Stanford | ENGINEERING
Computer Science

CS 129: Applied Machine Learning

Predicting the Success of a New Video Game Using Machine Learning

Gaming/Tech Category

Anthony Boukarim – 06276240

Jennifer Cheng – 06034126

Marco De Leon-Fadok – 05864927

12/12/2019

Introduction

With recent technological advancements, the video games market is experiencing a massive growth in audience and revenue, achieving $1.1 billion dollars by the end of 2019 (up 27% from last year) [1]. The Steam Store is one of the most developed platforms for gamers, with more than 90 million monthly active users and 30,000 video games [2]. It would be beneficial to developers if the success of the games could be predicted beforehand, as it would allow them to adjust their designs and make necessary changes to attract a larger audience. A recent dataset containing information on more than 27,000 Steam Store games is available on Kaggle. Some of its categories include: game description, number of players, release date, game price, number of positive and negative ratings, GPU requirements, and Steam tags.

In this project, we will focus on using machine learning algorithms (specifically logistic regression, neural networks, and decision trees) to predict the average yearly revenue of a newly released video game. The inputs to our algorithms are the following information for each video game: release date, number of players, price of the game, number of positive and negative ratings, age requirement, number of in-game achievements, game genre, and number of occurrences of Steam tags (for 368 different tags). The output of our algorithm is the predicted range of average yearly revenue.

Similar projects have been completed in the past. For instance, one researcher predicted a video game's success using the metric of average number of concurrent players in the first two months after release [3]. Among the models used, the most successful were support vector machine and a random forest algorithm (multiple decision trees) [3]. Researchers from Oklahoma State University used a neural network to predict revenue of movies as one of nine categories. The metric used is the average percent success rate of successfully classifying the movie's success within one class of its actual performance [4]. This is very similar to our problem, since we are also predicting financial success as a classification problem, although the features used are quite different. In another project, researchers in Japan used convolutional neural networks to predict the in-app purchases of players based on their in-game behavior and then estimate their value over the next year [5].

Dataset.

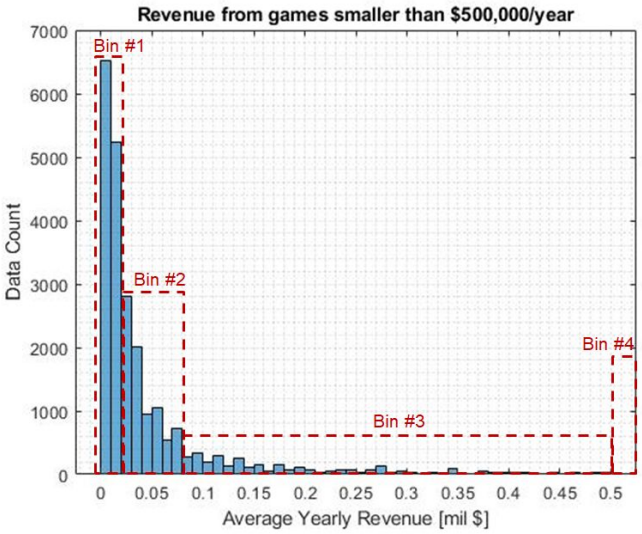
appid	name	release_da	english	developer	publisher	platforms	required_a	categories	genres	steamspy_t	achieveme	positive_ra	negative_r	average_pl	median_pl	owners	price
10	Counter-Str	11/1/2000	1	Valve	Valve	windows;m	0	Multi-playe	Action	Action;FPS;	0	124534	3339	17612	317	10000000-2	7.19
20	Team Fortre	4/1/1999	1	Valve	Valve	windows;m	0	Multi-playe	Action	Action;FPS;	0	3318	633	277	62	5000000-10	3.99
30	Day of Defe	5/1/2003	1	Valve	Valve	windows;m	0	Multi-playe	Action	FPS;World \	0	3416	398	187	34	5000000-10	3.99
40	Deathmatcl	6/1/2001	1	Valve	Valve	windows;m	0	Multi-playe	Action	Action;FPS;	0	1273	267	258	184	5000000-10	3.99
50	Half-Life: O	11/1/1999	1	Gearbox So	Valve	windows;m	0	Single-play	Action	FPS;Action;	0	5250	288	624	415	5000000-10	3.99
60	Ricochet	11/1/2000	1	Valve	Valve	windows;m	0	Multi-playe	Action	Action;FPS;	0	2758	684	175	10	5000000-10	3.99
70	Half-Life	11/8/1998	1	Valve	Valve	windows;m	0	Single-play	Action	FPS;Classic;	0	27755	1100	1300	83	5000000-10	7.19
80	Counter-Str	3/1/2004	1	Valve	Valve	windows;m	0	Single-play	Action	Action;FPS;	0	12120	1439	427	43	10000000-2	7.19
130	Half-Life: Bl	6/1/2001	1	Gearbox So	Valve	windows;m	0	Single-play	Action	FPS;Action;	0	3822	420	361	205	5000000-10	3.99
220	Half-Life 2	11/16/2004	1	Valve	Valve	windows;m	0	Single-play	Action	FPS;Action;	33	67902	2419	691	402	10000000-2	7.19
240	Counter-Str	11/1/2004	1	Valve	Valve	windows;m	0	Multi-playe	Action	Action;FPS;	147	76640	3497	6842	400	10000000-2	7.19

