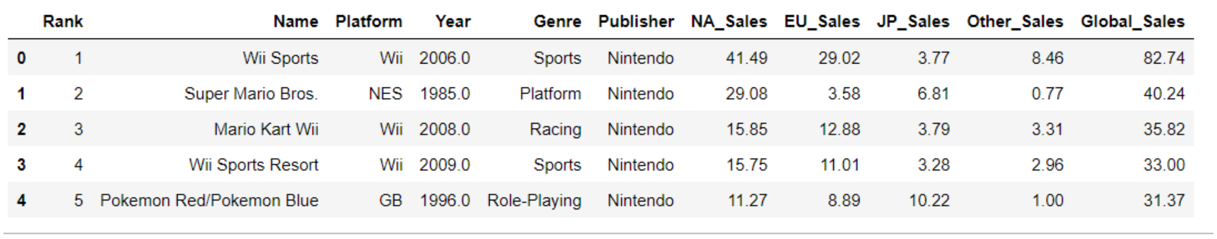
Technical Assessment Report

– PYGASAL –
VIDEO -GAME SALES PREDICTION USING PYTHON

Contents

1. Introduction

The gaming industry is definitely one of the booming industries in the modern era and one of the industries most affected by technological progress. With technologies such as augmented reality (AR) / virtual reality (VR) available in consumer products such as game consoles and even smartphones, the gaming market is showing great potential.
In Data mining, we as data scientists should use our analytical skills to predict video game sales based on certain factors (input features). There are 10 distinct factors that can affect video game sales.



	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37

Figure 1 - Database extract

During our research, we have found three papers that can be considered similar to the problem treated in this report, even though they refer to the film industry. See section 8. for further details.

2. Objective

Our goal as data scientists is to build a machine learning model that can accurately predict sales in millions of units for a given game. If we have the time, we may divide the predictions in world regions.

Additional databases can be extracted from Metacritic or jeuxvideo.com. BeautifulSoup library in Python will be used for web scraping purposes to extract the data out of Metacritic or jeuxvideo HTML and XML files. It creates a parse tree from page source code that can be used to extract data in a hierarchical and more readable manner.

3. Data

3.1 Data Structure

The database for this project will be limited to the database https://www.kaggle.com/gregorut/videogamesales

- containing a list of video games with sales greater than 100 000 copies. The database was created 26-Oct-2016 and it contains 16598 entries (see Figure 1) considering the following list of variables :
- Rank - Ranking of overall sales
- Name - The games name
- Platform - Platform of the games release (i.e. PC,PS4, etc.)
- Year - Year of the game's release
- Genre - Genre of the game
- Publisher - Publisher of the game
- NA_Sales - Sales in North America (in millions of copies)
- EU_Sales - Sales in Europe (in millions of copies)
- JP_Sales - Sales in Japan (in millions of copies)
- Other_Sales - Sales in the rest of the world (in millions of copies)
- Global_Sales - Total worldwide sales.(in millions of copies).

An analisis of the dataset (see annex 10.1) showing that the majority the games are released between years 2000 to 2016 (see Figure 2). The dataset is incomplete for the year after 2017.

