CS425 Final Project

Matt Porter
Tickle College of Engineering
University of Tennessee
Knoxville, US
mporte27@vols.utk.edu

Grant Ballard
Tickle College of Engineering
University of Tennessee
Knoxville, US
dballar4@vols.utk.edu

***Abstract*—The dataset used in this project is a collection of information about thousands of Steam games. There are over 12,000 different games listed, and there are various different features, such as: the name of the game, the release date, the recommendation count on Steam, the Metacritic score, and genre(s). The goal of this project has two parts: first, it tries to define a correlation between the Metacritic score and the recommendation count of games on Steam; and second, it tries to classify a game into one of two genres based on its Metacritic score, price, and recommendation count on Steam. The first part is attempted through linear regression, though it does not end up a significant correlation. This is likely due to high concentrations of data points in certain regions of the recommendation score. The second part is attempted through random forests and AdaBoost, and it is successful relative to the random chance of choosing correctly between two choices. This means that the classifier was able to learn something from the dataset.**

I. Introduction and Motivation

The dataset we are using is a collection of video games off Steam. We chose this dataset due to a high interest and previous knowledge of Steam and video games in general. Overall, we are using this dataset to try to classify genres and determine a relationship between Metacritic score and recommendation counts. We started by using linear regression, comparing user ratings and official news outlet ratings, but after some poor results we moved on to using a random forest classifier to classify two different genres. We chose this approach due to our dataset having small variations that we needed to account for. As you can see in Section 2B, there are not very apparent distinctions between any two genres with the features we had available. Success is determined by the testing accuracy score being anything over 50%. Given that one could predict between two genres using a coin flip (50-50 split), anything more than that means the model learned something.

The remainder of this report details the dataset, the machine learning used to accomplish the goals we have laid out, and our results. The results of the linear regression were not very great and it is not a large focus of this report. On the other hand, the results of the random forest classifier were good, and there was a lot more focus on this in the machine learning section of the report. Overall, we learned a lot from this dataset and more about machine learning with these two models, but there is more work to be done to extract more information about this dataset.

II. Dataset

A. *General*

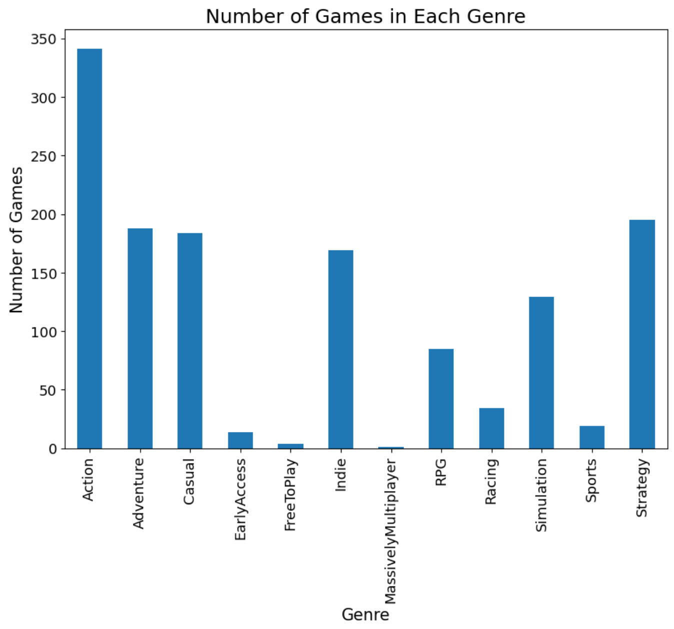
Our dataset is the “Steam Games” dataset from Kaggle uploaded by user “The Devastator”. The features of this dataset include: the name of the game, the release date, the Metacritic score, the count of steam users who recommended the game, whether the game is free or not, the genre of the game in the form of multiple genre columns containing Boolean responses, and the initial price of the game.