Figure 1. Sample of raw data points

The data we used is sourced from Kaggle and contains information on over 27,000 video games on the Steam store [6]. We used Python to clean our data and started by removing irrelevant data points (such as free games) since we are assuming that the average yearly revenue is exclusively from the initial purchase of the game (i.e we are ignoring in-game purchases). Since the raw data is not labeled with their corresponding average yearly revenue, we computed that ourselves using the release date, price of the game, and number of players. The following equation shows the exact calculation:

$$y^{(i)}_{(avg\ yearly\ revenue)} = \frac{price\ of\ game^{(i)} \times number\ of\ players^{(i)}}{number\ of\ years\ since\ release^{(i)}}$$

After multiple investigation, we decided to work with a multi-class classification problem problem since we felt that our features were not complex enough to predict accurate decimal values of the revenue. To select our classes (revenue bin sizes), we plotted a histogram of the average yearly revenue. Figure 2 shows that our data is highly skewed towards smaller revenues, and thus we selected four bins with different sizes.



Revenue Bins

Bin #1	*0 – $20,000*
Bin #2	*20,000 – $80,000*
Bin #3	*80,000 – $500,000*
Bin #4	*> $500,000*

Figure 2. Histogram of average yearly revenue and bin selection

Next, we needed to choose the features for our training algorithm. After researching the dynamics of the gaming industry and training the model for different features, we selected the following: price of game, ratio of positive to negative ratings, total number of ratings, minimum age required, number of in-game achievements, single player vs multiplayer, and a dictionary of 368 Steam tags that describe the genre of the game along with their number of occurrences. We performed feature scaling and normalization using the following formula: $x_j = \frac{x_j - featureAverage}{featureStandardDeviation}$.

The final step before training our model consisted of splitting our dataset using stratified sampling into training (60%), cross validation (20%), and test (20%) sets. There are 14,709 examples in the training set and 4,903 examples in the cross validation and test sets. A sample of the first five entries in our dataset is shown below (not all columns are shown, as there are over 300 features).

	total_ratings	singleplayer_x	multiplayer_x	required_age	ratings	achievements	avg_revenue	price	1980s	1990s	2.5d	2d	2d_fighter	3d
0	12.179710	0	1	-0.149343	0.973888	-0.130342	5.392500e+06	0.059529	5.508329	31.412767	-0.09207	-0.103417	-0.049079	-0.057376
1	0.287775	0	1	-0.149343	0.839787	-0.130342	1.425000e+06	-0.339755	-0.043865	3.885818	-0.09207	-0.103417	-0.049079	-0.057376
2	0.274628	0	1	-0.149343	0.895648	-0.130342	1.760294e+06	-0.339755	-0.043865	-0.078509	-0.09207	-0.103417	-0.049079	-0.057376
3	0.056408	0	1	-0.149343	0.826623	-0.130342	1.575000e+06	-0.339755	-0.043865	-0.078509	-0.09207	-0.103417	-0.049079	-0.057376
4	0.440068	1	1	-0.149343	0.947996	-0.130342	1.425000e+06	-0.339755	-0.043865	4.220831	-0.09207	-0.103417	-0.049079	-0.057376

