**Predicting Video Game Popularities**

**with Steam Games Dataset and Machine Learning**

**Group 4: Microl Chen, Peter Jeong, Jonathan Kim**

**Abstract**

The global gaming market is increasing in size, and the investment on individual games is also increasing. As a result, the success of each game is now crucial for the developing company’s fate. Therefore, predicting the success of a game became a very important task. Many other predictions were made on video game sales using different methods. Our project aims to produce a model that will accurately predict the popularity of a game based on several features. Popularity of a game is a more accurate estimator for a game’s success as it can also be an estimator for the success of the next game. We used the Steam dataset from Kaggle to train 3 different machine learning models. After training, we tested our models with the test data, and calculated the accuracy and the macro-average AUC for each model. The random forest model turned out to be the most reliable, with limitations as predictions for higher numbers of owners is unreliable.

**Introduction**

In 2022, the global gaming market size was valued at $249.55 billion and is expected to grow double its size (~$665 billion) by 2030 [1]. Video games are becoming more popular than ever before, and corporations are investing millions of dollars on market research. The ability to gauge a video game’s success before a substantial amount of effort is invested into its development prior to the open market will give game developers a strong edge in their competitiveness, saving them both money and time. Given this nature, a substantial amount of work and research has been conducted to tackle this very purpose. Yet there is still much to be improved – new games are always being released, and new trends are coming out every other week. Additionally, the majority of the previous research mainly focuses on the sales rate of the game, in order to track the video game’s success. However, sales rate can be an unreliable index since sales of video games may fluctuate depending on the price of the game; it may show the total profit but has inaccuracy to track the popularity of the product in the gaming market. Previous models trained in a bygone era may struggle to keep up with the diversity of games in today’s dynamic gaming market. With a relatively newer and larger dataset (web scraped approximately 4 months ago) and a novel selection of features, we hope to find the same success previous models have had for this new generation of diverse video games.

**Background**

Two different studies done previously were used as a basis for our project. The first study is done by Joe Cox [2], using data he collected [3]. In this study, Cox uses games released until 2011, and uses their lifetime sales as the dependent variable. Then using several different features as the covariates, he conducted a linear regression analysis to estimate the important factors that affect video game sales. Through this research, he tries to find what are features that make a game a blockbuster title. According to his study, three major factors were important for a game to be a blockbuster. First is the publishers. He finds that more well known publishers are likely to make blockbuster titles. Second is the platform the game is published for. Games built for popular platforms are more likely to result in higher lifetime sales. Third factor is quality, which is represented by the review scores. In other words, games which received higher critic scores were more likely to succeed. Cox’s study provided the inspiration for our project. However, since he was using linear regression, his methods were not applicable for our project, since our target label was ordinary.

The second study is done by Keerthana Bodduru [4], using 11 variables and 500 samples with a combination of categorical and numeric variables. She conducted a study by using multiple prediction models, such as linear regression, supported vector regression, random forest, and decision tree, to figure out the best model to predict the sales rate of the video game. According to the study, she figured out that the random forest model had the highest accuracy (96.05%) among the other machine learning models, based on the formula of (TP + TN) / (TP+TN+FP=FN). Bodduru’s study is similar to our project in that she uses machine learning algorithms to predict video game sales. While the method is similar, we differentiate our project by setting a different target label and using a bigger dataset.

**Methods**

For our project, we decided to explore three different machine learning algorithms. Those were: K-Nearest Neighbors, Decision Tree Classifier, and Random Forest.

*K-Nearest Neighbors (KNN)*

The K-Nearest Neighbors classifier provides a way of classifying samples based on feature similarity. There are many different games that have similar genres or categories in our dataset, and therefore, we believe KNN would be a good method to predict the different classes, as similar games are likely to attract similar groups of gamers, and therefore have a similar size of owners.

*Decision Tree Classifier (DTC)*

The Decision Tree Classifier is a method that follows a tree to classify samples. We chose DTC since using DTC will allow for robust classification of multiclass datasets. This classifier also allows us to estimate the hierarchy (which features are more important when classifying) of the features.