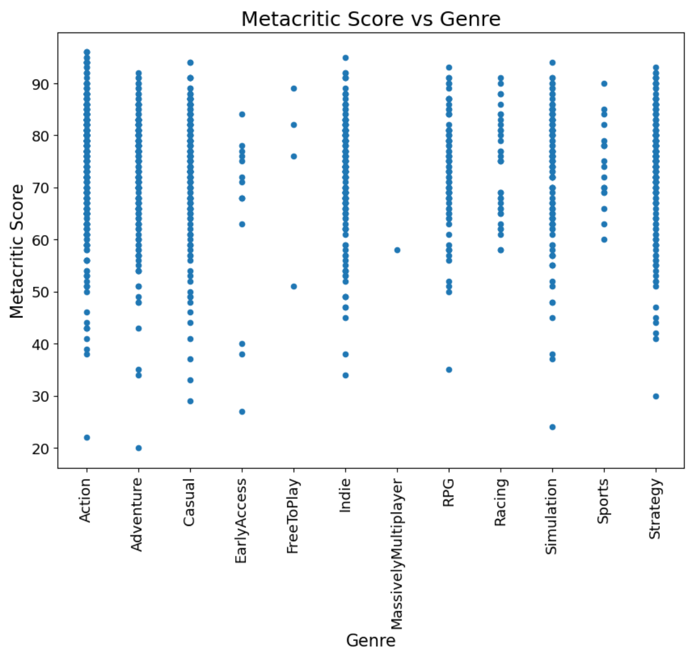
For our linear regression model, we considered the Metacritic (score) and RecommendationCount features. We chose these two features to determine if there was any correlation between how users felt about a game versus how critics did. We omitted the rest of the features because they did not have any visibly apparent linear correlations like these two features did. We also performed regularization on the RecommendationCount feature using StandardScaler(), because the values were extremely varied.

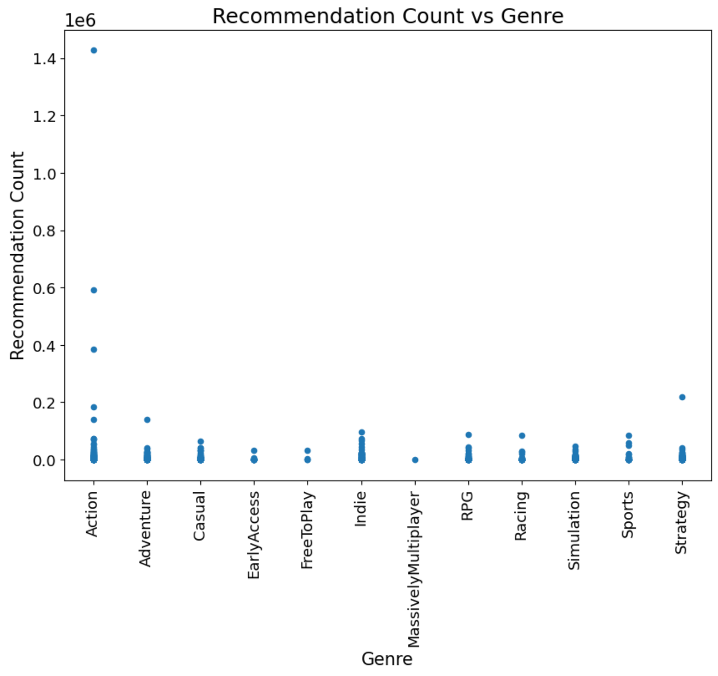
For the random forest model, we considered the genre, Metacritic (score), RecommendationCount, and PriceInitial features. We chose these features because they were the most relevant quantitative features, and we wanted as many quantitative features as possible to maximize random forest success. We omitted the ResponseName (name of the game), ReleaseDate, and IsFree features because we did not find that any of these features could have any use. Due to the way the dataset was formatted, each genre had its own individual feature column, making most algorithms identify them as individual features. However, for our work, we wanted one qualitative genre feature column. In order to do this, we wrote code that reclassified each data point to the actual genre name, rather than multiple Boolean values. We only wanted to consider one genre from each game to simplify everything. In order to not give any genre priority over another, we only considered the games that were only classified under one genre. This ended up significantly reducing the number of datapoints from over 12,000 to just 1363. However, there was a fairly good distribution among a few genres, as seen in the graph in section 2B: *Number of Games in Each Genre*.

