# Predicting Video Game Sales with Machine Learning

Jacob Miller
ITCS 5156
05/02/23

Data Interpretation and Video Games Sales Prediction Using Machine Learning Algorithms

*Dr.A.Manimuthu, Dr.U.Udhayakumar, A.Cathrine Loura, Dr.Peter Jose P, T D Gowri, Dr.L.Selvam, Dr.S.Roseline Mary*

Published: March 8, 2023

# 1. Introduction

### 1.1 Problem Statement

With video games increasing in popularity each year this has led to a large amount of data collected on people, such as their likes and dislikes of certain games and the style in which they play. Machine learning can be useful in figuring out sales predictions and upcoming business ventures in the video game industry. Having a way for developers, businesses, investors, and consumers to have accurate game sale trends can provide great marketing strategies, which can also be used to figure out if a new game will be considered a “hit” or not.

### 1.2 Motivation

My bachelor degree is in game development so I thought it would be interesting to find a topic that relates to video games. I decided on forecasting video games because I thought it would be interesting if I could implement an algorithm that would decide if a new game would be considered a “hit” or not.

### 1.3 Overview

Using machine learning algorithms such as Logistic Regression and Random Forest Classifiers is used in the prediction process for forecasting game sales. If a game has sold over a million copies then it is considered a “hit”. Finding out which game genre is the most popular can also help with determining if a game has a chance of becoming a hit.

# 2. Background

### 2.1 Summary of related papers

Both papers follow very similar steps in their data preprocessing, including cleaning the data, dropping null values, and finding a target. Both papers involve trying to find sales trends in video games and the marketing schemes that can be attached to it. Paper 1 tries to predict the top selling video games in North America between 1983 and 2016 by using linear regression. Paper 2 tries to find what algorithm can be used to find the highest accuracy for video game sales predictions. Both papers tested their models with an 80% train and 20% test ratio. Paper 2 concludes that Random Forest produces the highest accuracy among the tested algorithms with an accuracy of 96%.

### 2.2 Pros and Cons

Some of the pros for both papers is that Random Forest produces the highest testing percentages out of all the tests they ran. Both papers tested video game sales in different continents and found that North America produced the most games. Some of the cons is that Paper 1 only tests Linear Regression, which had the lowest results out of all the tests in Paper 2. Paper 2 has linear regression only having an accuracy score of only 0.5734.

### 2.3 Relation with topic

The survey papers relate to the main paper since they are all based on predicting video game sales and marketing of games. The main paper and Paper 2 have a lot in common and use very similar algorithms and machine learning techniques such as their uses of random forest.

## 3. Methods

### 3.1 Details of Methods and Algorithms

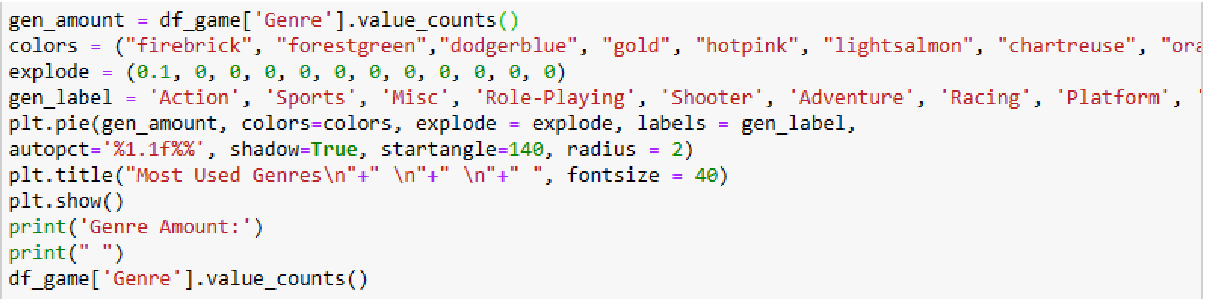
The method chosen in the paper was Multiple Regression and Random Forest, since they both provided the highest accuracy among all the tested algorithms. I also tested with Decision Trees since I thought it could provide a solid base for accuracy and comparison between Random Forest and Linear Regression since they had relatively similar accuracies.