*Random Forest Classifier*

Random Forest Classifier is the aggregation of multiple decision trees. While the decision tree itself can be accurate and reliable, it is prone to high variance. Therefore, we decided to use random forest in hopes to obtain an accurate model with lower variance.

**Models**

*Data*

We used the Steam Games Dataset from Kaggle, gathered by user Maxwell [5]. This dataset includes information about 71716 games from Steam, the most popular electronic software distribution service for games. This list of games was reduced to approximately 66,000 after missing values were removed from the dataset.

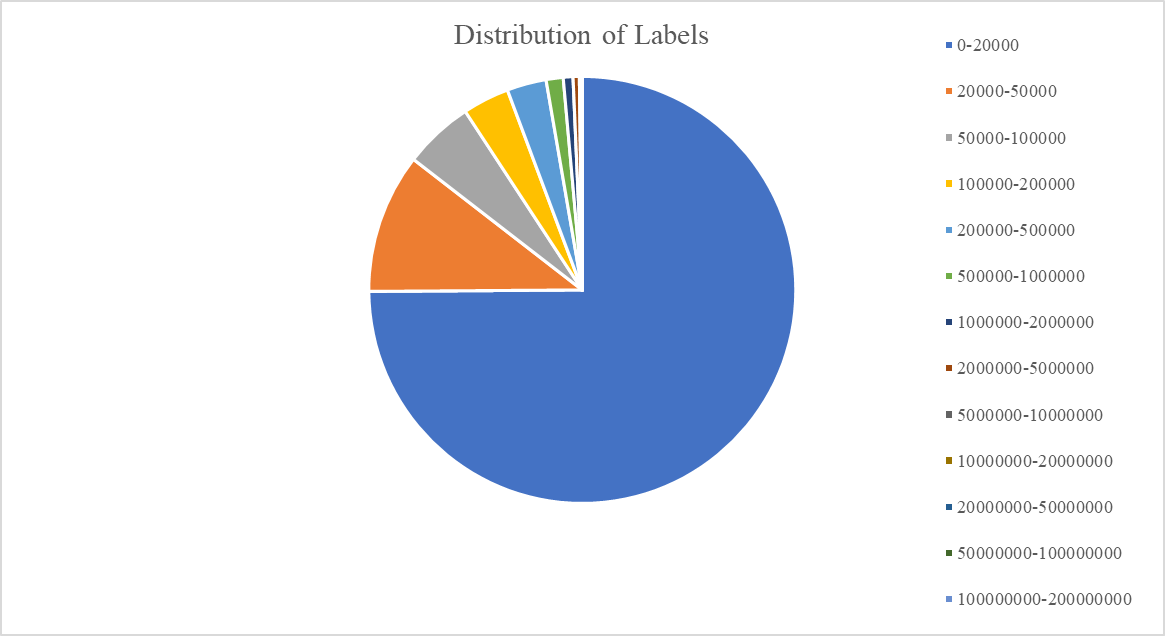
The dataset includes 39 columns, some of which are identification columns such as application ID or game title. For the purpose of our research, we excluded some columns, including game titles or publication information, reducing the number to 20 columns. These features are either completely irrelevant or not practical – any information you wouldn’t have if you didn’t already have the target label. Our target label was the “Estimated Owners” column, which is the interval of estimated number of users who own the game. Figure 1 shows the distribution of the different labels.

Figure 1

The original data has a skewed distribution of target labels. This is not unusual, as Steam has a very large number of games on sale, and a lot of them are from small or unknown developers. Inevitably, lots of games are estimated to have a very small number of owners or none at all.

