Predictive Model for Video Game Sales

Nikai Lambert
Brown University
DATA1030 Final Report
Repo Link:
https://github.com/nlamber2/data1030_project.git

1. INTRODUCTION

The machine learning problem that guided my project is whether or not we can predict how many copies a video game could sell, given video game data from 1980 to 2020. The target variable for this product is Global Sales, which is a continuous variable that describes the total number of copies (in millions) a game sold. That makes this a regression problem. In this dataset, there are 16 features with 16,719 entries.

Among the 16 features, there are 5 continuous features that relate to sales in different aspects. The NA_Sales, EU_Sales, JP_Sales are sales as they relate to North America, Europe, and Japan respectively. The Other_Sales are the sales from all other regions in the world and Global_Sales is the total of all four previously mentioned sales types. There are 4 continuous features (User_Score, User_Count, Critic_Score, and Critic_Count) that tell the ratings of the games from both critics and users. This data was retrieved from a web scrape of Metacritic. The remaining features are categorical characteristics of the game themselves, with information including the Platform, Genre, Publisher, Release Year, and the Developers. These features, including the sales numbers were retrieved from a web scrape of VGChartz, a website that has a lot of data regarding video games.

Although this dataset was available on Kaggle, the previous work on it was all Data Analysis.

2. EXPLORATORY DATA ANALYSIS

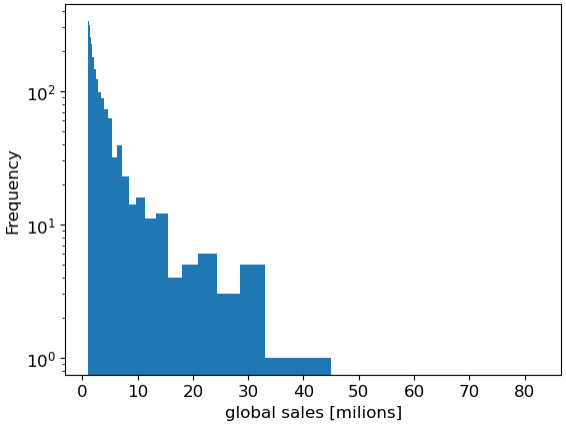

Fig 1. Histogram of the frequency of global sales numbers using log bins.

Fig. 1 shows that given our target variable of Global Sales, there is a positive skew, where a lot of the data is focused towards zero. In this dataset there are only 2,076 entries that have sold over 1 million copies. The remaining 14,000 games have sold less than 1 million. Its not too common for games to sell millions of copies, so this skew is justified.

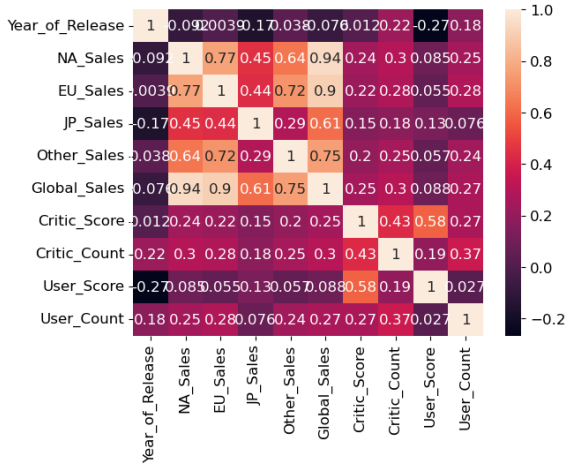

Fig. 2. Heatmap of the correlation between numeric features.

In Fig. 2, there is a noticeable correlation between the 5 sales features. The highest positive correlation is between the North American Sales and Global Sales, with European sales, Other countries' sales, and the Japanese sales following with decreasing correlation. There is a low correlation between the User Scores and the games' global sales, but while observing Fig. 3, I noticed that there may be more to this low correlation.