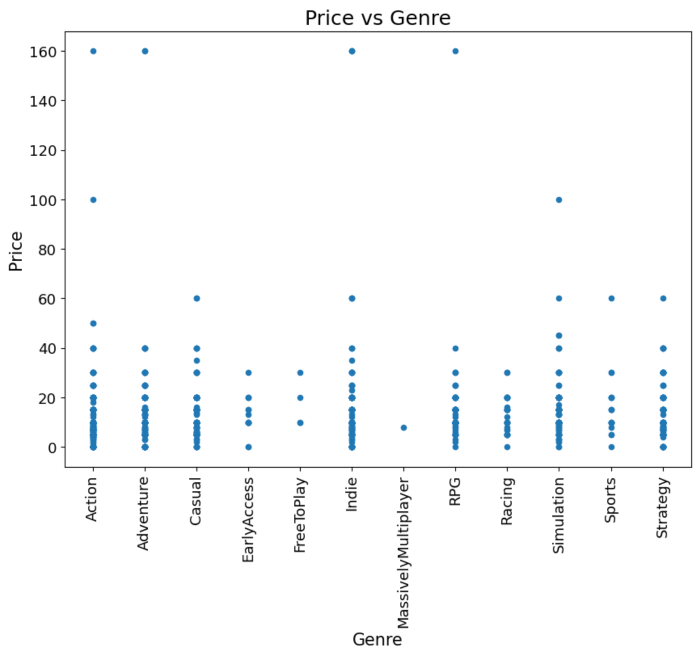
Additionally, for both approaches, we filtered out any nan values for the features we were considering for obvious reasons. We also filtered out games with a Metacritic score of zero, because that just meant it had not been reviewed on their website.

B. *Random Forest Approach*

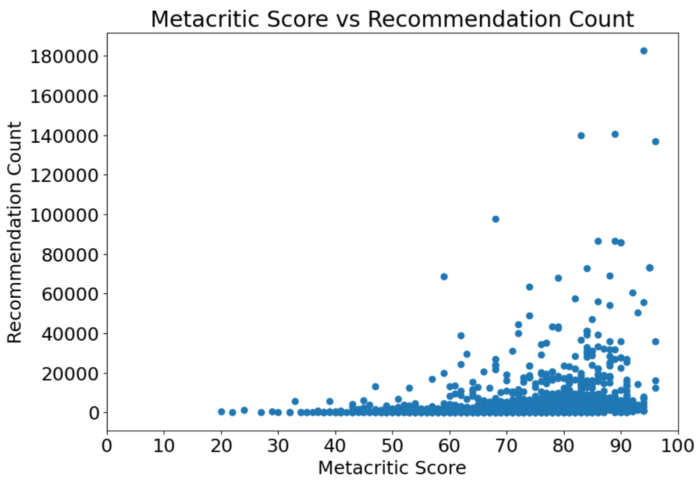








C. *Linear Regression Approach*



III. Machine Learning Approaches and Methodology

A. *General*

As stated in the introduction, the two methods that we explored for this dataset are a random forest classifier and linear regression. We also used AdaBoost in addition to a random forest classifier. There is no comparison to be made between the results of linear regression and the random forest classifier in our case, because they are using different feature sets. However, we did compare a random forest classifier by itself and a random forest classifier + AdaBoost to determine how the accuracy scores changed.

B. *Random Forest Classifier*

We decided to use a random forest classifier because we knew that it would be relatively easy to understand the results, and we wanted to be able to determine what features were important for deciding between genres. We also initially wanted to classify between more than two genres, so we could not have used any of the binary classifiers we had experience with. However, it turned out to be more difficult to work with more than two genres at a time, so we just evaluated pairs of genres. We also limited the genres to just the top five in terms of number of games (action, adventure, casual, indie, strategy), and we tried to classify each combination of pairs of those five genres.