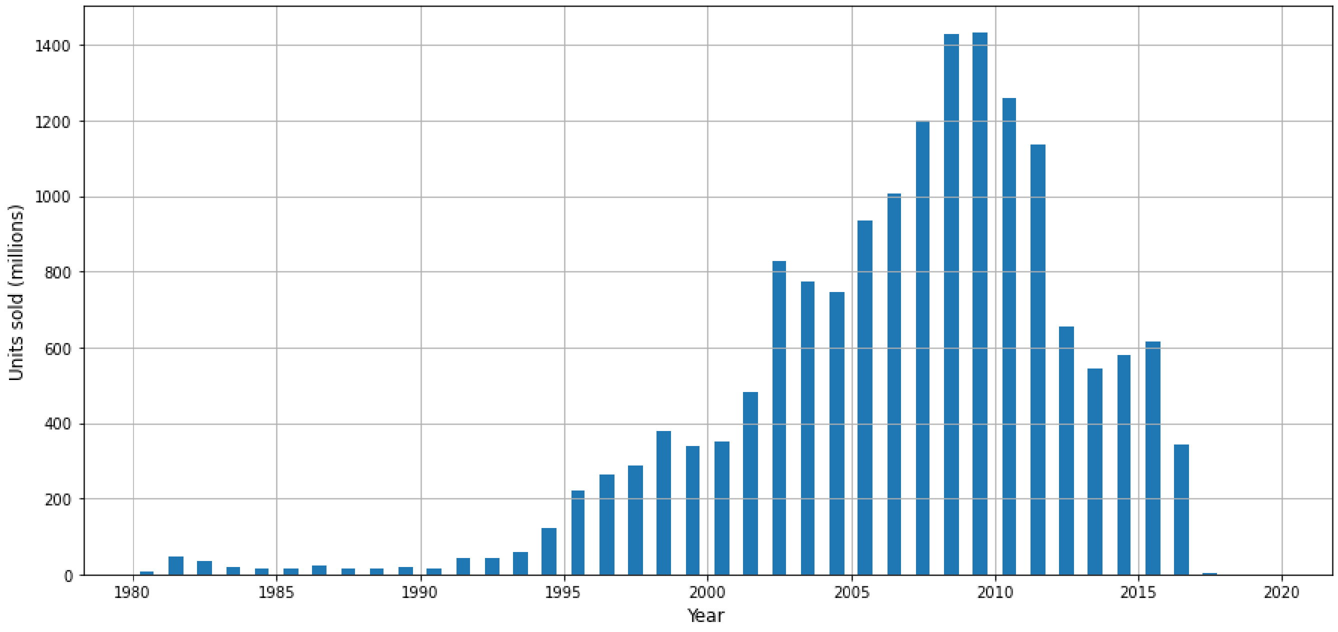


Figure 2 - Number of titles released per year

3.2 Data Completion

Unfortunately, the data we currently have only contains the total sales of the game, the studio, the country and the publisher which not enough to build a prediction model, so we need to gather extra factors/features that reflect the importance of each game from the view point of cutomers and users. Such information can be find on sites such as Metacritic or jeuxvideo.com.
Metacritic.com is a site that aggregates professional reviewer scores from video games (various platforms). Metacritic calculates an average score called Metascore, based on the various professional reviewers by converting the reviewers' local score into a score of 0 to 100 (e.g. a local score of 8 out of 10 renders a Metascore of 80). These scores are weighted (based on the quality and overall stature of the source) and finalized into a professional Metascore.
Regular non-professional users are also allowed to score the media on a scale of 0 to 10. The unweighted average of this score is presented by Metacritic as the Userscore. Non-professional users can also post their own reviews along with their score.
The user score is divided into three tiers: Positive, Neutral and Negative, where Positive is ratings 8 to 10, Neutral is ratings 5 to 7, and Negative is ratings 0 to 4. The rating tiers are color coded in green for Positive, yellow for Neutral and red for Negative. We will proceed scraping these ratings from the Metacritic website (https://www.metacritic.com).

3.2.1 Web Scraping

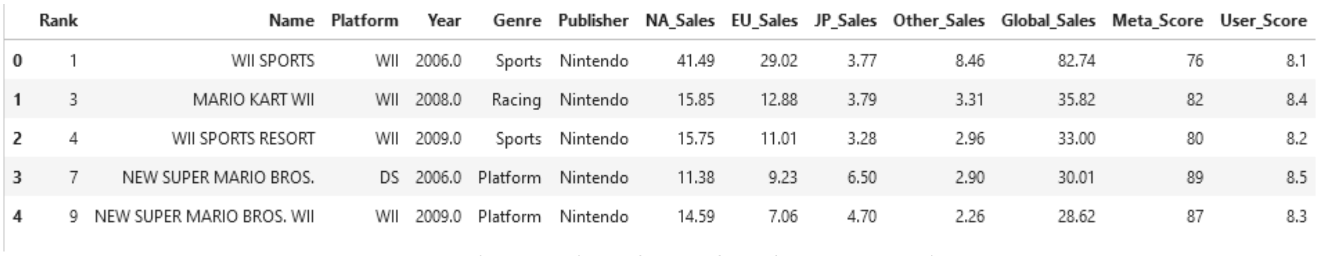
A notebook preparing for scraping the metacritic's webpage has been preprared using the library bs4 from BeatifulSoup and urllib.request (see annex).
When grouping by year A first look at the webpage, we have noticed critics are made for games released from 1996 to 2021. Typically, the webpage lists their database with an url as follows:
https://www.metacritic.com/browse/games/score/metascore/year/all/filtered?year_selected=2020&distribution=&sort=desc&view=detailed&page=5

Scraping is a long process. To avoid overloading the scrapped page with our requests, we have carried it out by 5-year period creating thus databases df0 to df4. Using a “for” loop, each url has been called upon by the algorithm for scraping. We have also used the sleep function from the time library to limit the impact of Metacritic's server.
To select the data within the webpage, we have used SelectorGadget for Chrome. This powerful tool has allowed us to select for each title the platform, year (of release), Meta score and User score for the 18817 titles in Metacritic's database.
We have combined our scraping data with other data scraped by other people found on Kaggle. The final database has a volume of 50 000 rows with after dropping the duplicated lines and the NaNs amount to the final 40 000 lines.

3.2.2 Databases Merging

The databases vgsales and Meta_vg will be merged so that all data is available in a single set for later operations. Given that the same title may be available for different platforms, we will use the triple criteria of matching columns “Name”, “Platform” and “Year” to select the rows to join. To maximize the number of matches, we previously proceed to normalize the fields “Name” and “Platform” as NFKD (to remove tildes, dieresis, …) and enforce uppercasing. See annex for further details.

