“Video Games Sales Analysis”

Project Report
Submitted for the course: Machine Learning(CSE4020)

By

APOORV TYAGI (16BCE0097)

NITISH SURANA (16BCE0526)

Slot: F2

Name of Faculty: Prof. Vijayashirley

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

Abstract

Video games are popular all over the world. They are enjoyed by all ages. Video game industry is huge and the spending on video games per year is huge too. Sales of different types of games vary widely between countries due to local preferences. According to the market research firm Super Data, as of May 2015, the global games market was worth USD 74.2 billion. By region, North America accounted for 23.6 billion dollars, Asia for 23.1 billion dollars, Europe for 22.1 billion dollars and South America for 4.5 billion dollars. There are different genres, publisher and platforms for video games. This project relates to the sales of these video games based on different regions and analyzes the sales. Also, we have analyzed which genre, platform or publisher is the most popular and has maximum number of sales. Some areas worth exploring can be stated as can range from Titles which are available for more than one platform, Top contending platform, which type of platform is most popular, top selling genres ,t op publishers by Global Sales, etc.

Table of Contents

1. INTRODUCTION

The video games market has seen a large growth since its inception in 1970s to the point where video games have become a daily form of entertainment for many people of all ages around the world. It is a highly profitable market, reaching $16.5 billion in U.S. sales in 2015. In comparison, the movie industry sold $29.2 billion in 2015 in the U.S. The PC games market has a seen an increase in digital sales and the number of releases after Valve launched its Steam Store1. This store saw a major growth after 2012 thanks to a program called Greenlight, allowing developers to relatively easily release their games on Steam without a publisher which had been very difficult until then. Over time, greenlight led to a massive increase in the number of releases on the store, reaching over 10,000 by the end of 2016. As a result, it became increasingly difficult for developers to stand out and even sell enough copies to fund the development of their games. If the success of games could be predicted beforehand, it would allow game creators to adjust the development to attract a larger audience or perhaps forsake their attempt to develop a game if it did not lead to a success. Thus, it would be useful to evaluate a concept; not an already finished game. Previous attempts at video games success prediction assumed the game was already released and made predictions based on that knowledge. In addition, they dealt mostly with pre-2010 console games when there was no incentive to study the PC games market. The goal of this thesis is to study known factors affecting the success of video games, create a database of PC games as no suitable one is available and, finally, estimate a game's success based on descriptive information such as genre, price, developer, or game features.

2. LITERATURE SURVEY

2.1 Major Studies

IEEE Conference on Computational Intelligence and Games1 stands out as a prominent source of papers dealing with applying machine learning methods in video games development. The most common topics include automatic content generation and agent planning. While the topic of predicting revenue is not present, there are papers utilizing machine learning methods to predict factors related to games' success, such as retention or player experience.

2.1.1 Predicting Retention in Sandbox Games with Tensor Factorization-based Representation Learning

The authors processed data about at what times players were playing and what they were doing within the game. The goal was to learn from 14 days of activity and predict if the players keep playing after the following 7 days. The study is heavily focused on spatio-temporal data, i.e. how players travel within the game and how to process this data. Ensemble methods were mostly used for evaluation, achieving precision of 81 % and recall of 75 % in the best case.

2.1.2 Predicting Player Churn in Destiny: A Hidden Markov Models Approach to Predicting Player Departure in a Major Online Game

The authors of this paper used very detailed data about player activities in a major title, Destiny. The data span across 17 months and included the activities of 10,000 players. Similarly, to the previous study, the goal was to predict whether a player quits the game after a certain time window, in this case 4 weeks. They focused on the use of multinomial Hidden Markov Model which returned the highest 1. http://www.ieee-cig.org/ 3 2. Related Work precision of 92 % with a relatively low recall of 43 % compared to other models.