Our definition of success for this classifier was anything greater than (1/n * 100)% accuracy score, where n is the number of genres to classify, because this is the minimum accuracy for random chance. For any scores higher than this value, it would mean that the model had learned something. For example, using two genres, the random chance accuracy would be ½ or 50%. Thus, anything greater than 50% would be considered successful.

The hyperparameters we considered for this model were: number of estimators, max depth, and minimum samples split. For each set of genres, we created a new data frame containing only those genres, we used KFold cross validation with 5 folds to determine the best values to use for each hyperparameter from the following:

```
"n_estimators": [100, 200, 300, 400, 500],
        "max_depth": [5, 10, 15, 20, 25],
        "min_samples_split": [2, 5, 10, 15, 20].
```

Then, we created a single random forest classifier based on those best performing hyperparameters. We also used AdaBoost on this classifier to compare our calculated metrics. We did not explore the hyperparameters of AdaBoost, but we arbitrarily set the number of estimators to 100 and the learning rate to 1.

The metrics we used to evaluate this classifier were the accuracy scores for the training and testing sets. These were calculated using Scikit-learn's accuracy_score function. As mentioned previously, we used a single random forest classifier and compared it with the AdaBoost version of that classifier.

C. *Linear Regressor*

We decided to use a linear regressor because graph 2C: *Linear Regression Approach* appeared to have a somewhat linear correlation between the Metacritic score and the recommendation count. However, we did not take into account the concentration of data points between 0 and 20000 recommendation counts. As you will see in the results section, the metrics of this model were very bad, and we decided to focus our efforts on the random forest classifier instead of exploring very many hyperparameters and/or other methods of improving the results.

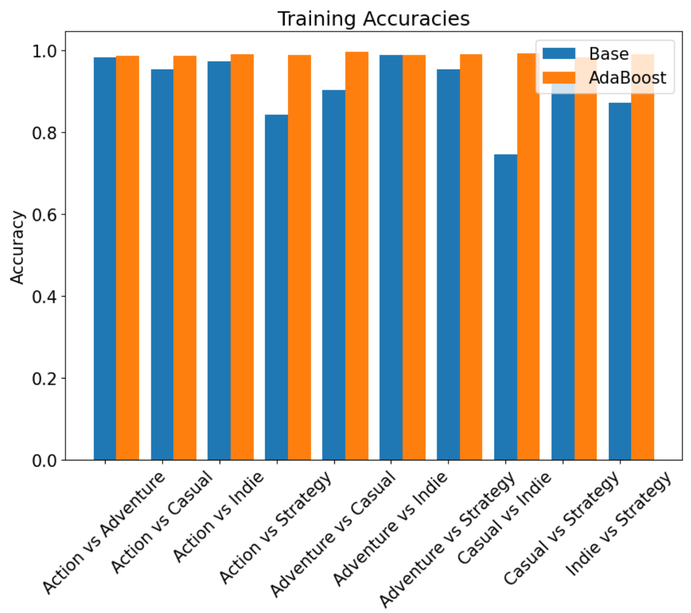
We did compare the two different types of linear regressors, ridge and lasso, and we evaluated different values for the hyperparameter alpha. We once again used kfold cross validation to determine the best value for alpha from the following: 0.1, 1, 10, 100, 1000, 10000.