Our resulting database is named “VG_Meta_Score.csv” and has 5044 rows and 13 columns.



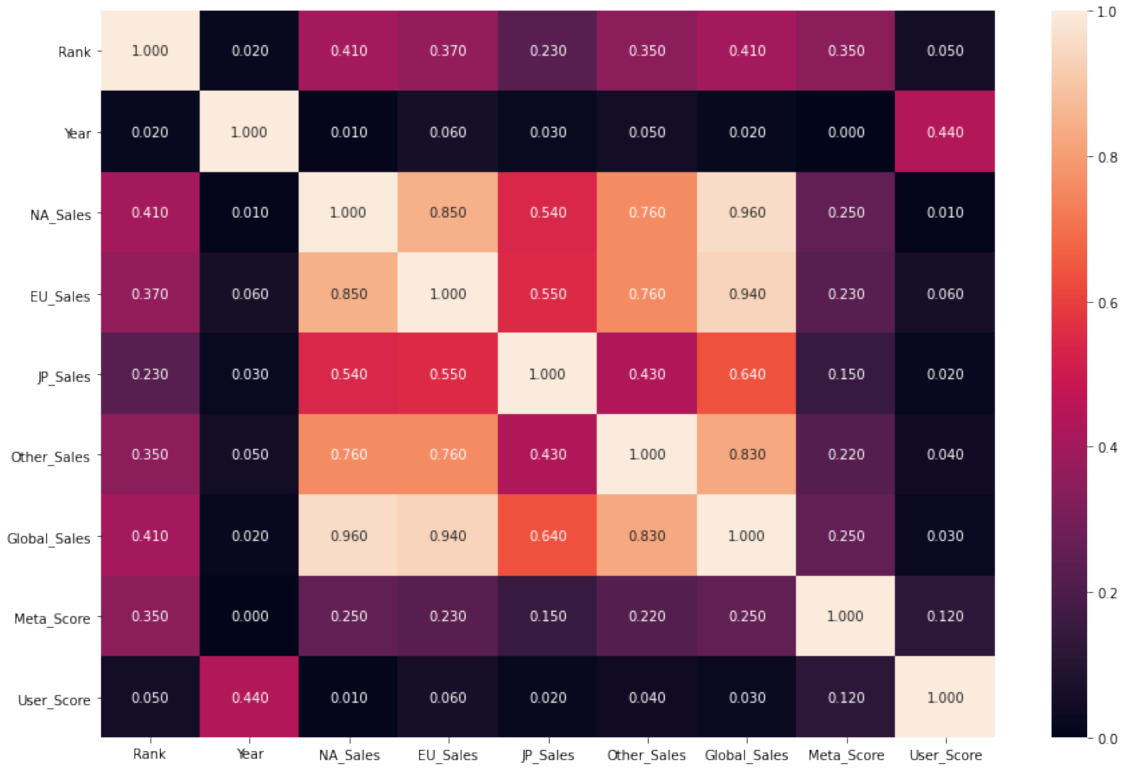
	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Meta_Score	User_Score
0	1	WII SPORTS	WII	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74	76	8.1
1	3	MARIO KART WII	WII	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82	82	8.4
2	4	WII SPORTS RESORT	WII	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00	80	8.2
3	7	NEW SUPER MARIO BROS.	DS	2006.0	Platform	Nintendo	11.38	9.23	6.50	2.90	30.01	89	8.5
4	9	NEW SUPER MARIO BROS. WII	WII	2009.0	Platform	Nintendo	14.59	7.06	4.70	2.26	28.62	87	8.3

Figure 3 – Extract from VG_Meta_Score (merged database)

3.3 Data cleaning and methodology.

A look at the dataset confirms that there are non NaN values.

A correlation analysis between the variables (see Figure 4) shows that the NA_Sales, EU_Sales, JP_Sales, and Other_Sales are the highest correlated with target variable Global_Sales, and even between each other. These higher correlations between sales variables help us to understand the global nature of the video game industry, so success on one continent usually means success on another as well. This finding support selecting the Global_Sales variable to be used as a dependent variable. There is also very weak correlation between Year and the other numeric variables.