*Data Encoding*

While many of the columns were integer values, some were unusable directly. Therefore, we encoded the values to be usable for our models. Table 1 shows the changes we made to the values. Observing the data, we found there was only 1 case of target label 12. Since it was only 1 case, an accurate prediction for that label was impossible. Therefore, we removed that sample from the dataset.

|  |  |  |
| --- | --- | --- |
| Column Name | Method | Values |
| Estimated Owners  (Target Label) | Converted to ordinal variables, ranging from 0 to 12. | Integer  0 to 11 |
| Supported Languages | Converted to the number of supported text languages | Integer |
| Supported Full Audio | Converted to the number of supported full audio languages | Integer |
| Windows, Mac, Linux | Converted to binary classification on whether the game supports the respective OS | Integer  0 or 1 |
| Category | Converted to binary classification on whether the game supports multiplayer. | Integer  0 or 1 |
| Genre | Converted to the number of genres the game belongs to | Integer |

Table 1

*Feature Selection*

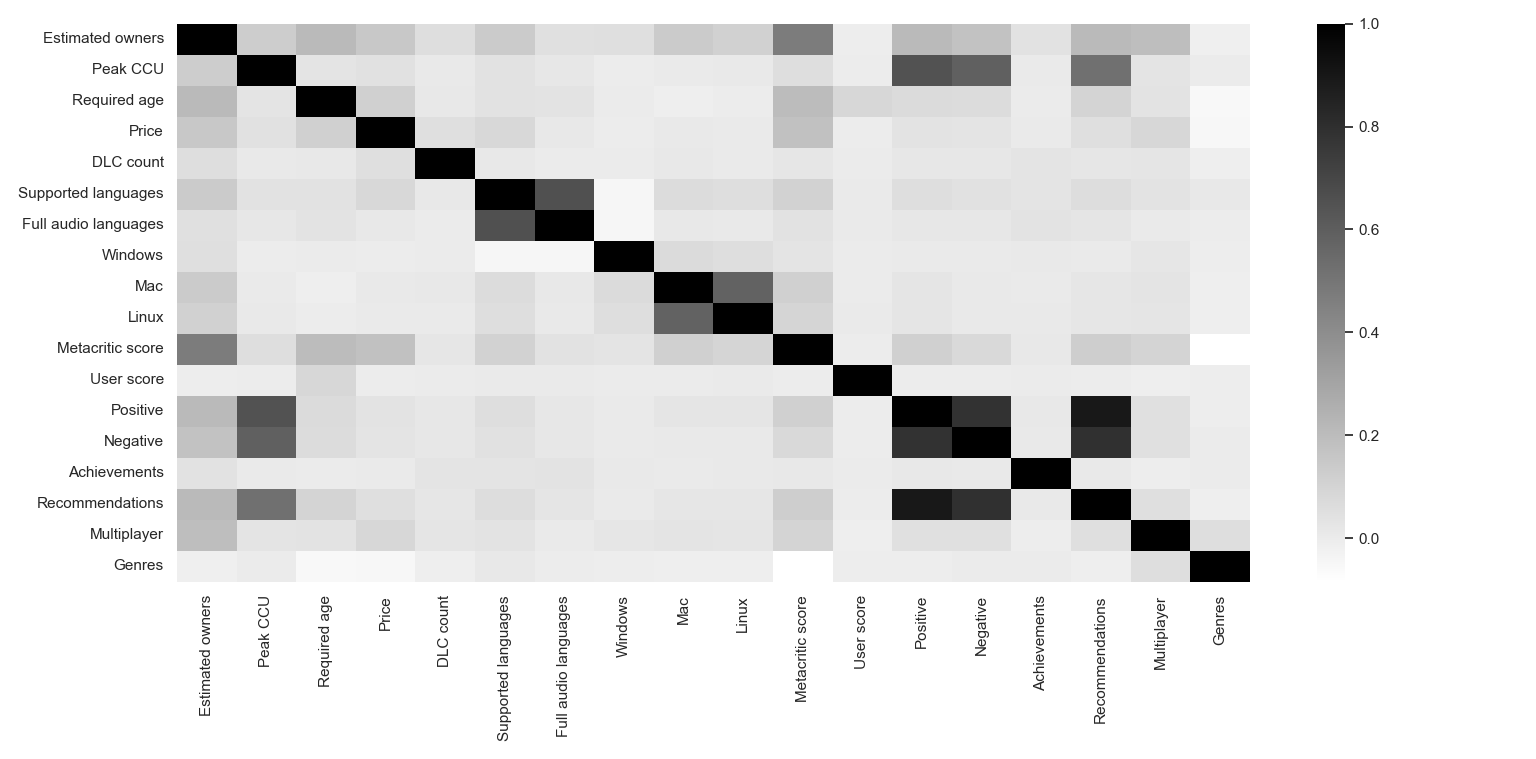
While the selected 20 columns or features were all related to the game, we could not use them all, as some may not be correlated at all with the target label, and some may be correlated with each other. Therefore, we used Pearson correlation to analyze the relationship between the features. Pearson correlation is mostly used for continuous variables. While our target label is not continuous, it is an ordinal variable, which shares some characteristics with continuous variables. Thus, Pearson correlation will provide some insight about the relationship. Figure 2 shows the heatmap for the correlations.