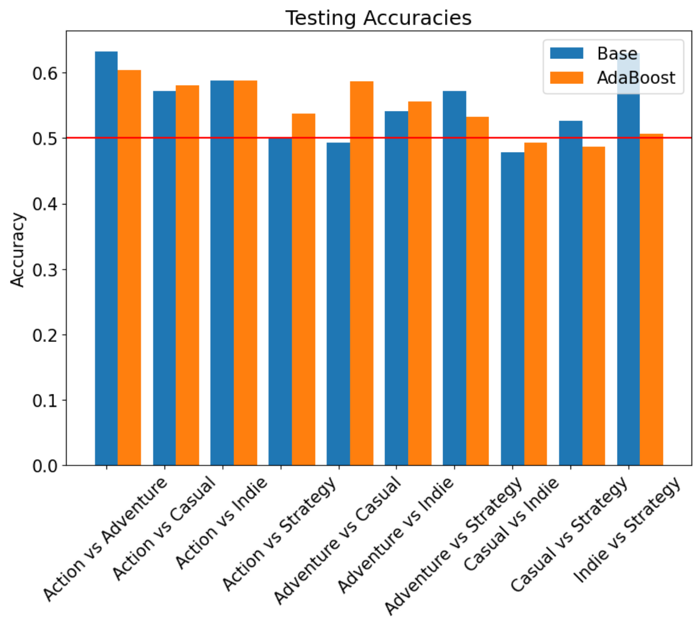
The metrics we used to evaluate the resulting regressor were the R^2 score and the mean squared error. These were calculated for both the training and the testing sets using Scikit-learn's mean_squared_error and r2_score functions.

IV. Results

A. *Random Forest Classifier*

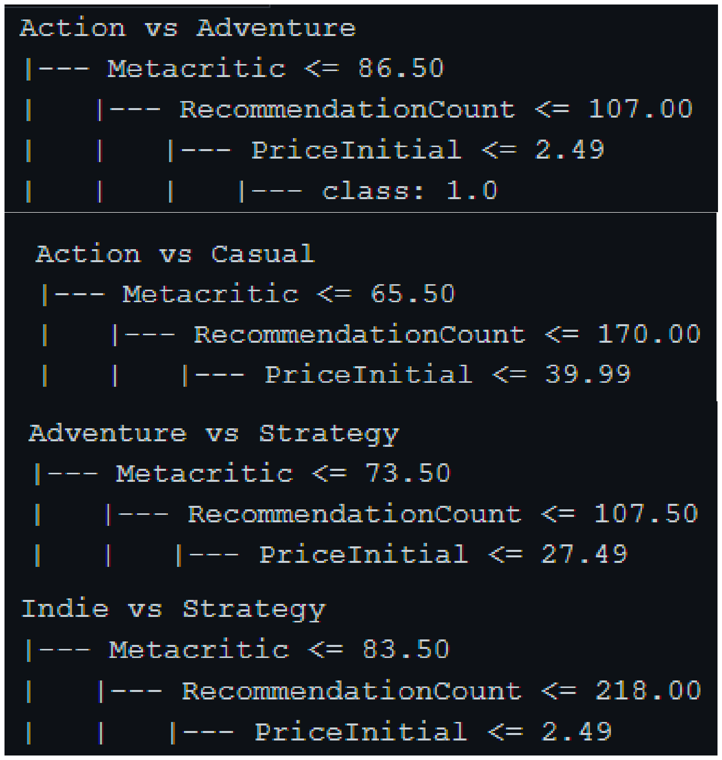
The results of the random forest classifier approach were good in terms of our definition of success. Almost all of the base classifiers (i.e. without AdaBoost) had testing accuracies that were higher than 50%. With AdaBoost, most of the accuracy scores improved, and all of the genre classifications had accuracy scores higher than 50%. Strangely, however, some of the scores decreased after using AdaBoost. We didn't have enough time to really investigate this, but this was probably due to overfitting on the training set. Many of the training accuracies were already in the mid-90%s, and AdaBoost improved many of them. This may have been too much of an increase in training accuracy that led to worse testing accuracy. The results for the different pairs are shown in the following graphs.





Note that the horizontal red line in the testing accuracies graph represents the random accuracy score of 50%. There are many pairs that are over this line, which means that they were able to learn something. It is interesting that the random forests (and sometimes even Adaboost) sometimes performed worse than random accuracy.