	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Meta_Score	User_Score
Rank	1.000	0.020	0.410	0.370	0.230	0.350	0.410	0.350	0.050
Year	0.020	1.000	0.010	0.060	0.030	0.050	0.020	0.000	0.440
NA_Sales	0.410	0.010	1.000	0.850	0.540	0.760	0.960	0.250	0.010
EU_Sales	0.370	0.060	0.850	1.000	0.550	0.760	0.940	0.230	0.060
JP_Sales	0.230	0.030	0.540	0.550	1.000	0.430	0.640	0.150	0.020
Other_Sales	0.350	0.050	0.760	0.760	0.430	1.000	0.830	0.220	0.040
Global_Sales	0.410	0.020	0.960	0.940	0.640	0.830	1.000	0.250	0.030
Meta_Score	0.350	0.000	0.250	0.230	0.150	0.220	0.250	1.000	0.120
User_Score	0.050	0.440	0.010	0.060	0.020	0.040	0.030	0.120	1.000

Figure 4 - Correlation matrix from appendix

3.3.1 Methodology

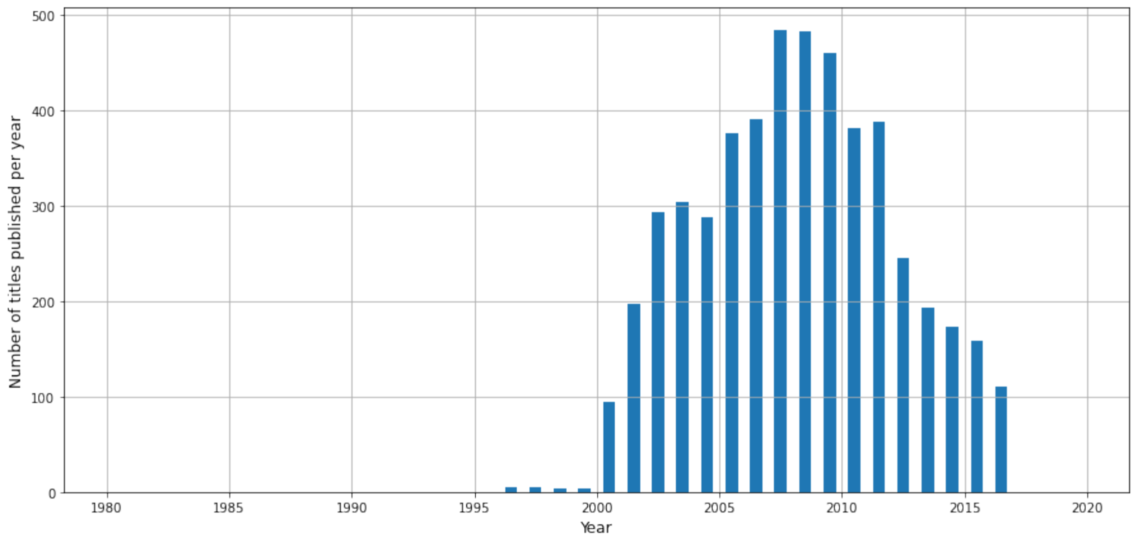
- For a first approach, numeric variables NA_Sales, EU_Sales, JP_Sales, and Other_Sales will be disregarded. We will keep only “Global Sales” as it is our target variable.
- We will proceed later with a second approach where NA sales will be taken into account. This will cover the case where a game is released in one region ahead of the release in another region. We expect this will increase the score of the predictions.

- Finally, a further study will be performed where alternative modelling options will be presented that may improve these scores.

Rank variable is also dropped as meaningless in the context of this database.
We also proceed dropping the variable "Publisher". This is a categorical variable with 204 unique values. We will apply a get_dummies method to the dataframe and keeping this categorical variable will generate a high number of columns that will be generate unnecessary processing time **for this iteration**.

3.3.2 Numerical Variables.

A quick look at the units sold in the database confirms the removal of all data before year 1996. We can also see we do not have data available from year 2018 onwards.