### 3.2 Data Preprocessing

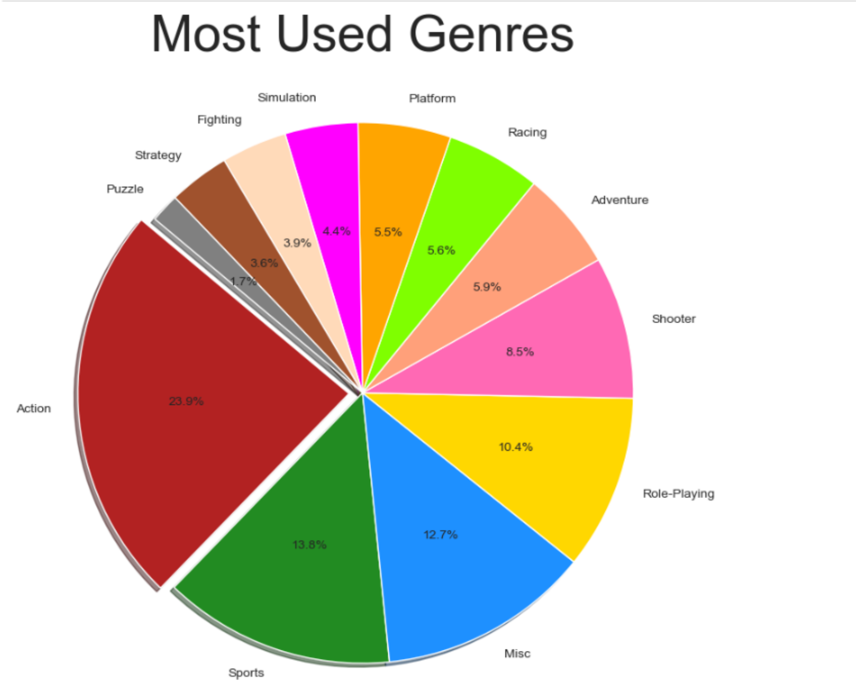
For data preprocessing I first tried to fill all missing values with the median of the surrounding values, but that skewed my data, so I instead opted to just drop all missing values. I did this by doing: df_names = df_names.dropna().reset_index(drop=True), which dropped all the null values in the columns. In the paper they also replaced null values with an appropriate chosen method. They also utilized feature selection where their goal was to avoid “Curse of dimensionality” where the less the number of features, the more accurate the model performs.

### 3.3 Overall Framework

The first step in the framework process that the paper did was figuring out what game genres are the most frequently developed. The results from the paper had action as the highest and sports as the second highest. So I tested my data as well and found very similar results.



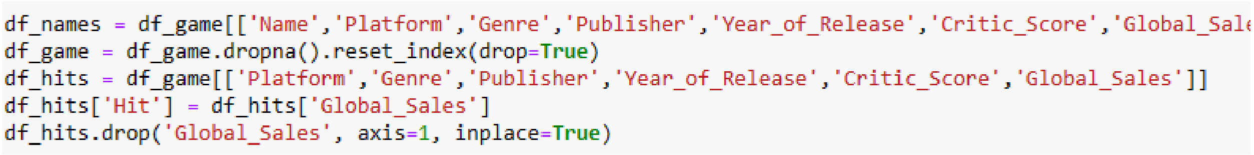
```
gen_amount = df_game['Genre'].value_counts()
colors = ("firebrick", "forestgreen","dodgerblue", "gold", "hotpink", "lightsalmon", "chartreuse", "ora
explode = (0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
gen_label = 'Action', 'Sports', 'Misc', 'Role-Playing', 'Shooter', 'Adventure', 'Racing', 'Platform', '
plt.pie(gen_amount, colors=colors, explode = explode, labels = gen_label,
autopct='%1.1f%%', shadow=True, startangle=140, radius = 2)
plt.title("Most Used Genres\n"+" \n"+" \n"+" ", fontsize = 40)
plt.show()
print('Genre Amount:')
print(" ")
df_game['Genre'].value_counts()
```



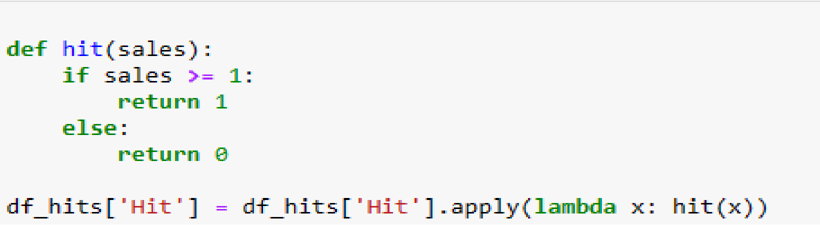
## 4. Experiments

### 4.1 Training Data

The first step in my experiment was setting up what would be considered a hit or not and then training the data with a 70% train size. If a game has sold over a million copies, then it will be considered a hit. This was done with the help of a Kaggle reference I utilized and reference throughout the paper.