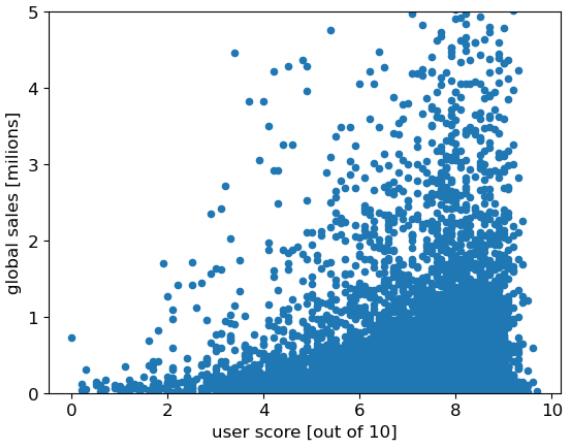

Fig. 3. Scatterplot of User Score and Global Sales, with a limit of 0 to 5 million sales.

As the user score is closer to 10, there is a higher number of sales. Of course, if more people like a game and feel its good to play, then more people would tend to buy it.

The missing data in my dataset is focused in the last 6 columns, which are mostly the Metacritic ratings, where 39-55% of the data is missing. Excluding column indexes 4 and 14 (Publisher and Developer), these missing values are all continuous features.

3. METHODS

a. Splitting

Before I split my data, I had to deal with issues that would make preprocessing tricky. First, I had to get rid of the 'Names' feature, since it would add an extreme amount of features to the preprocessing process. There are also two features that would unnecessarily clog up our features in preprocessing: Publisher and Developer.

Before trimming down its results, Publisher had over 1000 unique values, and Developer had a few hundred. This would add a lot of extra data points to preprocessing, so in order to limit the size of the features after doing so, I changed Publishers with less than 100 games and Developers with less than games to 'Other.'

This is an IID dataset that doesn't show group structure, so to split the data, I used a 60-20-20 split between the training, cross validation, and testing groups. Since there is no group structure in this data set and the machine learning question is a regression problem, a simple split like this should suffice.

b. Preprocessing

On my categorical features, I used the OneHotEncoder since they don't appear to be ordinal features and lack the ordering that those have. On the continuous features, I used the StandardScalar since there is no defined maximum for my features. After preprocessing, the training group's shape was (10031, 140), while both the cross validation and test sets had shapes of (3344, 140).

c. ML Pipeline

The Machine Learning Algorithms I used for this were XGBoost, L1 and L2 Linear Regression with regularization, and Random Forest Regression. However, because of my dataset containing missing data, I decided to first run XGBoost before doing anything further. In order to run more ML algorithms, I decided to use sklearn's iterative imputer and do Multivariate Imputation to fill in the missing values of the dataset.

After this, I was able to run Lasso, Ridge, and Random Forest Regression, as well as running XGBoost again with the missing data dealt with.

For lasso and ridge, I tuned the alpha values to .0001, .001, .01, .1, 1, 10, and 100 with the maximum iterations being set to 100000. For random forest regression, I tuned max depth with values of 1, 3, 10, 30, and 100 with max features being set to .25, .5, .75, and 1. In both instances of XGBoost, I tuned the learning rate and max depth. The learning rate was tuned to .03, .05, .10, .20, and .025, while the max depth, like in random forest, was tuned to 1, 3, 10, 30, and 100.

When running the algorithms, I ran 5 random states and ran all possible combinations of my tuned hyperparameters.

Since I am working on a regression problem, I used root mean squared error as my evaluation metric and took the mean and standard deviation of the scores from my best models.