Figure 3. First five entries in dataset

Methods

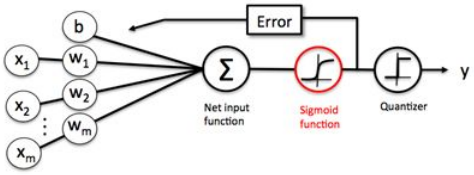


Figure 4. Schematic of logistic regression algorithm

In logistic regression, a sigmoid function is used to calculate the probability of a training example belonging to a class. Using one-vs-all, logistic regression can be turned into a multi-class classification algorithm. The cost function for logistic regression is shown below. The goal is to minimize the cost function with respect to the parameters Θ.

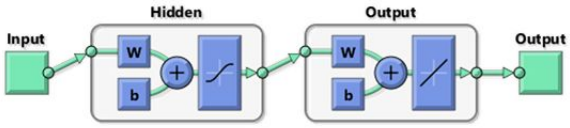


Figure 5. Schematic of neural network algorithm

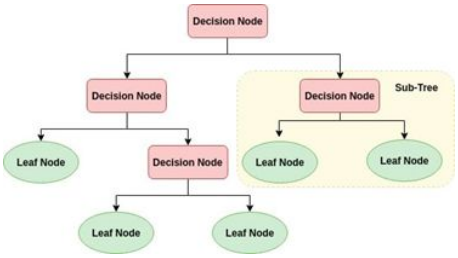


Figure 6. Schematic of decision tree algorithm

In a neural network, an input is passed through nodes and layers and an output is generated. Each layer can "learn" something different. The nodes are used to calculate a hypothesis, such as the sigmoid function and the edges are weights. The goal is to minimize the cost function with respect to Θ.

Decision trees are a form of supervised learning. It can be used for both classification and regression problems [6]. A decision tree consists of decision nodes and leaf nodes. The tree learns what decision rules to use at the decision nodes based off the data. The end goal is to predict a result or value [6]. As the depth of the tree (the layers) increases, the tree increases in complexity and can overfit.

Experiments, Results, and Discussion

Since our problem is highly skewed towards lower revenue bins, accuracy is not the best metric to consider. To test generalization, we decided to base our analysis on the F1 and AUC scores as primary metrics (shown at the end of this section). We also computed confusion matrices to see how far are our predictions from the actual category.

Logistic Regression

To train our model we used the scikit-learn library in Python. For logistic regression, we used the "saga" solver for multinomial logistic regression and 1000 iterations. The Python library automatically uses regularization.

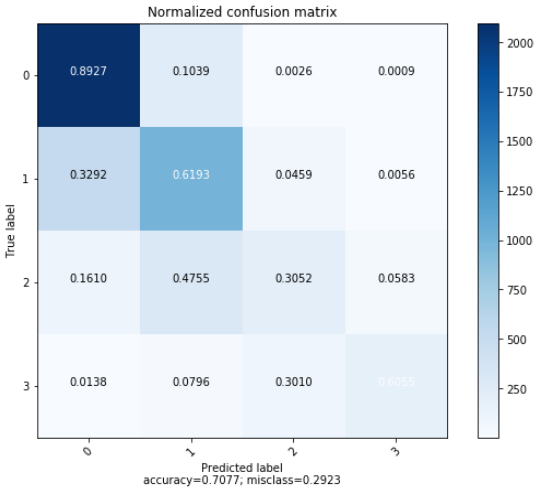


Figure 7. Logistic regression confusion matrix (test set)

The color coding in the confusion matrix shows the number of data points falling within each category. It is clearly seen that our data is skewed towards the lower revenue bins.

Neural Network

Searching for a more complex predicting model, we trained an artificial neural network (ANN) on our data set. To understand the sensitivity of network architecture, a learning curve were plotted.