Figure 2

As the heatmap shows, there was a very low correlation between the target label and the features. We had to choose a low threshold, which was 0.1. From the result, we selected 11 features: Peak CCU, Required Age, Price, Supported languages, Mac, Linux, Metacritic, Positive reviews, Negative Reviews, Recommendations, Multiplayer.

*Hyper Parameter Tuning*

After a basic model was tested and built, we used cross validation grid search to exhaustively evaluate each parameter with relevant values with respect to accuracy. Below are the optimal values for each parameter. In general, we took a standard approach to determine which values should be tested with some exceptions. For the k-nearest neighbors classifier, Manhattan and Euclidean distances are popular metrics. The range of k is usually determined as [1 to sqrt(#samples)]; we had approximately 66,000 samples, so an upper bound of 250 was ultimately selected. In the same manner, the values for criterion, max-depth, minimum-leaf-samples, n-estimators (number of trees), and max-features for the decision tree and random forest classifiers were also standard “trial” values. Unfortunately, there isn’t an “one value fits all” standard for these parameters, so we simply tried different values until we identified a test accuracy higher than our k-nearest neighbors classifier.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Relevant Parameters | Best Parameters | Best Test Accuracy |
| KNN | Metric: [manhattan, euclidean]  K: [1- 250] | manhattan  9 | 0.7709 |
| DTC | Criterion: [gini, entropy]  MaxDepth: [5, 10, 15, 20]  MaxLeafSample: [1, 2, 4] | gini  10  2 | 0.8001 |
| RF | Criterion: [gini, entropy]  MaxDepth: [5, 10, 15, 20]  MaxLeafSample: [1, 2, 4]  NEstimators: [5, 10, 15, 20]  MaxFeatures': [sqrt, log2] | gini  10  4  20  ~3 (√11) | 0.8069 |