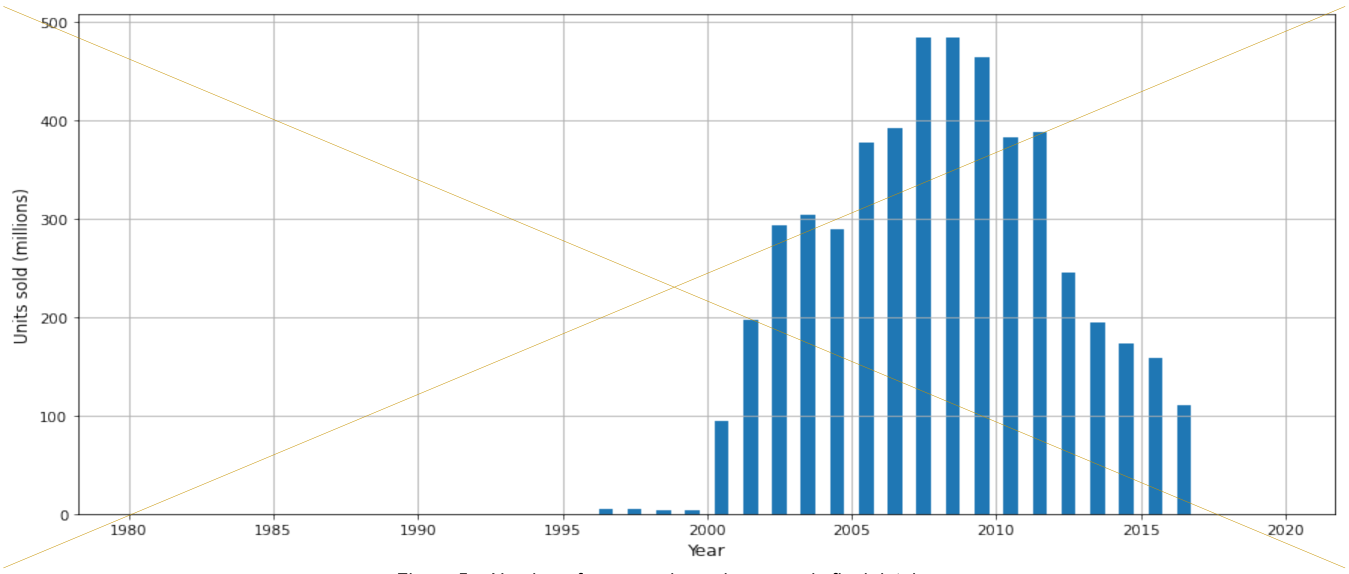


Figure 5 – Number of games released per year in final database

It is worth to note that figure above shows the units sold of a title per year of game release. The fact that there is a change of tendance between year 2008 and 2009, and that data is available up to year 2017, indicates the sales cycle of a game is around 8-9 years (to be confirmed by further study).

3.3.3 Categorical Variables.

The rest of the variables (Platform and Genre) are categorical. They count 16 and 12 unique values respectively. We will plot ~~Global_Sales~~the count of the number of titles against each categorical variable to better understand their relations. Please refer to appendix for further details.