2.1.3 Transfer Learning for Cross-Game Prediction of Player Experience

The paper describes learning of how players experience one game and making predictions about their experience in another game. The authors used statistical summarization of what players were doing and how well they were performing in two games. Players were then asked about their experience, namely engagement, frustration and challenge. The authors used two methods for the task of automatically mapping features between games, referred to as "supervised feature mapping" and "unsupervised transfer learning". Both methods produced accuracies above 58 % and 55 %, respectively, achieving 83 % accuracy on one of the subtasks (predicting challenge). These results were comparable with manual mappings created by experts.

2.2 Other Studies

There are some studies, typically conducted by university students, which attempt to predict sales figures. However, they often lack a proper description of the data or results. Nevertheless,

these are, to our best knowledge, the only publicly available studies on the topic of sales prediction.

2.2.1 Predicting Video Game Sales in the European Market

This study focused on game and console sales in Europe from March 12, 2005 to December 31, 2011. The authors used data about 2,450 games. The dataset contained 9 attributes and sales which they were attempting to predict. Simple regression models were fitted to predict weekly sales based on the first 2-6 weeks of sales. A prediction method for total sales was manually crafted and tested on all the data.

2.2.2 Predicting Video Game Sales Using an Analysis of Internet Message Board Discussions

The aim of this thesis was to collect gaming forum posts and use this data to predict sales of video games. The data was collected from 2008 and 2009 from a major gaming message board. The author extracted mentions of each game and used the number of these mentions as well as sales from previous two weeks to predict sales in the upcoming weeks. The only evaluation metric used is Mean Absolute Error, making any conclusion of the results difficult

2.2.3 The Game Prophet: Predicting the success of Video Games

This study was using data about US, Japan, and European sales from 2001-2008. There were six attributes and “site hits”, not further specified. The sale numbers were divided into 6 classes. The accuracy was mostly around 50 %, ranging from 70 % to 85 % based on the region when allowing a deviation by 1 class.

Discussion

All known studies focusing on prediction of sales use data from VGChartz. The site does not clearly state the origin of the data nor what kind of sales exactly they are tracking. Even if sales for consoles were accurate, the sales of PC games are clearly inaccurate. Some of the largest hits in the recent years, Stardew Valley3 and Undertale4, sold no units whatsoever according to the site. Examining the database suggests that VGChartz does not track digital sales, making it a completely unsuitable source of games’ success on the PC platform. We can only speculate about why there are no papers properly investigating sales prediction. There is a clear issue with the lack of accurate sales figures as publisher do not want to release exact numbers. This is understandable as it could, for instance, negatively affect stock prices. It is worth noting that the data used in the papers from the IEEE CIG conferences are directly provided by publishers and employees of these publishers are involved in these studies. A study on revenue prediction would have to be conducted with the help from a publisher. Understandably, this publisher would like to keep any results to themselves as revenue would be directly involved. Not only not to reveal any precise numbers but also to use these findings to gain an advantage over their competition.

3. OVERVIEW

3.1 Introduction

In this the main goal was to analyze the sales of video games in different regions. The regions are North America, Europe, Japan, other countries(combined) and then the global sales (total of all the regions). The main idea was to visualize the sales for different genres, publishers and platforms. This would give the basic idea about the most popular genres, publishers and platforms amongst all. Also analyzing the effect of genres on sales in different regions. For this project the data on which we have worked on can be found on Kaggle with the name of the dataset being "Video Game Sales Analysis with Ratings". This data gives us the idea about the sales of video games in different regions of the world. The distribution is with respect to genres, publishers and platforms. The various attributes present in the dataset are listed below: -

- Name: Name of the video game
- Platform: Console on which the game is running
- Year of Release: Year in which the game was released
- Genre: Genre the game belongs to
- Publisher: Name of the publisher who created the game
- NA Sales: Sales in North America (in millions of units)
- EU Sales: Sales in the European Union (in millions of units)
- JP Sales: Game Sales in Japan (in millions of units)
- Other Sales: Sales in the rest of the world, i.e. Africa, Asia excluding Japan, Australia, Europe excluding the E.U. and South America (in millions of units)
- Global Sales: Total Sales in the world (in millions of units)
- Critic Score: Aggregate score compiled by Metacritic staff
- Critic Count: The number of critics used in coming up with the Critic score
- User Score: Score by Metacritic's subscribers
- User Count: Number of users who gave the User score
- Developer: Party responsible for creating the game
- Rating: The ESRB ratings (E.g. Everyone, Teen, Adults Only, etc.)