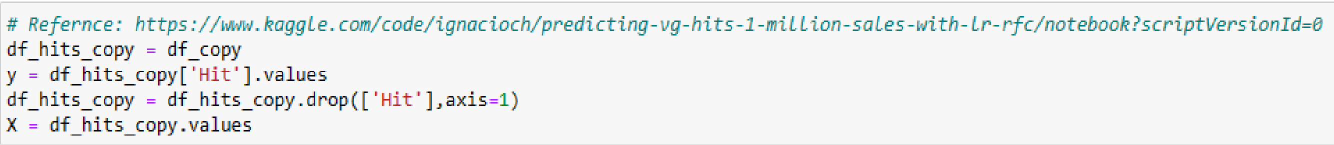


```
df_names = df_game[['Name','Platform','Genre','Publisher','Year_of_Release','Critic_Score','Global_Sale
df_game = df_game.dropna().reset_index(drop=True)
df_hits = df_game[['Platform','Genre','Publisher','Year_of_Release','Critic_Score','Global_Sales']]
df_hits['Hit'] = df_hits['Global_Sales']
df_hits.drop('Global_Sales', axis=1, inplace=True)
```

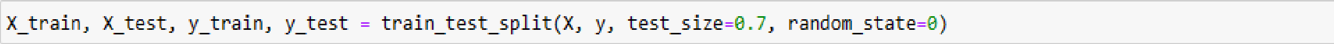


```
def hit(sales):
    if sales >= 1:
        return 1
    else:
        return 0

df_hits['Hit'] = df_hits['Hit'].apply(lambda x: hit(x))
```



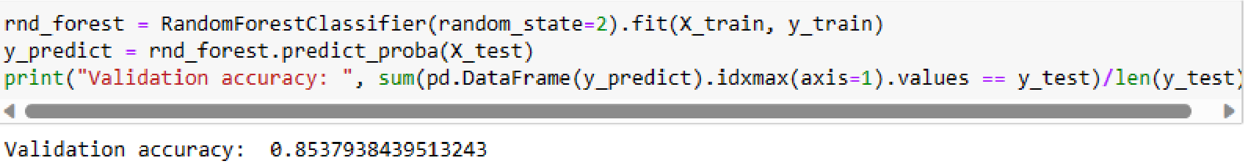
```
# Refernce: https://www.kaggle.com/code/ignacioch/predicting-vg-hits-1-million-sales-with-lr-rfc/notebook?scriptVersionId=0
df_hits_copy = df_copy
y = df_hits_copy['Hit'].values
df_hits_copy = df_hits_copy.drop(['Hit'],axis=1)
X = df_hits_copy.values
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=0)
```

## 4.2 Testing different methods

Once I trained all the data the first algorithm I tested was Random Forest Classification.

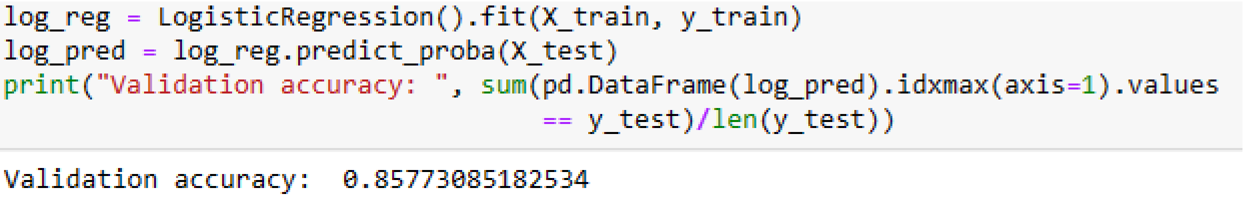


```
rnd_forest = RandomForestClassifier(random_state=2).fit(X_train, y_train)
y_predict = rnd_forest.predict_proba(X_test)
print("Validation accuracy: ", sum(pd.DataFrame(y_predict).idxmax(axis=1).values == y_test)/len(y_test)

Validation accuracy:  0.8537938439513243
```

I tested Random Forest with different random states and found its validation accuracy to be around 0.85%. Which is a pretty good percentage, but obviously much lower than the accuracy from the research paper which had it at 98%.