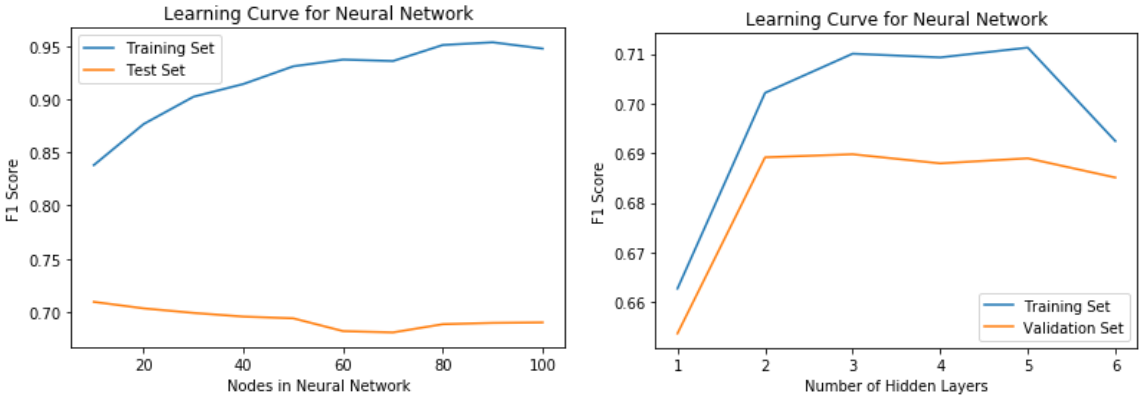


Figure 8 and 9. F1 Score vs. Nodes in ANN (left: layers fixed at 5, right: nodes fixed at 50)

The final architecture of the neural network was 5 hidden layers each containing 50 nodes. The maximum number of iterations was set to 800. Our model was still overfitting the training set (F1 score of 0.96 on training set and 0.61 on test set). We decided to hypertune the regularization parameter by looping over different values of λ and selecting the one with the highest F1 score on the cross validation set ($\lambda = 1.2$). The following confusion matrix summarizes the performance of the ANN predictor on the test set.

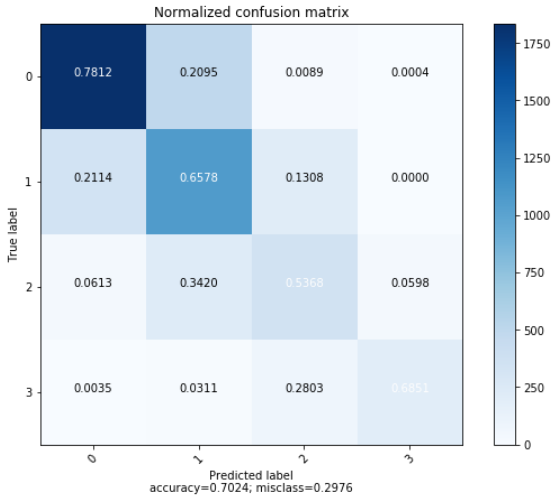


Figure 10. Neural Network Confusion Matrix (test set)

Decision Tree

The decision tree we trained reached a maximum depth of 57 layers. Plotting the F1-score versus the depth of the tree, we notice that the deviation between the training and validation sets occurs at about the 5th layer (Figure 11). To address the overfitting, we set the maximum depth of the decision tree to 5 layers.

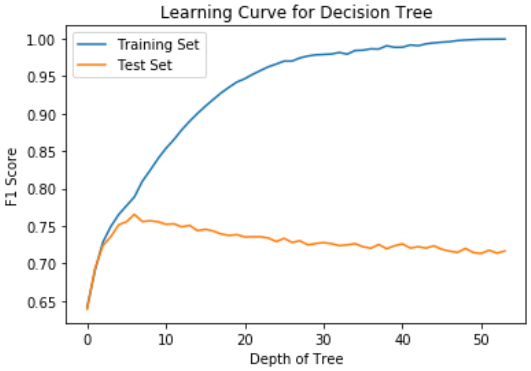


Figure 11. F1 Score vs. Depth of Tree

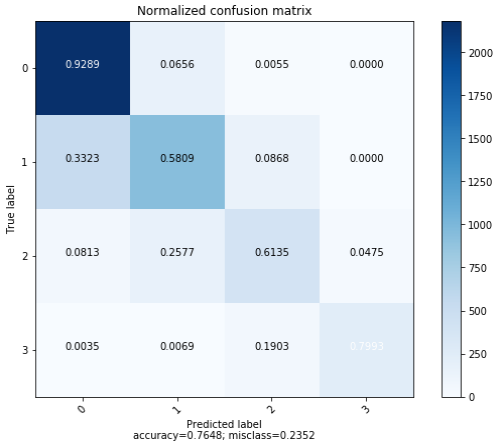


Figure 12. Decision Tree Confusion Matrix (test set)

The performance of the regularized decision tree on the test set is shown in the decision matrix below. This shows that the predictor was successful at predicting the range of yearly revenues.