3.2 Proposed System Architecture

After loading the dataset in our Python IDE, the very first step would be data cleaning. Since the data is very raw in nature, getting proper visualization before cleaning will be hard. Hence, this is the first step in our architecture. After the data cleaning is performed, we perform data exploration (EDA) on the cleaned dataset. We have performed exploration using seaborn library as well as the Plotly Library.

After completing the data exploration, lot of valuable inference about the data can be gained. From the visualization we find that nearly half of the entries do not have scored attributes. Hence, we decide to make two different models in which the first model will remove all the entries in which score is not present while the other model will use weighted factor to control the scoring of the other entries. But before proceeding towards training the model, respective encoding schemes needs to performed on the categorical entities.

After splitting the data respectively, we plan to apply 8 different machine learning algorithms on the new dataset to find which model gives us the minimum error and the best fit for the data. The respective errors for the models are calculated and the model with the least error is taken. We further try to minimize the error output by finding out which are the right hyperparameters for the data. We used randomized searching after this to which the number of iterations required to get the best output after placing the hyperparameter grid respectively. The final output after fitting the respective hyperparameters are hence calculated and compared with the initial error value. This procedure followed for both the model and the final inferences are presented.

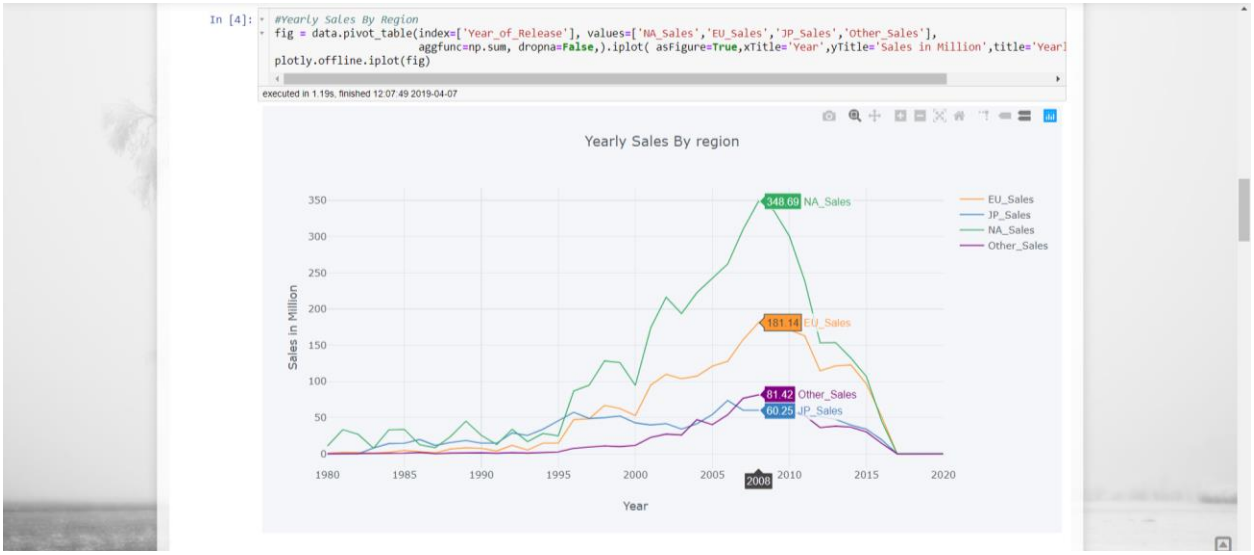
4. PROPOSED SYSTEM ANALYSIS

Data Preprocessing and Cleaning

The data is first downloaded from Kaggle (https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings) and then load into our IDE. We then explore the data and clean it respectively to benefit us for training and Exploration of data. From looking at the data we can point out data which are categorical or nominal or ratio respectively.

