# Bálint Márk Domián – videogames own project submission report

*(I am not a native english speaker so apologies if any language errors appear in my report because I did not use any kind of AI even for language correction purposes to not violate the Honor Code)*

*I did not include any code in this report to be as reader friendly as possible. My full code can be found in the R script file.*

## Introduction/Overview/Executive Summary

The objective of this project is to develop a machine learning algorithm to predict the commercial success of video games. I chose this dataset because I really like video games, and I thought this would be interesting to find out if I can predict if a videogame would be a success or not. Also, it is a really important task for developers and publishers as well to know what potential video game can become a global success. My goal in this project is to build an algorithm that can determine whether a new game will be a commercial "hit" or "flop" based on its own attributes. I will evaluate the models' performance by finding out its overall accuracy, sensitivity, and specificity. For reference, sensitivity means that the model correctly identifies hits, and specificity means that it correctly identifies flops.

My analysis is based on the Video Game Sales with Ratings Dataset from Kaggle (Video Game Sales with Ratings). It contains over 16000 entries with features including the game's regional sales, user and critic scores, the platform, genre, and the publisher as well.

I started off my algorithm development with an extensive data cleaning procedure and feature engineering. I had to transform the continuous sales data into the binary target variable "Success". The first model was a logistic regression algorithm, which had an overall accuracy of 0.884834, sensitivity of 0.9566257, and a specificity of 0.378133. This is a pretty good result, but I wanted to improve the model's specificity, so I tried combating this problem with a random forest classifier. The random forest classifier achieved an overall accuracy of 0.8875262, sensitivity of 0.9996585, but the specificity has decreased significantly to 0.09638554. My third and final algorithm was a balanced forest classifier so that it can combat the severe data imbalance. It achieved an accuracy of 0.8782531, sensitivity of 0.9231557, and specificity of 0.5614458.

## Methods and analysis

The initial dataset required significant cleaning and transformation to be adequate for this kind of model. First, target variable engineering had to be done. I converted the continuous Global_sales coloumn into a binary factor, "Success". I defined a game as a "hit" if its Global_sales were above 1 million, otherwise I classified it as a "flop". This revealed that the flops significantly outnumber the hits, which leads to severe class imbalance, which will become a problem later.

There were a lot of missing values in the dataset, so next I had to tackle this problem. For the numerical data, like critic_score and user_score, I imputed the median of these coloumns to maintain data distribution. For the categorical data coloumns like rating and developer, I introduced a new distinct category called "Unknown" to preserve these rows. I could not simply remove these because each had more than 6600 datapoints missing, which would make my dataset much smaller.

The next step was feature encoding. For the categorical values like platform, genre, publisher, developer and rating I converted them into numerical dummy variables using a method called one-hot encoding from the caret package (caret::dummyVars). After this, I split the preprocessed data into 2 groups. 80% of the dataset became the training set and the other 20% became the testing set for the final performance evaluation. I used a stratified split so that I can maintain the imbalanced hit/flop ration in both the samples.

My modeling approach was to start off simple so that my computer can handle the calculations and maybe add more complexity if it is possible.

The first model was a baseline logistic regression. This model uses this formula:

$$P(\mathrm{Hit}) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i X_i)}}$$

The overall accuracy and the sensitivity of the model was excellent, but the specificity was poor. This was an indication that the model was biased towards predicting the majority class flop because of the underlying data imbalance, which has led to a high number of false positives.

The second model, I tried to tackle this problem with an uncontrolled random forest model, which is more advanced and complex. While it is powerful, the model made the imbalance issue much worse. It achieved a staggering 99,97% sensitivity which is nearly perfect, but the specificity fell to 9,64%. This means that the model just simply predicted every game as a “hit”. This is a good demonstration of overfitting to the minority class.

The third and final model is a balanced random forest classifier. This final model used a random forest combined with a synthetic minority oversampling technique (SMOTE). SMOTE is a technique applied during training to generate synthetic datapoints for the minority class “hit”, so it can balance the training set without altering the real testing data.

**Results**

This table compares the performance of each model on the testing set:

| Metric | Logistic regression | Uncontrolled random forest | Balanced random forest |
|---|---|---|---|
| **Overall accuracy** | 0.8848 | 0.8875 | 0.8783 |
| **Sensitivity** | 0.9566 | 0.9997 | 0.9232 |
| **Specificity** | 0.3781 | 0.0964 | 0.5614 |

As you can see from the table, the 3. and final model achieved significantly higher specificity than the first 2 models, with keeping the accuracy and specificity almost the same. The first logistic regression model achieved 0.8848 overall accuracy and 0.9566 sensitivity, but with a lower 0.3781 specificity. The second uncontrolled random forest model increased the sensitivity to 0.9997 which is a clear sign of overfitting, but the specificity fell down to 0.0964. The accuracy of the second model stayed almost the same as the first one 0.8875. The final model was a balanced random forest with smote technique to balance the severe data imbalance in this dataset. The accuracy of the model was 0.8783, the sensitivity was 0.9232, but the important part is that the specificity achieved 0.5614, which is a significant increase from the previous 2 models.

**Conclusion**

Doing this project was very challenging but rewarding. I successfully developed and optimized a random forest classification algorithm, that can predict the commercial success of video games with high and balanced performance metrics. The most challenging part of this project was to handle the severe data imbalance issue, but I can honestly say that I succeeded.

**Summary**

My final approach combined the predictive power of the random forest model and the structural correction SMOTE provided. This combination ensured that the model learned the defining characteristics of successful games and games that would be classified as failures, rather than just defaulting to the majority class. My final model achieved a balanced performance score of the metrics on the unseen testing set.

**Limitations and future work**

While this model works well, it still has some limitations of course. The model only predicts if a certain game will be a hit, not how many of it will sell. So, it will give a hit to a 1000001 sales game and also a hit to a 10000000 sales game as well, and as you can imagine there is a big difference between them. Because of time and computational constraints I have, the random forest model was trained with simple control methods. For future work could involve rigorous hyperparameter tuning to achieve a significant increase in accuracy and stability. My model does not account for the time effect (what season or month it is), which is a very important thing in the gaming market. And finally, the high number of publisher dummy variables may still be too noisy. For future work to simplify the model, smaller publishers could be grouped into a single category called “independent” or “indie”.