the landscape of development as a whole. Contact and proper communication between large developers and consumers has been an ever growing problem for years in the industry, and is starting to have impactful negative effects. Use of this process and whatever algorithm proves most accurate in review prediction could help developers discover and cater to what their given audience truly enjoys. This improvement on developer-to-consumer relations would only help the industry grow in a positive manner. As such, focus on this project to help consumers and developers alike was a huge motivation overall.

III. Main Contributions and Objectives

- Determine if various machine learning algorithms can be used to predict public perception of a video game.
- Process a dataset to create a new data usable for machine learning algorithms.
- Analyze the performance of machine learning models in the relation to video game reviews.
- Determine the best algorithm for classifying review percentages of Steam games.
- Determine if machine learning algorithms and dataset hold future research opportunities and general usability in the video game industry.

IV. Related Works

In the execution of this project, many different machine learning algorithms were tested and used, all with different approaches to a similar problem. Understanding how each algorithm and its relative algorithms is important to ensure that the right tool is used for the job. For this project, we compared a wide arrangement of algorithms all used in class, including a series of Naive Bayes algorithms, SVM, Logistic Regression, Linear Regression, K-Nearest Neighbors, and Decision Tree. All of these algorithms are tested against the data set and compared in accuracy to one another to see which one reaches the highest accuracy score. For the sake of the project, research went into the direct implementation and

use of each aforementioned algorithm to ensure it was used to the highest efficiency possible.

A. Gaussian Naive Bayes

The first of the four Naive Bayes related algorithms that was tested and used was the Gaussian variant. Gaussian referring to a 'normal distribution', this Naive Bayes algorithm works on data sets by breaking down each individual piece of data and classifying it within a set field [1]. This field is then plotted on a graph in the form of a normal distribution, or a Gaussian curve. From there, with supplied information that formed this curve, testing values reference the graph(s) and subsequently fall in whatever group they are more closely related to. In ideal scenarios, the graphs are visually distinct and far apart from each other and ensure the chances of a new entry being distinct with what group it is a part of [1]. Multiple graphs can be made for multiple different pieces of data retrieved and tested.

B. Multinomial Naive Bayes

The second of the four Naive Bayes related algorithms that were tested and used was the Multinomial variant. A machine learning algorithm that works best with discrete data like counting text in a file, Multinomial Naive Bayes works by comparing the amount or presence of a given piece of data in a test input with different desired outcomes or groups, often times using a histogram in the process of analyzation [2]. Depending on which trends the test values present when comparing to the different classifications, said test values are determined to be of one group or another. This algorithm also follows the Naive Bayes principle for classification in its process.

C. Complement Naive Bayes

The third variant of the four Naive Bayes related algorithms that were tested and used was the Complement variant. Most similar to Multinomial Naive Bayes, Complement Naive Bayes works on the same principle of calculating the probability of items belonging to certain groups. Where it differs, however, is that it calculates the probability of a given item being a part of every class, and subsequently calculating which group the item is then closest to or most similar to [3]. Multinomial Naive Bayes only calculates the probability of a given item being a part of a single group, which comes with drawbacks. Specifically, the Complement approach does well with varying data, or data sets that have a degree of bias to one group compared to another [3].

D. Bernoulli Naive Bayes

The final variant of the four Naive Bayes related algorithms that were tested and used was the Bernoulli variant. The Bernoulli variant, a supervised or "labeled" variant of the Naive Bayes algorithm suite, is special in that it deals in binary. Not only does it give a classification of if an item is part of a group or not, which is binary itself, it deals with binary dataset values. Some modification is required in particular cases to make it work, but it's often used to detect if a word is in a text document or not [4]. Even still, if other dataset information can be converted into binary values (in this project's case, detecting whether a score or review percentage is above a specific threshold), then it can easily be used for implementation.

E. Support Vector Machine (SVM)

The next of the listed algorithms being used for this project is Support Vector Machine (SVM). SVM works by classifying data into particular categories, and determining the biggest boundary between said categories. In doing this, it makes classifying new data simple by detecting which side of the barrier the new point lays. From there, the item is then added to its relative group. The barrier this algorithm uses is called a hyperplane, and in order to make its volume as large and distinct as possible, it checks the two closest items from different groups [5]. The hyperplane is made in between these close, yet opposing items to maximize its efficiency in a plethora of different situations [5]. For this project, its use is no different, and provides useful results.