Data Exploration

The data is then explored using data exploration libraries like Seaborn and Plotly. Various different kinds of plots are drawn to different inferences from the data to understand how they are linked with each other. An example is shown below, which compares sales of different regions across the globe.



Outlier Maintenance

After EDA, we can find that that are quite a number of outliers present in the dataset. But these outliers too have a logical sense attached to it. It is completely natural for any game to become a blockbuster hit and get sales in large number of millions, and also a game to crash and burn as a failure and earn nearly 0 to nothing. But due to these values present in the dataset, all the other games suffer a lot. Hence, we have removed the outliers temporarily from the dataset and have done the analysis on the remaining dataset.

Encoding

Since our main object is to maximize the global sales of the video games, we will have mostly continuous and Ratio data to deal with. There are different categorical data present in the dataset and since we aren't categorizing any data here, we will have to convert them into numeric values. We have implemented different type of encoding like "One-hot" Encoding scheme to plot all unique categories as vectors in 3D space as well as "Ordinal" Encoding scheme to order the categories as per needed.

Prediction Model

Before we start predicting model, we divide our dataset into training and testing data respectively. After splitting the data, we define two function which will be used to fit the models and calculate the respective errors obtained by different models. The machine learning models that are used on the dataset are: -

- Linear Regression
- Support Vector Regressor (SVR)
- K-nearest Neighbors (KNN)
- Gradient Boosting Regressor
- Multi-layer Perception (MLP) Regressor
- Random Forest Regressor
- Ridge Regression
- Lasso Regression

Hyperparameter Tuning

After applying the machine learning models that are defined above, we get different error values for the models. We find that the Gradient Boosting Algorithm gives us the least error and hence we plane to fit the hyperparameters to this model. The hyperparameters grid that we have defined are generally the loss function, maximum depth for the tree, minimum samples present in the leaf node, etc. After applying the Hyperparameter Tuning, we fit the model using the Randomized Grid Search to find the number of iterations needed. After this, we find the mean absolute error again and the result shows the decrease in the error value.

Results

At the end, we have different graphs to summarize the data and show how the results of the advanced model (Model with Weighted Score) and the basic model (Model without the weighted score) and how the errors vary among them.

5. IMPLEMENTATION

Main Libraries Used

- Pandas (for data analysis)
- NumPy
- Matplotlib (for plots)
- Seaborn (EDA)
- Cufflinks (EDA)
- Plotly (EDA)
- Category_encoders (Encoding for categorical subset in data)
- Sklearn.model_selection (Splitting of data in training and test set, Randomized search, and Grid Search)
- Sklearn.linear_model (Linear Regression, Lasso Regression and Ridge Regression)
- Sklearn.ensemble (Random Forest Regressor, Gradient Boosting Regressor)
- Sklearn.svm (Support Vector Regressor)
- Sklearn.neighbors (K nearest Neighbors)
- Skleran.neural_netword (Multi-Layer Perceptron Regressor)

Initial Overview of Data Set

```
data.describe(include="all")

#The top value in User_Score column is "tbd".
#There are high outliers in sales columns (NA, EU, JP, Other, Global) and User_Count column.
```