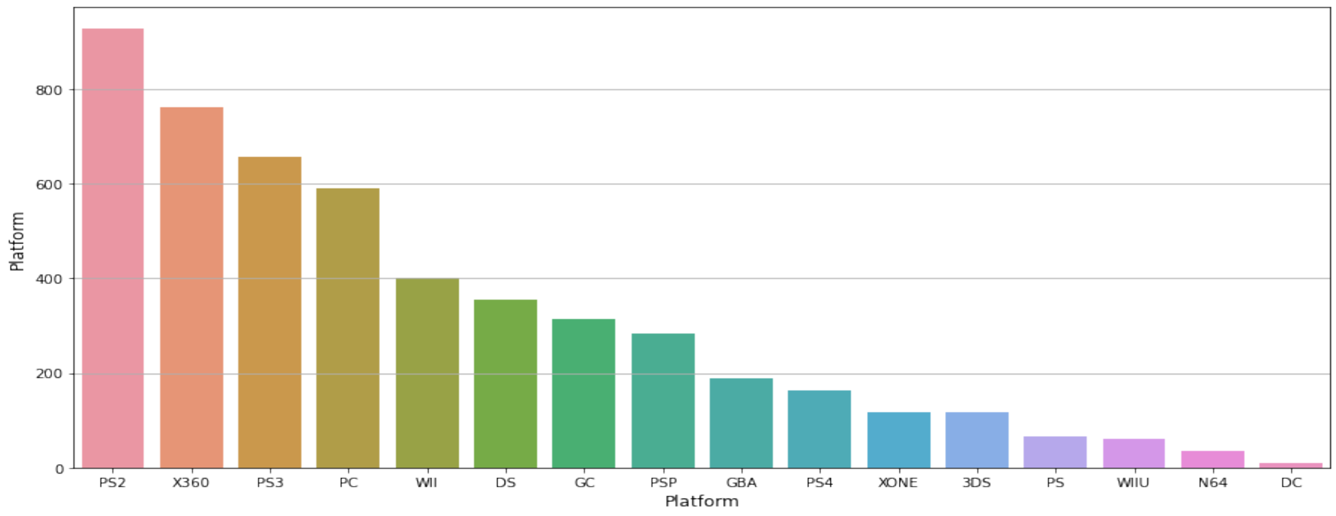


Figure 6 - Number of ~~units~~ games released~~sold~~ grouped by platform

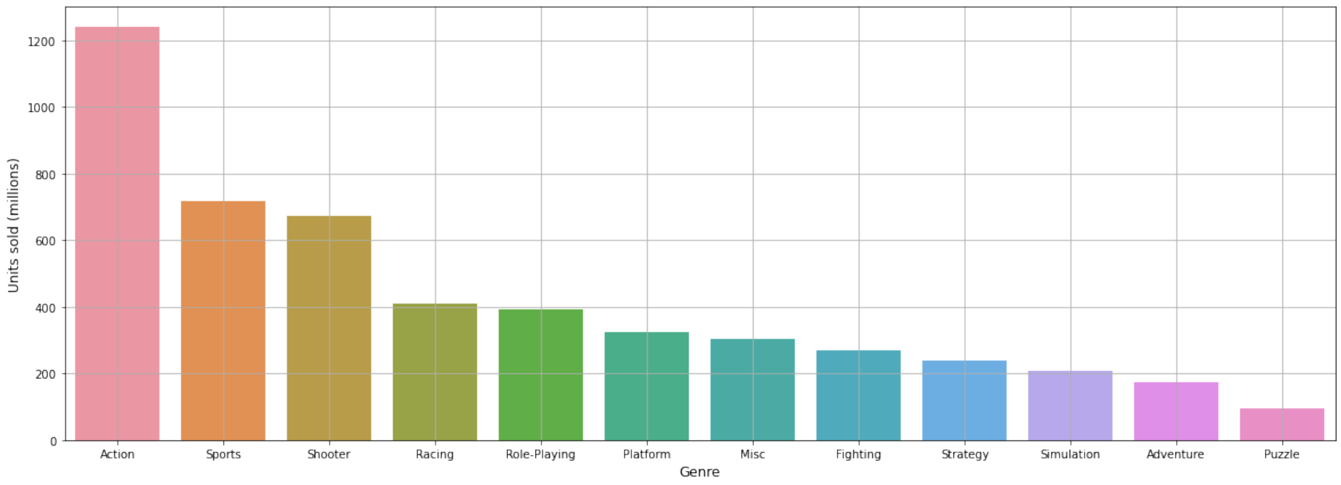


Figure 7 - Number of ~~units~~ games released~~sold~~ grouped by genre

4. Initial Approach

4.1 Description

Our target is to predict global sales of a video game which is a continuous variable. This limits our model choice to Regression model. To enlarge the number of models available we will also try to transform the Regression problem into a classification problem (Logistic Regression).

4.1.1 Data Preprocessing:

As we have noted above, no NaNs are present and all data is ready. We have therefore proceeded with the following steps (see for further details):

- Apply the *pd.get_dummies* function to the categorical features (Platform and Genre).
- Divide the database between the target variable and the feature variables.
- As values range at very different scales for all variables, apply Standard Scaler to normalize them:
- Proceed with a train-test split of 20%.
- **Proceed applying a logarithmic transformation to the target label: This improves results.**

4.1.2 Linear Regression Model:

The above set has been trained using a liner regression model with the following results:

score train : 0.27
score test : 0.28
mse test: 0.37

4.2 Model Selection and Optimisation

4.2.1 Gradiant Boosting Regressor with GridSearchCV

We have performed a Gradient Boosting Regressor algorithm to the data together with a grid search to find the parameters that yield the best results. We have tried the following grid:

- Maximum number of features: 6, 8, 10, 12.
- Maximum depth: 5, 7 and 9.

The best parameters are : {'max_depth': 5, 'max_features': 8}, achieving a test score of 0.57.

4.2.2 Optimisation of the Linear Regression using Lasso.

We have used a Lasso model using different alphas ([10, 1, 0.1, 0.01, 0.001, 0.0005]) to find the best value for alpha. We obtain the best alpha is 0.0005, and that there are ~~3~~ 2 variables (out of 31) to be eliminated.
The scores are:
score test: 0.~~28~~30
mse test: 0.~~37~~50, showing very little improvement from those obtained in 4.1.2.
~~Therefore we will try a Logistic Regression model.~~

4.2.3 Logistic Regression Model:

The set of data has been divided in four bins (quantiles): Silver, Gold, Platinum and Diamond according to the global sales. With a maximum number of iterations of 1000, ~~T~~the score with this approach goes up to 0.~~473~~484.

4.2.4 SVM Model:

Similarly the data has been trained on a SVM model yielding a slightly lower score of 0.440.

~~5.~~ Second Approach

5.1 Description

A similar approach to that following in section 4. is followed, but this time we will consider in our models the regional sales. The notebook including the model described in this section can be found in annex .

5.1.1 Data Preprocessing:

- NA sales~~, EU_Sales, JP_Sales andv Other_Sales~~ data ~~are~~ is this time kept as feature~~s~~. For the missing values, we will use the SimpleImputer function from sklearn. We will replace numerical data with the mean value for columns ['Critic_Score', 'Critic_Count', 'User_Score', 'User_Count'].
- We will also use the categorical_imputer function from feature_engine.imputation library to replace NAN values in the ['Genre','Rating'] columns by the most frequent value.
- Proceed with a train-test split of 20%.

5.1.2 XGBRegressor model

We implement the regressor using XGBRegressor (where XGB stands for extreme gradient boosting). XGBoost is an ensemble machine learning algorithm based on decision trees similar to the RandomForest algorithm. However, unlike RandomForest that makes use of fully grown trees, XGBoost combines trees that are not too deep. Also, the number of trees combined in XGBoost is more in comparison to RandomForest. Ensemble algorithms effectively combine weak learners to produce a strong learner. XGBoost has additional features focused on performance and speed when compared to gradient boosting.
We have set the parameters for the number of estimators to 200 and the learning rate at 0.08.
We obtain a test score of 0.89, a r2 score of 0.893 and a RMSE of 0.503.