F. Logistic Regression

Continuing with the list of applied algorithms, the next variant is Logistic Regression. This algorithm classifies a binary output given different test information. To accomplish this, Logistic Regression uses what's called a sigmoid function [6]. Visually, it looks like an "S" shaped curve between two horizontal lines, each respectively either of the binary outcomes of the particular test [6]. Depending on where the new test data lies in respect to this shape determines which binary output the new item belongs to. This algorithm works particularly well with larger datasets, making it particularly robust and accurate in ideal circumstances. It does suffer from outliers, however, making it vulnerable to varying datasets.

G. Linear Regression

Following Logistic Regression in the list of different tested algorithms used for the project comes Linear Regression. This algorithm outputs a numerical classification to given test items based on multiple variables at once. Possible to be viewed in 2D or 3D depending on the amount of variables used at a given time when visualizing, Linear Regression works by plotting new item information on a linear plot found from the given dataset [7]. The fact that it gives a numerical value can be incredibly useful where a binary classification might be too generalized. While you can further classify the results of the Linear Regression algorithm into binary values, it can help give a respect and relative weight of the variable based on the values used to classify it. For being such a simple algorithm, it is naturally very robust and can easily be used for larger datasets.

H. K-Nearest Neighbor (KNN)

The next algorithm used and tested for this project is K-Nearest Neighbor (KNN). This algorithm works by plotting all

variables of the dataset on a plot, grouping them depending on their classification, and referencing new items added to said plot. When a new item is added, the algorithm checks the nearest dataset values. Depending on which classification has more closer values to this test value, its own group classification reflects the majority of said dataset plots [8]. In simple terms, the closer and more numerous a certain classification, the more likely the test data is to be classified the same. This is a very simple algorithm to use, and doesn't require a test period. Since it simply works on relative plotting, it does not need to 'test' itself, rather giving classifications straight away when given an ample data set [8].

I. Decision Tree

The final algorithm used in this project for testing and implementation is the Decision Tree algorithm. This algorithm works by creating testing nodes that incrementally grow until a classification condition is reached. Depending on how long this tree can 'grow', this algorithm can be incredibly versatile, simple, and easy to visually understand its process. After ample learning and subsequent testing, the algorithm forms these testing nodes in a way that a new item can flow through each connected node, slowly analyzing each variable of the new item, and eventually reaching its classification condition [9]. These nodes are optimized to lessen the amount of nodes and classification steps in total, meaning it can become quite efficient as well.

All of these algorithms have separate strengths and weaknesses, which is ideal for the testing nature of this project. Having a wide variety of capable algorithms can lead to unique situations, useful finding, and eventually, an algorithm or multiple algorithms that seem especially proficient in the user review classifying the nature of the assignment. With these algorithms and a comfortable grasp on how they each function, implementation and use is far easier to understand and work with. Eventually tallying the respective accuracy of all these algorithms will prove which particular algorithms work better than others.

V. Proposed Framework

A. Topic

This experiment is designed to evaluate game performance on the computer video game marketplace Steam and evaluate how different machine learning algorithms perform in the prediction of the marketplace. The way this experiment looks at how the market likes games is based on what percentage of a game's reviews were positive. This is calculated by the number of positive reviews divided by the total number of reviews to create a positive review percentage. An average positive review percentage for a game on Steam comes in around 80% to 85% [10]. Because of this, a positive review threshold is set at 85% or 0.85, meaning that a game that ranks above the threshold is considered to be reviewed above average. Multiple machine learning algorithms will be applied in order to attempt to classify games into one of two classifications, one being above the positive review threshold and zero being below the threshold. To predict the classification of the game, the models are given three different features to examine. The first feature used is Metacritic score. Metacritic is an external website that gives a score based on how they perceive the quality of the game. The game's score is then divided to place the score between zero and one. The next feature is the median playtime of the game in minutes over the duration of the games lifetime. The minutes are then placed into a logarithmic function to minimize the impact of the large outliers in the data and then is normalized. The final feature in the evaluation is the game's original price, this feature is also then normalized to be placed between zero and one.