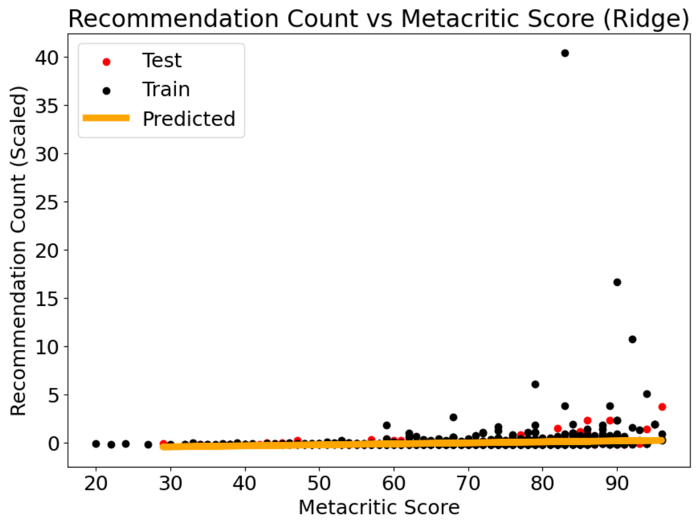
The following screenshots are from the first decision trees of every random forest for a few of the genre pairs. As you can see, the first feature that they all split on is the Metacritic score. This seems to imply some connection between genres and Metacritic scores, but further research should be done to confirm such a correlation.

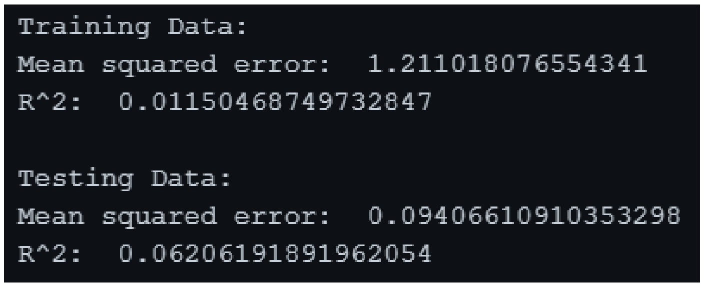

```
Action vs Adventure
|--- Metacritic <= 86.50
|   |--- RecommendationCount <= 107.00
|   |   |--- PriceInitial <= 2.49
|   |   |   |--- class: 1.0

Action vs Casual
|--- Metacritic <= 65.50
|   |--- RecommendationCount <= 170.00
|   |   |--- PriceInitial <= 39.99

Adventure vs Strategy
|--- Metacritic <= 73.50
|   |--- RecommendationCount <= 107.50
|   |   |--- PriceInitial <= 27.49

Indie vs Strategy
|--- Metacritic <= 83.50
|   |--- RecommendationCount <= 218.00
|   |   |--- PriceInitial <= 2.49
```

B. *Linear Regressor*

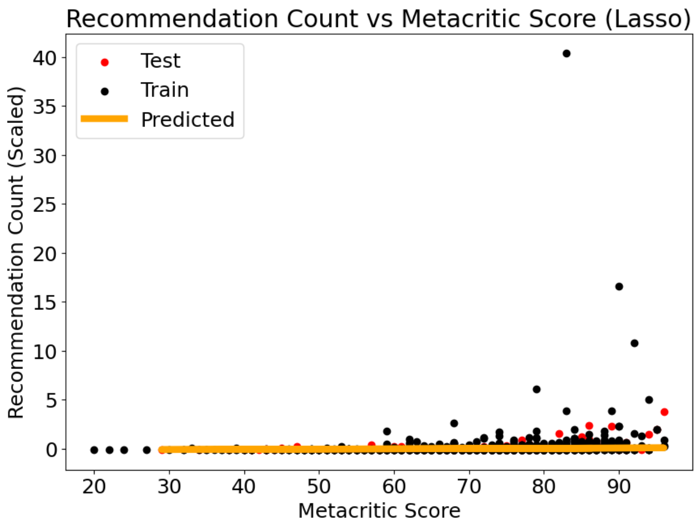
The results for linear regression were very poor. Our initial attempts using linear regression resulted in very low R^2 scores and mean_squared_errors. At one point, the testing R^2 score was negative, which was bizarre but apparently possible due to how Scikit-learn calculates it.