The next algorithm I tested was Logistic Regression which produced the highest validation accuracy.

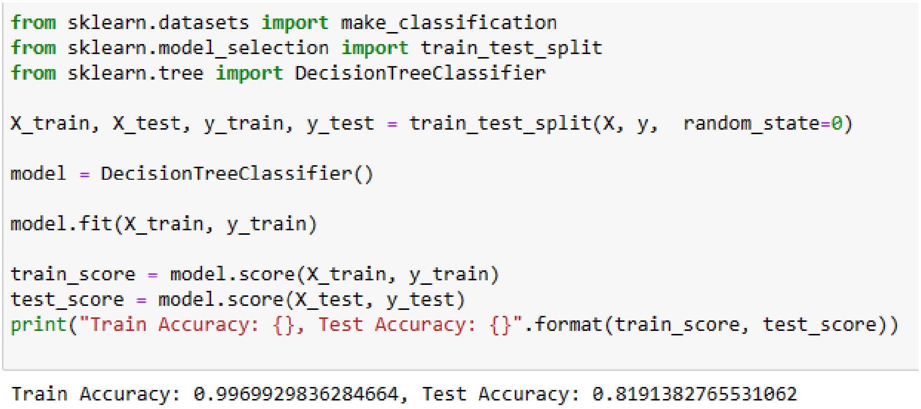


```
log_reg = LogisticRegression().fit(X_train, y_train)
log_pred = log_reg.predict_proba(X_test)
print("Validation accuracy: ", sum(pd.DataFrame(log_pred).idxmax(axis=1).values
                                   == y_test)/len(y_test))

Validation accuracy:  0.85773085182534
```

After testing I found Logistic Regression to have an average validation accuracy of 86%. This means that so far Logistic Regression has shown the highest accuracy among the tested algorithms.

I then wanted to test the data with Decision Trees since they are easy to work with and can be used as a middle ground basis for my tests results.



```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y,  random_state=0)

model = DecisionTreeClassifier()

model.fit(X_train, y_train)

train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score, test_score))
```

```
Train Accuracy: 0.9969929836284664, Test Accuracy: 0.8191382765531062
```

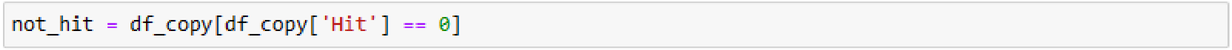
As expected, Decision Tree performed the worst out of the three tested algorithms with a validation accuracy of around 82%. After this testing I was now ready to start predicting potential hit games.

### 4.3 Displaying Results

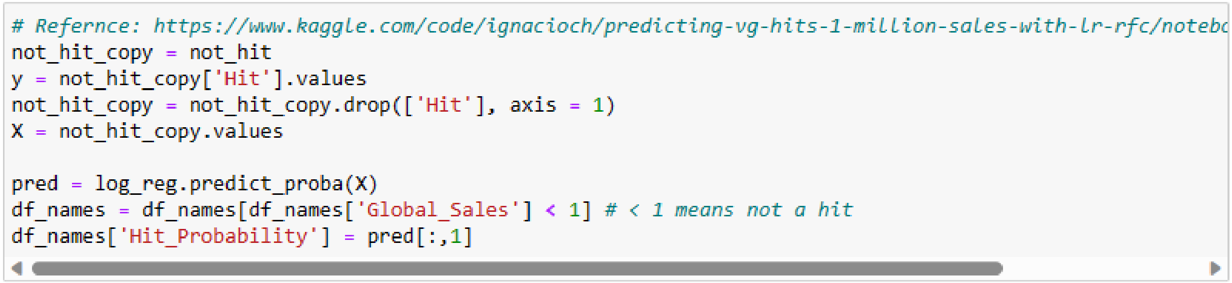
I was not able to produce the same results as showcased in the paper and one of the reasons is that I used a similar, but different data set compared to the one used in the paper. The research paper also did a much better job with data preprocessing and I was not able to replicate that well. Even though validation accuracy in my experiment was much lower than the one referenced, it still had the same overall results. For example in both my experiment and the paper, North America was shown to have the largest amount of game sales. The results also showed that action is the most popular gamer genre with sports coming in a close second. However the paper didn't talk about games becoming a success and rather focused on the growth of marketing around video games as a whole. So, I decided to take it a step further, and with the help of Kaggle, I was able to test what video games still had a potential of becoming a hit.