B. Algorithms

Many simple machine learning algorithms are used to compare their capabilities in classification of the Steam games positive review percentage. The first classifier used is a support vector machine in the form of a linear support vector classifier or linear SVC. The next algorithms used are Naive Bayesian models. Complement Naive Bayes, Multinomial Naive Bayes, Gaussian Naive Bayes, and Bernoulli Naive Bayes algorithms. For the comparisons between types of classifiers the Naive Bayes methods will be represented by the model that scores best in terms of accuracy. The next algorithm used to predict classification of the games data is a linear regression model. This model does not directly predict classification [7], but instead outputs a probability for being over the positive review percentage threshold. If a prediction is greater than or equal to 0.5, then the prediction is categorized as being in the one classification. If the prediction is below the threshold, then the classification is considered to be zero. The next model used for classification is logistic regression. Another set of algorithms will be run using the k-nearest neighbor algorithm. The predictions are run with varying values of k. One is run with the value of k=5, another ran with k=7, one ran with k=9, and finally a k-nearest neighbor algorithm will be run with k=11. After running all algorithms, the model that predicts the classification of Steam games with the highest accuracy and then the lowest number of k, will be the algorithm used to evaluate the k-nearest neighbor algorithm against the other algorithms run. The last simple machine learning algorithm used to evaluate the dataset is the decision tree classifier. The tree algorithm will run on a modified dataset. This dataset will contain the same entries but will be in a different format. For the other algorithms the features are normalized between zero and one. For the decision tree, each feature of a game will be categorized into ten different buckets based on the tenths place of the feature, resulting in ten buckets. The decision tree will then be run with this dataset to limit the number of possible branches and to increase the number of filled branches. An unpruned tree will first be run to determine the effectiveness of the unpruned tree. Since a tree allowed to reach max depth will overfit on the training data provided [11], the tree will be pruned to limit this phenomenon. To prune the tree we will apply a post pruning effect. This post pruning algorithm is the cost complexity pruning technique implemented in [11]. After

finding the optimal mapping of the tree, the then post-pruned tree will then be run again and evaluated. For comparison with other models, the post-pruned tree will be used with the assumption that the accuracy improves after pruning.

C. Evaluation

The dataset will be split into training and testing data, where the training data consists of 80% of the data and 20% belongs to the testing data. Each model will be fit to the same training set. These models will then be given the testing features and provide an output of predicted classifications for each label. Once a predicted label is made, the algorithms will then be able to be evaluated. The primary evaluator is how well the models can accurately predict the classification of the data. Other evaluators collected will be the weighted precision, recall, and f1-score for each algorithm, as well the independent precision, recall, and f1-scores of each classification. The evaluators that are not accuracy will be used to evaluate any biases within the model as well as evaluate the models for balance. Once all algorithms have run, the models used to represent the Naive Bayes and k-nearest neighbors will be selected. The algorithms will then be examined and compared to each other. Of the models the accuracy will be compared to the others as well as any strengths or weaknesses that may be found within the models. Potential weaknesses that are to be evaluated are a lack of accuracy, an imbalance in predicted classes, such that a high variation in precision or recall between classes, and if the models appear to bias towards a single class. During the evaluation, an attempt will be made to decide what is the best model, if a singular best model exists, that is the best model to use in the classification of whether or not a game is reviewed positively or negatively at a rate that is above average. A goal of this experiment is to evaluate the usability of the algorithms. This will be based on how accurate the models are able to predict the classifications. If the algorithms are unable to accurately predict the classification of the data then the use of the models would be very little. On the other hand, a highly accurate classification percentage would show that the features used were good predictors of how well the game is perceived. Although the analysis is based around how well a game is perceived, if the accuracy is high, game developers may be able to tailor their games to perform well in these feature categories. If the accuracy of these algorithms are high, it may suggest that the features and predicted values may be able to be altered to predict game success instead of just the ability to judge how well the game is received. Along with this, the results will be looked at to determine if a single algorithm may be best to predict future data.