	Name	Platform	Year	Genre	Publisher	NA	EU	JP	Other	Global	Critic_Score	Critic_Count
count	16448	16448	16448.000000	16448	16416	16448.000000	16448.000000	16448.000000	16448.000000	16448.00000	7983.000000	7983.000000
unique	11429	31	NaN	12	579	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	Need for Speed: Most Wanted	PS2	NaN	Action	Electronic Arts	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	12	2127	NaN	3308	1344	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	2006.488996	NaN	NaN	0.263965	0.145895	0.078472	0.047583	0.53617	68.994363	26.441313
std	NaN	NaN	5.877470	NaN	NaN	0.818286	0.506660	0.311064	0.187984	1.55846	13.920060	19.008136
min	NaN	NaN	1980.000000	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.01000	13.000000	3.000000
25%	NaN	NaN	2003.000000	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.06000	60.000000	12.000000
50%	NaN	NaN	2007.000000	NaN	NaN	0.080000	0.020000	0.000000	0.010000	0.17000	71.000000	22.000000
75%	NaN	NaN	2010.000000	NaN	NaN	0.240000	0.110000	0.040000	0.030000	0.47000	79.000000	36.000000
max	NaN	NaN	2020.000000	NaN	NaN	41.360000	28.960000	10.220000	10.570000	82.53000	98.000000	113.000000