4. RESULTS

	mean of test RMSE	std of RMSE	# of std away from baseline
XGBoost	0.552413	0.257051	0.038532

XGBoost (imputed)	0.535523	0.265967	0.100745
Lasso	1.771265	0.307290	3.934227
Ridge	0.012013	0.001079	510.193416
Random Forest	1.353118	0.259527	3.047088

Fig. 4. Table of results from ML algos

The baseline score of my test data was 0.5623. When comparing the data to my baseline, it appears that XGBoost and XGBoost with imputed data have the closest scores. However, Ridge is shown to be my best model as it has the lowest mean root mean squared error of 0.012013, with a standard deviation of 0.001079. However, it has 510 standard deviations above the baseline which is pretty strange. When looking at the global importance of features, I noticed that the predictive model followed the trend shown in Fig. 2.

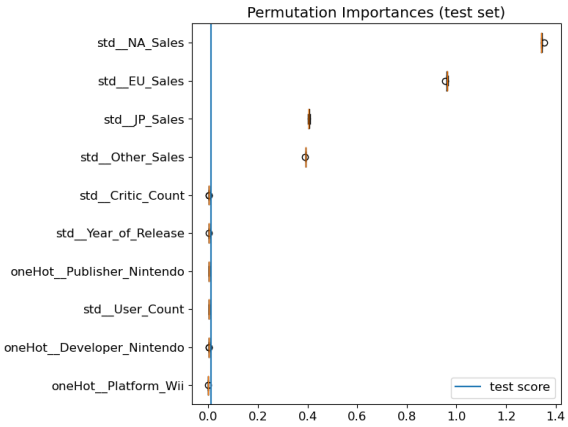

Fig. 5. Plot of Permutation importance for Ridge model 2

The North American sales, followed by European and Japanese sales have the highest correlations with the game selling more. Of course, higher sales in all countries will lead to there being higher global sales altogether. When looking at non-sales related features, we can tell that the next most important feature is critic count. In the context of this situation, if more critics were willing to play the game, there would be more ratings on the game which would possible sway people to or from playing a game. Similarly, user count is another important feature. If more people play the game, then the game would likely have more sales as its obviously popular. In the timeframe the data in my dataset was collected, Nintendo had the highest correlation with global sales. With Nintendo selling a lot of franchises like Pokémon and Super Mario Bros., it is understandable that games that sell more are published by them since they have a lot of recognizable and popular IPs.

In the dataset, the game that had the highest number of sales was Wii Sports, a game which was published by Nintendo on the Nintendo Wii, which is another important feature on the plot.

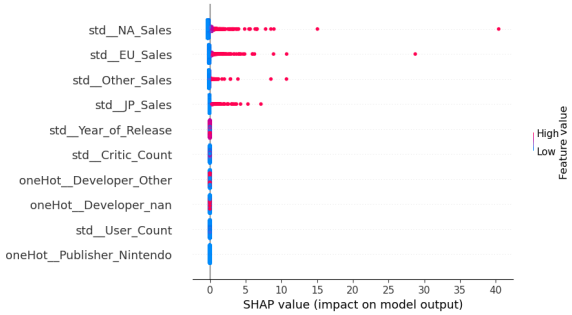

Fig. 6. SHAP summary plot of feature importance

I found it surprising that in both the permutation importances and the SHAP summary plot, other consoles like Playstation or XBox weren't present, since games like Grand Theft Auto V or Call of Duty Black Ops were really popular on those devices. Perhaps Nintendo and the Wii just made games that were more popular.

5. OUTLOOK

To improve the model, I would like to work on understanding why ridge has such an extremely high number of standard deviations above the baseline.

If I had the opportunity to use a faster/stronger computer, I would like to tune more of the XGBoost hyperparameters and to run more random states to allow for more training to be run on this dataset. Even without a different computer, I would like to use the reduced-feature model instead of using imputation, since even though it is statistically sound, imputing data for something like game ratings could artificially boost the results of a game which otherwise wouldn't be highly rated.

In order to improve on model performance, I would like to try to find an accurate method of gaining the ratings for games that don't have them. For some situations, the game wouldn't have ratings since doing them wouldn't have been popular at the

time, or the game itself wasn’t popular enough to get enough ratings to where there isn’t an overwhelming bias.