VI. Data Description

The data used in the experimentation was taken and modified from a Kaggle.com dataset under the name 'Steam Games Dataset' uploaded by Martin Bustos. From the dataset columns of Metacritic score, price, median playtime forever, positive, and negative. The positive and negative columns refer to a review from a user on Steam. The positive and negative columns were then used to create a positive review percentage by taking the positive reviews divided by the sum of positive and negative reviews. If the positive review percentage was less than the imposed threshold of 0.85, then it would be classified as a zero, while it would be classified as one if greater than or equal to the threshold. Games without a Metacritic score, less than five positive and negative reviews, less than ten or greater than twenty thousand minutes median playtime were removed from the dataset. This left 3128 entries in the dataset. The Metacritic score was divided by 100 to be between zero and one, price of the game was also normalized between zero and one, and finally the log of the median playtime was normalized. A logarithmic function was used because most of the values of median playtime were relatively small in comparison to the maximum value. The dataset rows were then randomized. This dataset is modeled in Fig. 1, with the green dots representing data over the 85% positive review threshold while the red dots represent data under the threshold. The dataset was then modified for use in decision trees by grouping each feature into one of ten buckets by multiplying the value by 10 and parsing into an integer. Both datasets were then split 80% training and 20% testing data, with each algorithm getting the same training and testing sets.

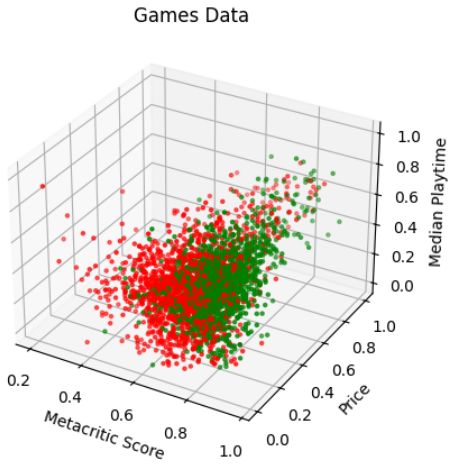

Fig. 1. Mapped Dataset. Red is negative class, green positive class.

VII. Results / Experimentation and Comparison / Analysis

A. SVM

The first machine learning algorithm used to predict whether a game is above or below the positive review percentage threshold of 85% was a support vector machine in the form of a linear support vector classifier or linear SVC. After the SVM's training had taken place, the SVM was able to accurately classify the training data at an accuracy of 0.7186. On the testing data, the SVM was able to correctly classify the game with an accuracy of 0.7109. The difference between

the accuracy of the training data and the testing data comes to only 0.0077. This low variance shows that the SVM performs similarly on both training and testing datasets and suggests that given the information available to the algorithm, it is likely fit properly. Other metrics, weighted by the number of games belonging to each class and used to evaluate the performance of the SVM include a precision of 0.71, a recall of 0.71, and f1-score of 0.71. The zero and one classes were also generally well balanced with precision and recall scores for both being between 0.70 and 0.72. Because all of these statistics and the accuracy of the model are all roughly equivalent, it suggests that the model and data are both balanced.

B. Naive Bayes

The next kind of machine learning algorithms used were Naive Bayesian models. Four different kinds of Naive Bayes models were used, those being multinomial, Bernoulli, complement, and Gaussian Naive Bayes models. Two of the models performed almost the exact same. Those models are the multinomial and Bernoulli. The multinomial algorithm ran with an accuracy of 0.5128, while the Bernoulli algorithm ran with an accuracy of 0.5112. In both models, they predicted the classification of every game to be below the positive review percentage threshold. The only exception to this is one game that the Bernoulli algorithm predicted to be over the threshold, however that prediction was a misclassification and explains the slightly lower accuracy found in the Bernoulli model. One possible explanation for this phenomenon is that they were both choosing the dominant class. An imbalance training dataset could lead to models routinely choosing the dominant class [1]. Even though the dominant class was below the positive review percentage threshold, the training data was still only about 53% made up of that class, making an imbalance in classes less likely the reason for the multinomial and Bernoulli algorithms to choose the same class for every game. Other than the slight variance in accuracy, the two algorithms performed the same in the other weighted metrics. They both scored 0.26 precision, 0.51 in recall, and 0.35 f1-score. The next Naive Bayes algorithm did not perform much better. This algorithm was the Complement Naive Bayes approach. It scored a 0.5495 accuracy to pair with a 0.55 weighted score in each precision, recall, and f1-score. Although the accuracy was only a little better than the multinomial and Bernoulli approaches, it was much more balanced as seen in the other scores. Despite being far more balanced, the complement approach still performed better on the data of the dominant or negative class. The last Naive Bayesian approach was the Gaussian Naive Bayes method. This method far outperformed the other Bayes methods by over 15%, accurately classifying 0.7157 of the testing data. The Gaussian method measured weighted scores of 0.72 in precision and recall and 0.71 in f1-score. The Gaussian method did have a higher precision with the negative class and higher recall in the positive class. These indicate a potential bias towards the positive class. Even though the Gaussian Naive Bayes method may contain bias, it still vastly outperformed the other Naive Bayes models and is the algorithm later used in comparison with other algorithms. The accuracy comparison of the Naive Bayes models is represented in Fig. 2.