~~5.1.3 XGBRegressor Model~~

~~We implement the regressor using XGBRegressor (where XGB stands for extreme gradient boosting). XGBoost is an ensemble machine learning algorithm based on decision trees similar to the RandomForest algorithm. However, unlike RandomForest that makes use of fully grown trees, XGBoost combines trees that are not too deep. Also, the number of trees combined in XGBoost is more in comparison to RandomForest. Ensemble algorithms effectively combine weak learners to produce a strong learner. XGBoost has additional features focused on performance and speed when compared to gradient boosting.~~
~~We have~~
~~set the parameters for the number of estimators to 200 and the learning rate at 0.08.~~
~~We obtain a test score of 0.89, a r2 score of 0.893 and a RMSE of 0.503.~~

~~5.1.4~~ Gradient Boosting Regressor with GridSearch

We have performed a Gradient Boosting Regressor algorithm to the data together with a grid search to find the parameters that yield the best results. We have tried the following grid:

- Number of estimators: 200 and 500
- Maximum number of features: 6, 8, 10, 12.
- Maximum depth: 5, 7 and 9.

The best parameters are : {'max_depth': 5, 'max_features': 12, 'n_estimators': 500}.
We obtain a best cross-validation score of 0.72 and a test score of 0.55 ~~but we only achieve a test score of 0.38~~.

5.1.5 Linear Regression model:

Similarly to the previous section, the above set has been trained using a liner regression model with the following results:

score train :	0.~~99~~89
score test :	0.~~99~~90
mse test:	0.49

5.1.6 Logistic Regression model:

The set of data has been divided in four bins: Silver, Gold, Platinum and Diamond according to the global sales. The test score with this approach is 0.~~90~~74.

6. Further work to improve the modelling.

6.1 Influence of outliers

Review of the data shows the presence of outliers - see boxplot in figure below:

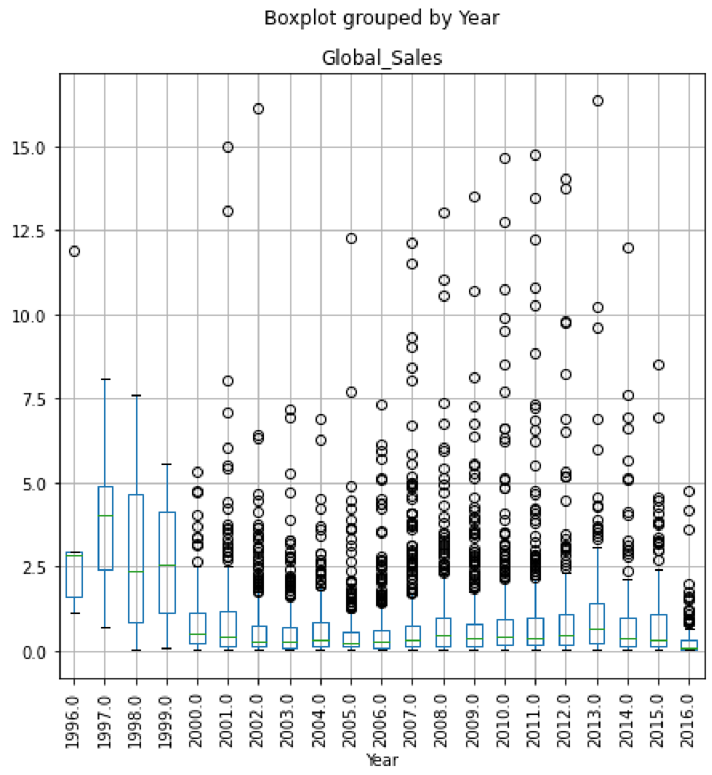


Figure 8 - Boxplot of data grouped by year

The analyses described in previous sections have been repeated by gradually removing the outlyers down to global sales of 8.0 million copies and above. The obtained scored are included in the notebook in section showing that the removal of outliers data does not impact significantly the results.

6.2 Multi-layer back propagation neural network model approach.

We also propose a multi-layer back-propagation neural network model based on (Rhee & Zulkernine, 2016). This work proposes a neural network for the prediction of whether or not a film will become a hit (binary classification). Our case is a 4-class classification meaning that we would expect slightly worse scores.

6.2.1 Neural network model.

We propose a network which features 235 variables across 3 layers:
- 1 input layer of 64 nodes.
- 1 hidden layer of 16 nodes.
- 1 input layer of 4 nodes, one for each category.

Activation function is set to relu except of the output layer which is set to softmax given the class nature of the output. The data has been normalised using a MinMaxScaler which presents better results than the StandardScaler.

6.2.2 Results

Results show that the trained data shows a score of 0.67- However after comparing with the test set, a score of 0.51 is achieved, denoting some degree of overfitting as anticipated. The confusion matrix is shown in Table 1

Classes	**0**	**1**	**2**	**3**
0	154	70	33	13
1	50	104	51	25
2	32	71	82	79
3	16	20	36	173

Table 1 - Confusion matrix

The figures above show that the algorithm is better to predict low and high sale classes than intermediate classes.