### 4.4 Experimenting with potential hits

Since the paper had no reference to a game being a "hit" I used a reference from Kaggle to help implement this formula. The first step to finding out if a game has the chance of becoming a hit was to calculate all the potential hits.

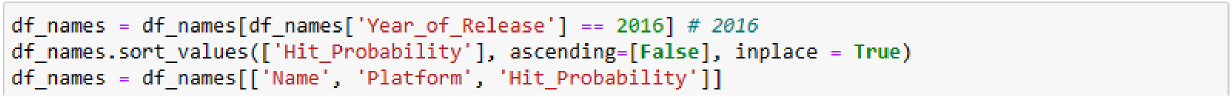


```
not_hit = df_copy[df_copy['Hit'] == 0]
```



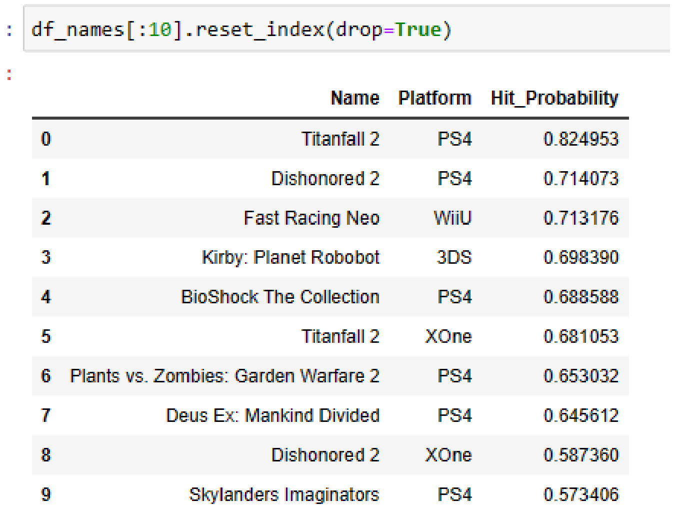
```
# Refernce: https://www.kaggle.com/code/ignacioch/predicting-vg-hits-1-million-sales-with-lr-rfc/notebo
not_hit_copy = not_hit
y = not_hit_copy['Hit'].values
not_hit_copy = not_hit_copy.drop(['Hit'], axis = 1)
X = not_hit_copy.values

pred = log_reg.predict_proba(X)
df_names = df_names[df_names['Global_Sales'] < 1] # < 1 means not a hit
df_names['Hit_Probability'] = pred[:,1]
```



```
df_names = df_names[df_names['Year_of_Release'] == 2016] # 2016
df_names.sort_values(['Hit_Probability'], ascending=[False], inplace = True)
df_names = df_names[['Name', 'Platform', 'Hit_Probability']]
```

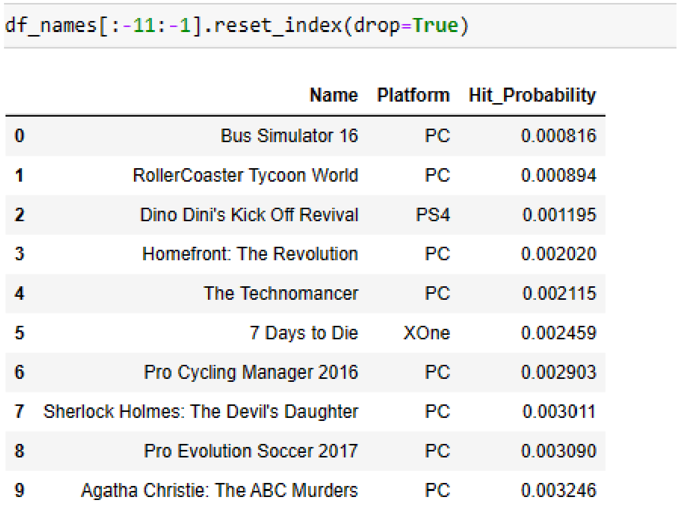
Since Logistic Regression provided the highest accuracy I used it to predict the probability of a game becoming a hit. Due to the fact that a lot of the newer games had missing or null values in the dataset I had to use 2016 as the target. This however can still be very informative since we can now look at these past games to find out if they really did become a success. Below is the top 10 highest and lowest games with the potential of becoming hits:



```
: df_names[:10].reset_index(drop=True)
```