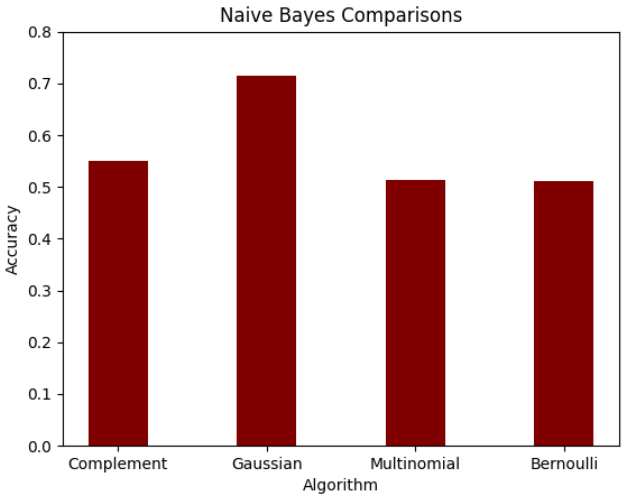

Fig. 2. Accuracy Scores for Naive Bayes classifiers.

C. Linear Regression

The next machine learning algorithm used in the analysis of the games data was linear regression. The linear regression model was trained and then used to predict the y-value. The y-value in this case was in a range from zero to one. If the predicted value was greater than or equal to 0.50 it would be predicted to be classified as a one. If the value predicted was less than 0.50, then the predicted value would be classified as zero. After training, the linear regression model accurately predicted the class of the game in the testing dataset at a rate of 0.7109. This number is about the same as the other models covered so far. The linear regression model had weighted scores in precision of 0.71, in recall of 0.71, and had an f1-score of 0.71. This indicates a fairly well balanced dataset as well as a well fit model. It is clearer to understand the strong balance of the model by looking at the precision, recall, and f1-score of the zero and one classifications separately. In this case the zero value means that the value is under the 0.85 positive review percentage, where a one class is greater than or equal to the threshold. For the zero class, the linear regression model measured a precision score of 0.72, a recall score of 0.71, and a f1-score of 0.72. The one class measured in a precision score of 0.70, a recall score of 0.71, and a f1-score of 0.70. Given how close the scores are between classes it is again possible to conclude that the data and model is well balanced.

D. Logistic Regression

Another machine learning algorithm used to predict the classification of a Steam games positive review percentage is logistic regression. The model was able to accurately predict the classification of the game at a rate of 0.7093. This rate is in line with the linear support vector classifier, Gaussian

naive Bayes, and linear regression algorithms. The logistic regression algorithm was measured with weighted scores of 0.71 in precision, 0.71 in recall, and 0.71 in f1-score. These scores can be broken down further by classification. The games that had a real classification of zero in the testing dataset were evaluated by the logistic regression classifier with scores of 0.71 in precision, 0.73 in recall, and 0.72 in f1-score. The one class in the testing data was scored with 0.71 precision, 0.69 recall, and a f1-score of 0.70. These scores of the zero and one classes show that the data is well balanced. However, there also appears a very slight bias towards the negative class of the recall and f1-scores are slightly higher towards the zero class. That being said, with how minimally the scores differ it is possible for the results to be to variation. In the case where it is not due to variation, the logistic regression model still performs well and could be considered for use alongside the other algorithms discussed. The logistic regression model is both fairly well fit and about as accurate as the other models discussed.