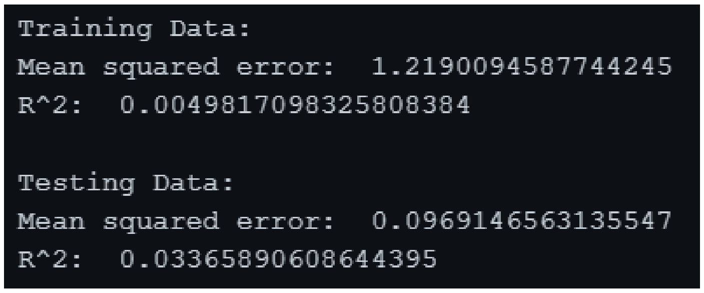
At this point, we began incorporating different methods to improve the performance in any way possible. We eventually landed on performing regularization on the y-axis and comparing the Ridge and Lasso regressors using different values of alpha. The results of these two different regressors are shown in the following two graphs and accompanying images.





```
Training Data:
Mean squared error:  1.211018076554341
R^2:  0.01150468749732847

Testing Data:
Mean squared error:  0.09406610910353298
R^2:  0.06206191891962054
```





```
Training Data:
Mean squared error:  1.2190094587744245
R^2:  0.0049817098325808384

Testing Data:
Mean squared error:  0.0969146563135547
R^2:  0.03365890608644395
```

As you can see in the above graphs, the linear regressor did not really follow the upward trend very much because there were so many data points in the low range of the recommendation count. Note that these are after regularization, so the recommendation count is not entirely accurate for this graph. Overall, the R^2 scores were really low. The highest score was the ridge regressor's testing R^2 score. Interestingly, the training scores for both of these regressors were lower than their respective testing scores. This is likely due to the excessive number of points in the lower region of the recommendation count. Since the testing data is only 20% of the entire dataset, this lower section of data probably did not affect the results as heavily. Regardless, the results are not good in any form, and it is difficult to form any conclusions from them.

V. Discussion, Conclusion, and Future Work

The linear regression results were very poor, likely due to a high concentration of data points in one area. It is safe to say that linear regression does not perform well for datasets that are not well distributed. The random forest results were relatively good. Obviously, it would be ideal to have an accuracy score closer to 100%, but, as stated already, anything higher than 50% was a success in our view. It was nice to see that AdaBoost improved most of the accuracy scores compared to our original results. However, the higher training accuracies from AdaBoost sometimes led to overfitting, which hurt the testing scores.

If we had more time, we would like to further explore Adaboost and other ensembling methods along with random forests. It seemed promising, but we did not have time to explore all the different parameters that went along with it. The linear regression did not seem very feasible, so we probably would not continue any further with that. In terms of other machine learning methods, it would be interesting to see how a neural network could classify genres based on the same features as the random forests. One question that was brought up during this project was: "Is there any correlation between the Metacritic scores and the recommendation counts within genres?" For example, do the recommendation counts show any biases towards certain genres or do the Metacritic scores show any biases towards certain genres? Another part of the dataset we would have liked to explore is whether or not a game is free and if that has any correlation to other features.

VI. Contribution of Team Members

Matt wrote most of the code for the project, based on some of the ideas from Grant. He also wrote the abstract, machine learning methods, results, and conclusion sections.

Grant wrote the introduction and dataset sections of the report, and he made most of the final powerpoint for the recorded presentation. He also wrote the basis for the code the transformed the multiple genre features into one, as well as did some of the data visualization code.

The in-class presentation was not very much work in the long run, but we did an equal amount of work in that.