:

| | Name | Platform | Hit_Probability |
|---|---|---|---|
| 0 | Titanfall 2 | PS4 | 0.824953 |
| 1 | Dishonored 2 | PS4 | 0.714073 |
| 2 | Fast Racing Neo | WiiU | 0.713176 |
| 3 | Kirby: Planet Robobot | 3DS | 0.698390 |
| 4 | BioShock The Collection | PS4 | 0.688588 |
| 5 | Titanfall 2 | XOne | 0.681053 |
| 6 | Plants vs. Zombies: Garden Warfare 2 | PS4 | 0.653032 |
| 7 | Deus Ex: Mankind Divided | PS4 | 0.645612 |
| 8 | Dishonored 2 | XOne | 0.587360 |
| 9 | Skylanders Imaginators | PS4 | 0.573406 |



```
df_names[:-11:-1].reset_index(drop=True)
```

| | Name | Platform | Hit_Probability |
|---|---|---|---|
| 0 | Bus Simulator 16 | PC | 0.000816 |
| 1 | RollerCoaster Tycoon World | PC | 0.000894 |
| 2 | Dino Dini's Kick Off Revival | PS4 | 0.001195 |
| 3 | Homefront: The Revolution | PC | 0.002020 |
| 4 | The Technomancer | PC | 0.002115 |
| 5 | 7 Days to Die | XOne | 0.002459 |
| 6 | Pro Cycling Manager 2016 | PC | 0.002903 |
| 7 | Sherlock Holmes: The Devil's Daughter | PC | 0.003011 |
| 8 | Pro Evolution Soccer 2017 | PC | 0.003090 |
| 9 | Agatha Christie: The ABC Murders | PC | 0.003246 |

### 4.5 Thoughts on Model and Results

After looking through the results most of the probabilities were relatively accurate, for example TitanFall 2 had a 84% chance of becoming a hit and it indeed became a hit and sold well over a million copies. However there obviously were some outliers. For example, 7 Days to Die only had a .002% chance of becoming a hit, but obviously that's wrong since it sold well over a million copies. However this did take multiple years to do so. So if you instead considered a game to only be a hit if it sold 1 million copies in its first year of release, then the probabilities would be looked at as more accurate and 7 Days to Die would indeed be considered not a hit. However, for this project I didn't take "time" into consideration as a factor for hits since I would have to research each game to see how long it took to sell a million copies.

# 5. Conclusion

## 5.1 Thoughts on project

This project has taught me about marketing in video games and helped me gain a better understanding of the uses of Machine Learning in the field of video games. It also showed me the importance of using an accurate model and how developing one can be time consuming and difficult.

## 5.2 Future work

If I ever did this again in the future I would like to have an up-to-date dataset without any missing values to test with. If I had a dataset that was always updating then making charts and tables in software like Tableau that showcases real-time game sale trends could be very beneficial and informative. I also think taking the cost of the game into consideration for future testing could change some of the results. Games that cost a lot of money can have low sales if they are not marketed correctly. Overall I think there is a lot more work that can be done with video game sales predictions and I expect to see even more people tackling this topic in the future.

# Citations

Reference Paper:

- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4382007

Survey Papers:

- https://ijcrt.org/papers/IJCRT2005182.pdf
- https://www.jetir.org/papers/JETIR1907H50.pdf

Kaggle/Dataset:

- https://www.kaggle.com/code/hamizanfirdaus/machine-learning-of-video-games-sales
- https://www.kaggle.com/code/ignacioch/predicting-vg-hits-1-million-sales-with-lr-rfc/notebook?scriptVersionId=0