E. K-Nearest Neighbor

The next machine learning algorithm used was the k nearest neighbor algorithm or KNN. The algorithm was run with varying numbers of k-neighbors. Those being k=5, k=7, k=9, and k=11. For the k-nearest neighbors where k=5, the algorithm correctly classified the game at a rate of 0.6789. This was paired with weighted scores of 0.68 precision, 0.68 recall, and a f1-score of 0.68. Because the accuracy and f1-score are roughly equivalent, the algorithm can be assumed to be well balanced. However, the recall for the zero class was .09 higher than the recall for the one class. This shows that the model is more likely to choose the zero class, likely from a slightly larger number of zero classes in the training data set. The algorithm was run again with the number of neighbors set to 7. This model accurately predicted the class of the Steam game at a rate of 0.7045. This was an improvement over the score of k=5 by 0.0256, a decently improved performance. The algorithm with k=7 also scored weighted averages in precision, recall, and f1-score all of 0.70. When compared to the model's accuracy, this showed a balanced model, however once again the recall of the zero class was 0.07 higher than the one class. This was better than the k=5 algorithm but still contains a small amount of bias. The k-nearest neighbor models were k=9 and k=11. Both of these models accurately predicted the classification of the Steam game at a rate of 0.7109. This shows a marginal increase in accuracy from the k=7 model of 0.0064 or just a little over half of a percent and a fairly strong predictor relative to the other models. The k=9 and k=11 models did not perform exactly the same but very similarly. Both models measured weighted scores of 0.71 in precision, recall and f1-score. Despite the highest accuracy, the models had the highest variance in precision and recall. Both models had a 0.03 higher precision on the zero class than on the one class. Even more variance was found in the recall. The k-nearest neighbor algorithm where k=9 had a recall 0.14 higher in the zero class than the one class, while the k=11 was also higher by 0.12. This shows a fairly strong bias in the algorithm in favor of the zero class. Despite the strongest bias, the k=9 and k=11 had the highest accuracy and f1-scores out of the k-nearest neighbors algorithms. Because of this, the k=9 model was chosen to compare and contrast with the rest of the models. The accuracy of the different models is represented and exaggerated to show difference in Fig. 3, as the y-axis has been moved to 0.65.

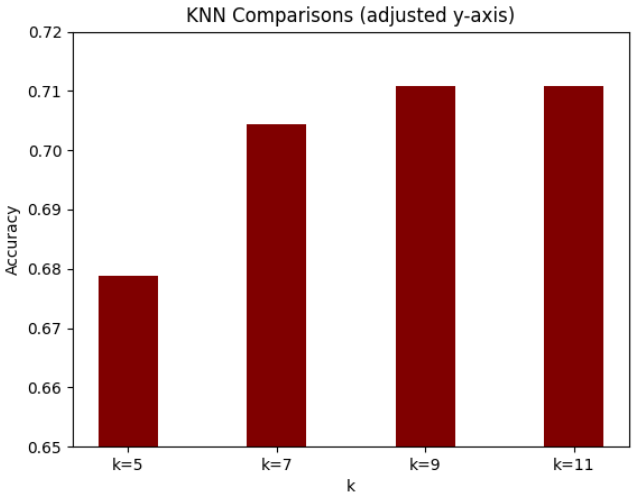

Fig. 3. Accuracy scores for knn with varying values of k. The y-axis of this figure has been altered

F. Decision Tree

The final classifier used in the evaluation on the Steam games data was the decision tree. The decision tree was run with modified data. Instead of each variable being a continuous value between zero and one, the values were given values between zero and nine based on the continuous value. This was done to better group the data in a more usable way for the decision tree [9]. The decision tree was first run without any pruning in effect. This tree earned an accuracy score on the testing data of 0.6965. The unpruned tree measured weighted scores in precision, recall, and f1-score of 0.70. The precision, recall, and f1-scores were also very similar between classes, with the zero class all being 0.70 and the one class earring scores on all categories of 0.69. This model was relatively similar to the other algorithms and was very balanced between classes. A post pruning algorithm was run to find the best path of the tree. After finding the path, the tree was run again. This post-pruned tree accurately predicted the class of the game at a rate of 0.7157. This mark was tied for the best performing algorithm. Fig. 4 contains a chart detailing the exaggerated comparison between accuracy of an unpruned tree vs a post-pruned tree. The algorithm measured weighted averages in precision, recall, and f1-score of 0.72. Although the model had a high mark for accuracy, it had more imbalanced scores in precision and recall between the zero and one classes. The model had a 0.06 higher precision for the zero class than the

one class, where the recall is 0.09 higher in the one class than the zero class. This shows that the model has a bias to predicting the one class, but was otherwise accurate.