7. Description of tasks carried out

7.1 Repartition of tasks by team member and along the project.

Iteratio n	**Task nr.**	**Task performed in chronological order**	**Main contributor**
I		Initial data exploration	A. Hasan
		Web scraping – modelling	A. Hasan / C. Gallardo
		Web scraping – reporting	C. Gallardo
II		Github setup and maintenance	C. Gallardo
		First approach - modelling	A. Hasan
		First approach – reporting	C. Gallardo
		First approach – report review	A. Hasan
III		Second approach - modelling	A. Hasan
		Second approach - reporting	C. Gallardo
		Second approach – report review	A. Hasan
		Further study – Influence of outllers	C. Gallardo
		Further study – Neural network model	C. Gallardo
IV		Final reporting	A. Hasan / C. Gallardo
V		Presentation / StreamLit	A. Hasan / C. Gallardo

7.2 Main encountered difficulties

The global sales of a title depend on many subjective parameters and not only on some features related to the games themselves. It also depends on other factors like the audience they are intended for, the time of the year they are released and the political, economic and social characteristics of the market they are released into.

To take into account the above, we have used not only the historical data but also score reviews from specialized websites. But the amount of data is available upto 2016. We would need to complete the data which is not readily available for free.

8. Bibliography

Galvão, M., & Henriques, R. (2018). Forecasting Movie Box Office Profitability. Journal of Information Systems Engineering & Management. *10.20897/jisem/2658*, (p. 3).

Quader, N., Gani, M., & Chaki, D. (2018). Performance evaluation of seven machine learning classification techniques for movie box office success prediction. *10.1109/EICT.2017.8275242*, (pp. 1-6).

Rhee, T., & Zulkernine, F. (2016). Predicting Movie Box Office Profitability: A Neural Network Approach. *10.1109/ICMLA.2016.0117*, (pp. 665-670).

9. Conclusions and further study.

9.1 Conclusions

A summary with the test results for each model we have built is shown in

Situation	**Model**	**Test Score**
Initial (regional North American sales unknow)~~n)~~	Linear Regression model	0.28
	Gradiant Boosting Regressor with GridSearchCV ~~**Linear Regression model with Lasso**~~	**0.57~~0.28~~**
	Linear Regression model with Lasso	0.28
	Logistic Regression (4 bins)	0.48~~7~~
	SVM model	0.44
Second (regional North American sales known)	XGBRegressor	0.80
	~~XGBRegressor~~ Gradient Boosting Regressor with GridSearchCV	~~0.89~~ 0.~~38~~55
	Linear Regression model	**0.90~~9~~**
	Logistic Regression (4 bins)	0.~~90~~74
Additional (regional North American sales unknow)	Multi-layer back propagation neural network model	0.51

Table 2 - Summary of results

In the case the regional sales for NA is~~are~~ known, then the Linear Regression model works best. In the other hand, when the NA ~~if the~~ regional sales is unknown, we obtain better results using a Gradiant Boosting Regressor, followed very closely by the multi-layer back propagation neural network model. ~~Multi-layer back propagation neural network model. However, and given that neural networks introduce overfitting~~

9.2 Further study

9.2.1 Twitter counts

It has been found that the number of tweets generated the week preceding the release of a movie is critical to the box office revenue (Rhee & Zulkernine, 2016). This same principle can be applied to video games. We present here two free tools to scrape tweets:

- snscrape: it is a scraper for social networking services like Tweeter, Facebook, Reddit. It scrapes user profiles, hashtags, or searches and returns the discovered items. In we have tried an example with videogame "Red Dead Redemption 2", tweets 48 hours before release. We collect 6132 tweets.
- twint: is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API. We have tried the same search as above and obtain 60 tweets. A second search yields 100 tweets. We reach the conclusion that this tool is not reliable.

9.2.2 Other options for further study

- ~~We think that social data like the number of tweets generated by each title one week before release can contribute to better predictions.~~

- Reduce overfitting by dropping features in the neural network.
- Use the name’s title as a feature (e.g. games with the name of Mario or FIFA traditionally sell well).

10. Annexes

10.1 VG_Project.ipynb

A1_VG_Project.ipynb vgsales.csv

10.2 Web scraping.ipynb

A2_Web scraping.ipynb

MetaDF.csv

10.3 Merging of vgsales and Meta_vg databases – Project_VG_1.ipynb

A3_Project_VG_1.ipyn
b

VG_Meta_Score.csv

10.4 PYGSL.ipynb

A4_PYGSL.ipynb

Video_Games_Sales_as_at_22_Dec_2016.csv

10.5 PYGALSAL_CNN~~-~~.ipynb

A6_PYGALSAL_CNN.ipynb

df_clean.csv

10.6 A6_Twitter.ipynb

A6_Twitter.ipynb