To summarize our results we tabulated the performances of each algorithm in the following tables. Table 2 shows the final results after adding features and regularizing each of our algorithms. The three algorithms performed relatively well in predicting the revenues. The neural network and decision tree predictors provided better AUC scores than logistic regression. Decision tree performed slightly better based on the F1 score.

	Training Set		Validation Set	
	F1 Score	**AUC Score**	**F1 Score**	**AUC Score**
Log Regression	0.731	0.774	0.708	0.743
ANN *(5 layers, 50 nodes each)*	0.823	0.910	0.702	0.776
Decision Tree *(5 layers)*	0.765	0.816	0.765	0.817

Table 1. Final F1 and AUC Scores

Conclusion/Future Work

Our learning model exceeded our expectations. We initially believed that the raw data did not have enough correlation between its features and thus expected that our model will be unable to accurately predict revenue ranges. The three algorithms performed relatively well in predicting the revenues. In the confusion matrices, we see that the algorithm is predicting well and misclassification occurs primarily around the neighboring bins. The neural network and decision tree predictors provided better AUC scores than logistic regression. Decision tree performed slightly better based on the F1 score. Looking at a higher level, the selected features provided enough information for the ANN and decision tree algorithms to learn from the data (and even overfit it).

For future work, we recommend adding more complex features that better describe the dynamics of the video gaming industry. Examples of these features are: capital invested for marketing the game, computer specifications for a good gaming experience, update patterns, availability of sponsorships and funding for professional tournaments. Additionally, data on consumer (gamer) behavior can be highly beneficial to find patterns.

Contributions

As a group, we brainstormed all aspects of the project together, worked on the distribution of the bins, feature selection, the parameters of each learning algorithm, and contributed to the writing of the deliverables.

For individual tasks:

- Anthony: data visualization, histogram analysis for bin selection, stratified sampling, error analysis on neural network, adding features (Steam tag dictionary, age, etc.), regularization of neural network
- Jennifer: preliminary data cleaning and feature creation; code for: linear regression, logistic regression, neural networks, decision trees; error analysis (F1, AUC, confusion matrix, learning curves) for all models
- Marco: error analysis, adding features, architecture experimentation, hyperparameter tuning

References

[1] “Global esports revenues to top $1 billion in 2019: report - Reuters.” [Online]. Available: https://www.reuters.com/article/us-videogames-outlook/global-esports-revenues-to-top-1-billion-in-2019-report-idUSKCN1Q11XY. [Accessed: 16-Nov-2019].

[2] “Steam now has 30,000 games | PC Gamer.” [Online]. Available: https://www.pcgamer.com/steam-now-has-30000-games/. [Accessed: 16-Nov-2019].

[3] M. Trněný, “Machine Learning for Predicting Success of Video Games.” https://is.muni.cz/th/k2c5b/diploma_thesis_trneny.pdf. [Accessed: 11-Dec-2019].

[4] “Predicting box-office success of motion pictures with neural networks - ScienceDirect.” [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417405001399. [Accessed: 11-Dec-2019].

[5] “Customer Lifetime Value in Video Games Using Deep Learning and Parametric Models - IEEE Conference Publication.” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8622151. [Accessed: 11-Dec-2019].

[7] “Softmax regression” [Online]. Available: http://rasbt.github.io/mlxtend/user_guide/classifier/SoftmaxRegression/. [Accessed 08-Dec-2019].

[6] “Steam Store Games (Clean dataset).” [Online]. Available: https://kaggle.com/nikdavis/steam-store-games. [Accessed: 06-Dec-2019].

[8] “How does the decision tree algorithm work?” [Online]. Available: https://ineuron.ai/how-does-the-decision-tree-algorithm-work/. [Accessed: 16-Nov-2019].

[9] “1.10. Decision Trees — scikit-learn 0.22 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/tree.html. [Accessed: 06-Dec-2019].