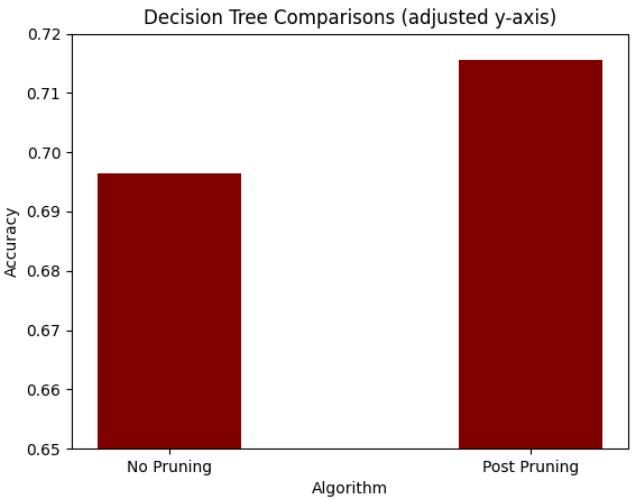

Fig. 4. Accuracy scores for unpruned and post-pruned trees. The y-axis has been altered.

G. Analysis

Each of these six models were used to predict the classification of a Steam game as having over or under an 85% positive review percentage, where under the amount was the zero class and over being the one class. In the event that multiple kinds of a similar algorithm were used, the best performing model in terms of accuracy was used. This means that the Gaussian Naive Bayes model, k-nearest neighbors where k=9, and the post-pruned decision tree were used. There were two algorithms that tied for the best accuracy of classification. Those were the Gaussian Naive Bayes and the post pruned decision tree, each accurately predicting the games at a rate of 0.7157. Following those were k-nearest neighbors with k=9, linear regression, and linear support vector classifier, all scoring an accuracy of 0.7109. Followed last by logistic regression with an accuracy of 0.7093. Despite the decision tree and Naive Bayes algorithms performing the best, all six of the classifiers accuracy fell in a range of just .0064 or just over half a percent. Given the very small variance between the accuracy's of each algorithm, there is no clear answer to which one is the most viable at predicting the correct classification for this problem, and in fact all appear to be viable based solely on performance based on accuracy. Fig. 5 is the accurate depiction of the accuracy of all of the models used. Fig. 6 shows an exaggerated model of the accuracy of the algorithms, with the y-axis starting at 0.70 and ending at 0.72. Since the accuracy's are similar, it may be desirable to look at the balance of the algorithms. Despite having the highest accuracy, the decision tree had the highest variance of precision and recall amongst all of the algorithms. However, it was the only algorithm that did not earn a f1-score of 0.71, but still only earned a 0.72. Other algorithms that had higher variance than the others were the k-nearest neighbors algorithms and the Gaussian Naive Bayes. The most well balanced algorithms in the set were the linear support vector classifier and logistic regression. These entirety of the results show that there was not much variation between the machine learning algorithms and lead to the belief that there is not a clear cut best algorithm for the problem. The post-pruned decision tree performed the best in accuracy and f1-score just barely, but had the highest variance in precision and recall of all the algorithms. Linear SVC and linear regression performed just worse than the decision tree but had much more balanced results. However, all models accurately predicted the classification of a Steam game at around 70%. Since all models were in the same ballpark, it may be possible that a variable or variables may be missing to help explain the remaining 30% of games that are misclassified. These results also point to the fact that just one simple algorithm may not be able to accurately predict results and instead using an ensemble approach could lead to better results. With the decision tree landing in the most accurate models, using an algorithm such as random forest could provide improvement to the accuracy of classification. Due to the relatively strong ability to predict the classification, this points to the fact that the features used in this experiment are decently strong predictors of how well a game is received. Because of this game developers may be able to try and tailor their games toward these metrics. With modification to the data, these algorithms may also be used to predict game success before release as opposed to predicting how well a game is received.

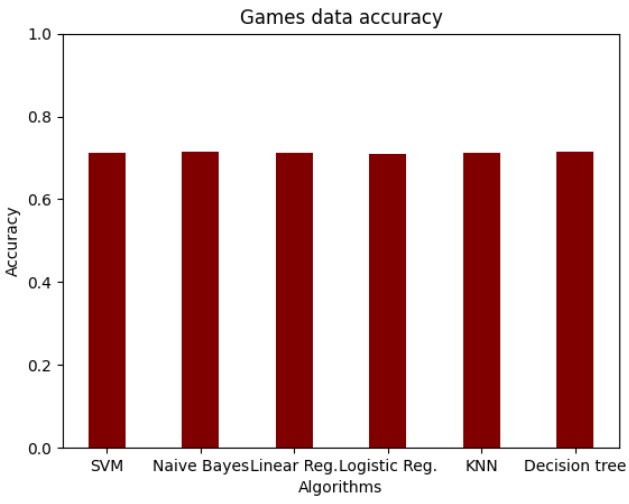

Fig. 5. Bar chart of all algorithms' accuracy.