Table 2

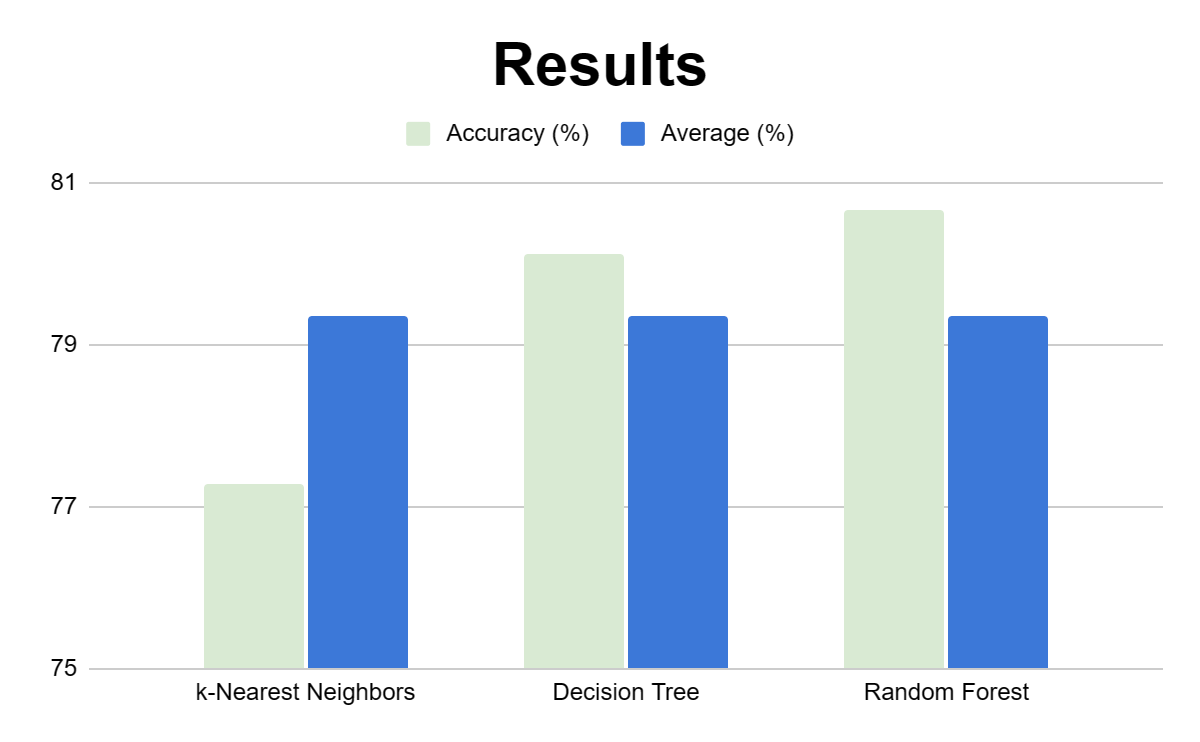
*Results*

Figure 3

With the optimal parameters, each model was built with the complete 80% training split and tested on the 20% test split and evaluated with simple accuracy. The accuracies were reflective of the hyperparameter validation scores with the k-nearest neighbor, decision tree, and random forest models resulting in accuracies of 0.7728, 0.8013, and 0.8065 respectively.

However, as it was referenced earlier, the ground truth data is skewed since steam is known to host smaller games with little or no owners – hence we recognize that this would not be the optimal approach to express the accuracy of the three models. To further evaluate the model, we decided to look at an “one-vs-rest” AUC curve.

Table 3 shows the one-vs-rest ROC AUC results for each label. As the result shows, KNN and DT have similar AUC values, which are relatively high and reliable. However, they have an AUC value of 0.5 when predicting for the highest label. This is the same as the chance level, which means the prediction is not reliable for a higher number of owners. On the other hand, the random forest model has higher AUC over all target labels compared to the other two models. However, AUC value for labels 8 and 11 is 1.0, which means there is a perfect prediction made. This is very unlikely, but the model is still reliable for most labels.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ROC AUC | | |
| Label | KNN | DT | RF |
| 0 | 0.86 | 0.9 | 0.91 |
| 1 | 0.7 | 0.78 | 0.82 |
| 2 | 0.73 | 0.82 | 0.88 |
| 3 | 0.77 | 0.86 | 0.91 |
| 4 | 0.83 | 0.89 | 0.95 |
| 5 | 0.84 | 0.89 | 0.97 |
| 6 | 0.89 | 0.89 | 0.97 |
| 7 | 0.91 | 0.91 | 0.99 |
| 8 | 0.82 | 0.8 | 1 |
| 9 | 0.92 | 0.79 | 0.96 |
| 10 | 0.86 | 0.86 | 0.93 |
| 11 | 0.5 | 0.5 | 1 |

Table 3

Once we gathered the one-vs-rest ROC AUC for each label, we aggregated them and calculated the macro-average AUC. This approach results with the following:

KNN One-vs-rest Macro Average AUC: 0.80065

Decision Tree One-vs-rest Macro Average AUC: 0.81054

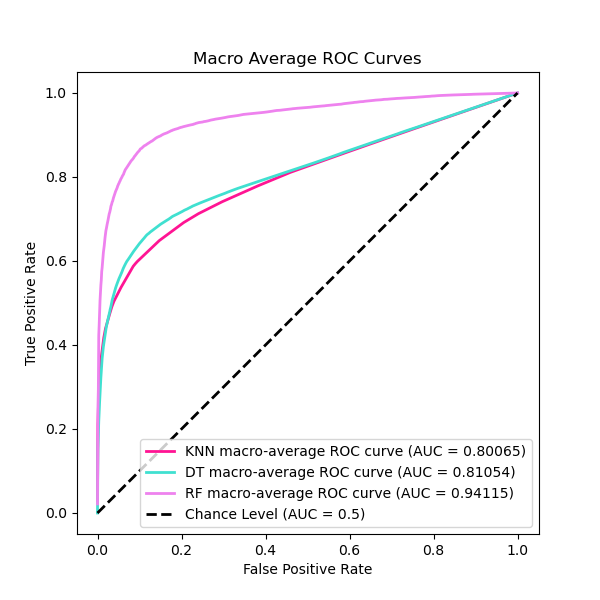
Random Forest One-vs-rest Macro Average AUC: 0.94115

Figure 4

Figure 4 shows the ROC curve for the macro-average AUC values. It demonstrates that random forests perform much better than the other two models.

**Discussion**

Our model contributes to the research on video game statistics through utilizing different machine learning algorithms. It is different from many previous research, as mentioned above. Our target label was the estimated number of owners in intervals, which is very different from the sales. By predicting the number of owners, video game developers can gain more insight on how popular their games will be. On the other hand, sales may be inflated, as not only the number of purchases, but the price of each video game also affects the total sales. Popularity is more important when determining the success of a certain video game because owners of a game are also likely to purchase future games that are made by the same developers. Therefore, our model provides insight which previous studies have not provided.

We have tested three different models-K nearest neighbors classifier, decision tree classifier, and random forest classifier. By calculating their accuracy and the ROC AUC, we have concluded that the random forest model provided the best predictions. This is expected, as the random forest is an aggregation of results from several decision trees.

Our models have some notable limitations. As mentioned previously, while our models successfully predict lower range of labels, they are nearly random chance predictions when they are predicting higher range of labels. This result is happening because compared to the number of lower target labels, the higher target labels are marginal. For an improved model, a data set with equal distribution of each label would be required.

**Contributions**

Microl Chen

* Discussion over the dataset and which columns to be initially picked
* Codes for hyperparameter tuning
* Writing the results sections with graphs and introduction

Peter Jeong

* Discussion over the dataset and which columns to be initially picked
* Background and previous studies research
* Codes for feature selection
* Writing the backgrounds section

Jonathan Kim

* Discussion over the dataset and which columns to be initially picked
* Codes for dataset cleaning and encoding
* Codes for model training and testing
* Writing the abstract and methods section

**References**

Link to codes and dataset:

[https://github.com/JKim0212/CS334-Machine-Learning-Project](https://github.com/JKim0212/CS334-Machine-Learning-Project212/CS334-Machine-Learning-Project)

[1] “Gaming Market Size, Share & COVID-19 Impact Analysis,” fortunebusinessinsights.com. <https://www.fortunebusinessinsights.com/gaming-market-105730>(Accessed Dec. 10, 2023)

[2] J. Cox, “What Makes a Blockbuster Video Game? An Empirical Analysis of US Sales Data,” *Manage. Decis. Econ,* vol. 35, pp. 189-198, April 2013.

[3] A. Corey, “Video Games CSV File From the CORGIS Dataset Project,” corgis-edu.github.io. <https://corgis-edu.github.io/corgis/csv/video_games/?fbclid=IwAR1rDafQp0_EWaJnxztsUP0MMuy-CsbnTFz0sfjhtu9_K_mmk325vU67OcY> (Accessed Nov. 13, 2023)

[4] K. Bodduru, “Sales prediction on video games using Machine Learning,” *Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 6, pp. 326-33, June 2019.

[5] “Sales Prediction on Video Games using Machine Learning”

Maxwell, “SteamGames (71k games).” kaggle.com. <https://www.kaggle.com/datasets/mexwell/steamgames/data> (Accessed Nov. 13, 2023)