.

REFERENCES

[1] C. Martins, "Gaussian naive Bayes explained with Scikit-Learn," Built In, https://builtin.com/artificial-intelligence/gaussian-naive-bayes

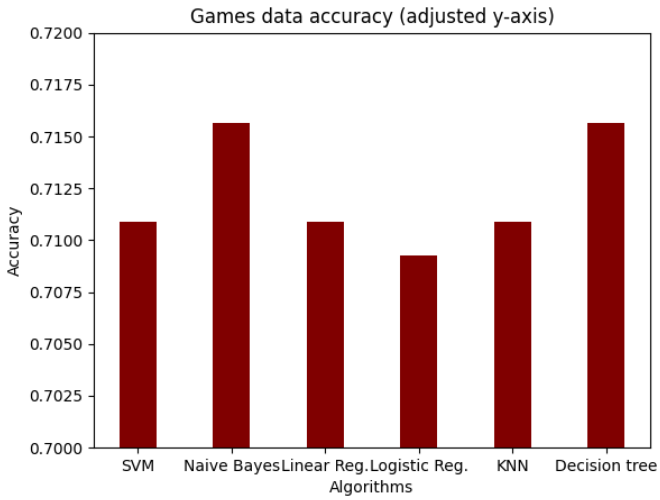

Fig. 6. Y-axis adjusted chart of algorithm accuracy.

(accessed Apr. 20, 2024).

[2] A. V. Ratz, "Multinomial NAIVE Bayes' for documents classification and Natural Language Processing (NLP)," Medium, https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6 (accessed Apr. 20, 2024).

[3] GeeksforGeeks, "Complement naive Bayes (CNB) algorithm," GeeksforGeeks, https://www.geeksforgeeks.org/complement-naive-bayes-cnb-algorithm/ (accessed Apr. 20, 2024).

[4] N. Sharma, "Bernoulli naive Bayes and it's implementation," Medium, https://medium.com/@nansha3120/bernoulli-naive-bayes-and-its-implementation-cca33ccb8d2e (accessed Apr. 20, 2024).

[5] V. Kanade, "All you need to know about support vector machines," Spiceworks Inc, https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/#: :text=A%20support%20vector%20machine%20(SVM)%20is%20a%20machine%20learning%20algorithm,classes%2C%20labels%2C%20or%20outputs (accessed Apr. 20, 2024).

[6] V. Kanade, "Logistic regression: Equation, assumptions, types, and best practices," Spiceworks Inc, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/#: :text=Practices%20for%202022-,What%20Is%20Logistic%20Regression%3F,1%2C%20or%20true%2Ffalse (accessed Apr. 21, 2024).

[7] V. Kanade, "What is linear regression?- Spiceworks - Spiceworks," Spiceworks Inc, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/ (accessed Apr. 21, 2024).

[8] "What is the K-nearest neighbors algorithm?," IBM, https://www.ibm.com/topics/knn#: :text=The%20k%2Dnearest%20neighbors%20(KNN)%20algorithm%20is%20a%20non,of%20an%20individual%20data%20point (accessed Apr. 21, 2024).

[9] N. Chauhan, "Decision tree algorithm, explained," KDnuggets, https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html (accessed Apr. 21, 2024).

[10] O. Pramod, "Decision trees," Medium, https://medium.com/@ompramod9921/decision-trees-8e2391f93fa7#: :text=max_depth%3A%20This%20parameter%20controls%20the,a%20maximum%20value%20for%20max_depth. (accessed Apr. 21, 2024).

[11] M. Baas, "Heuristics for assessing steam game reviews [part 1]," M Baas, https://rf5.github.io/2020/12/13/steam-reviews.html#: :text=A%20%25%20positive%20review%20score%20of,top%2020%25%20of%20games). (accessed Apr. 21, 2024).