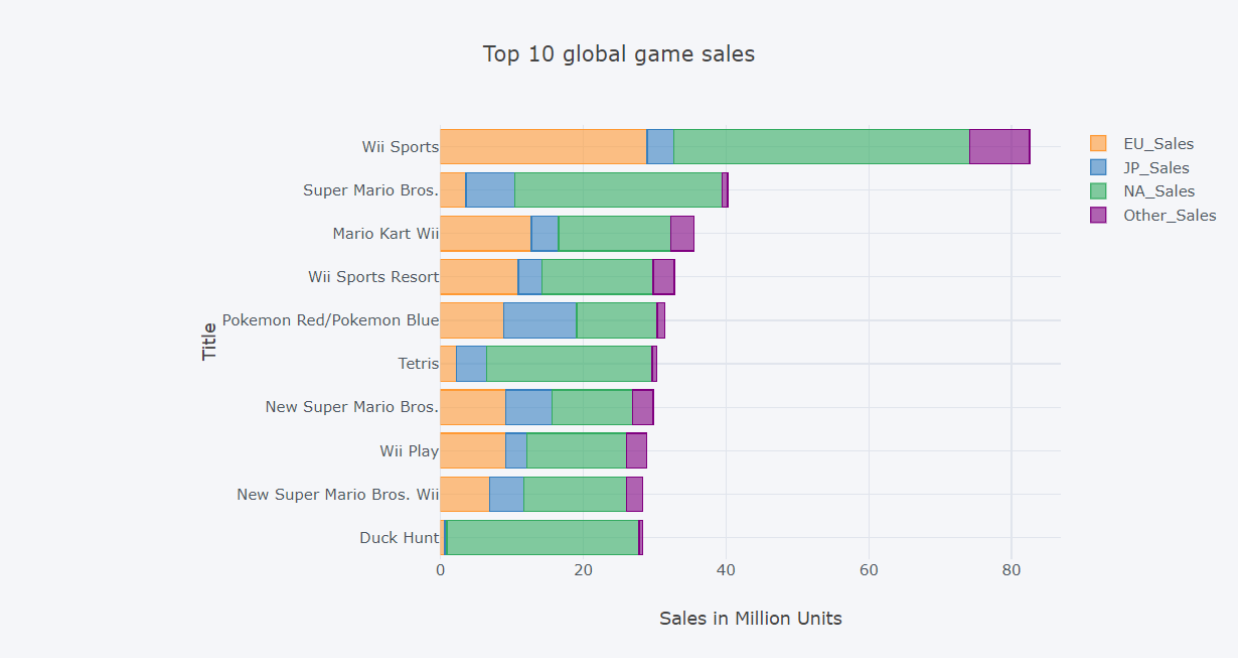
Overview of the Data after Cleaning

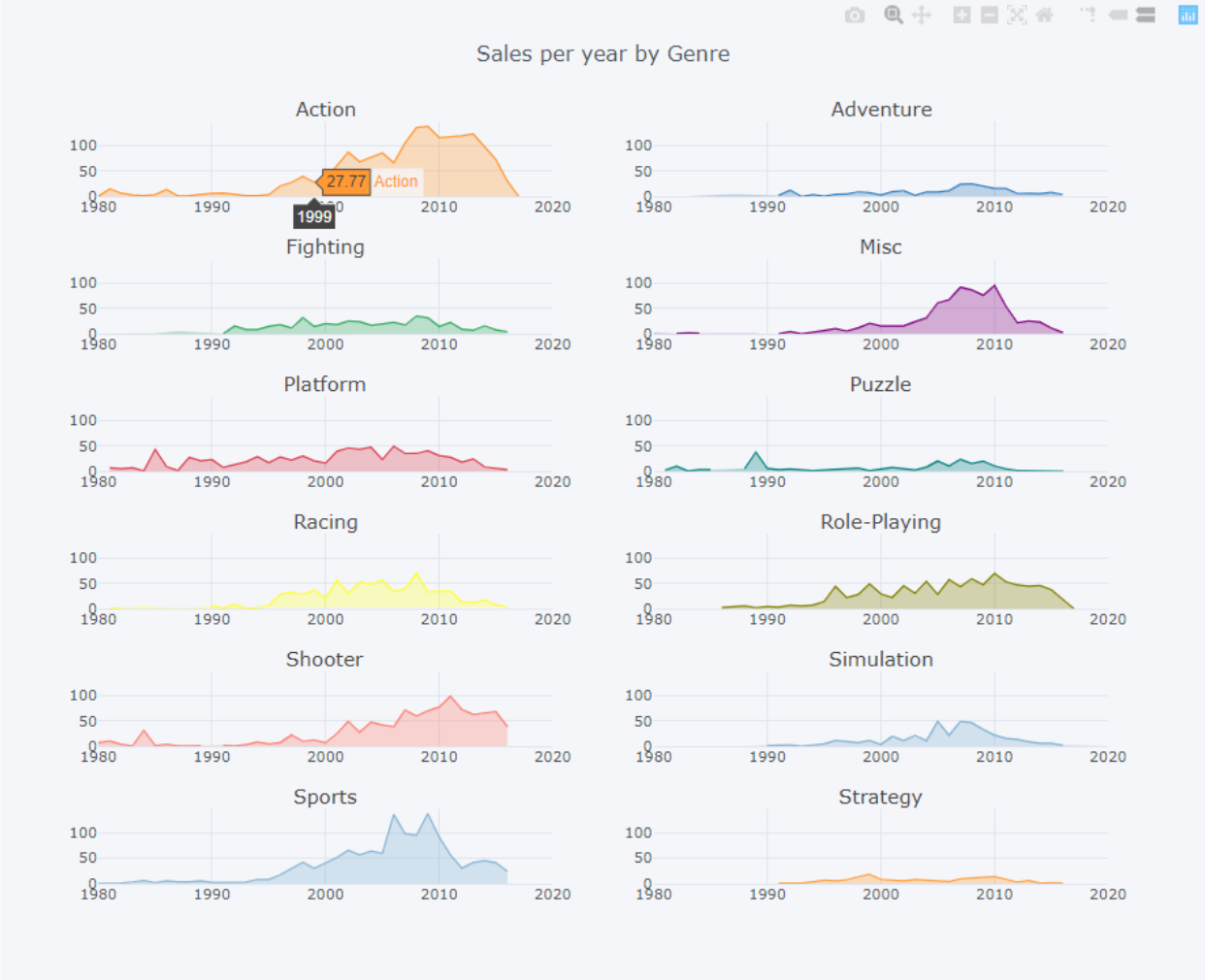
```
data = data.rename(columns={"Year_of_Release": "Year",
                            "NA_Sales": "NA",
                            "EU_Sales": "EU",
                            "JP_Sales": "JP",
                            "Other_Sales": "Other",
                            "Global_Sales": "Global"})
data = data[data["Year"].notnull()]
data = data[data["Genre"].notnull()]
data["Year"] = data["Year"].apply(int)
data["Age"] = 2018 - data["Year"]
data["User_Score"] = data["User_Score"].replace("tbd", np.nan).astype(float)
data.describe(include="all")
```

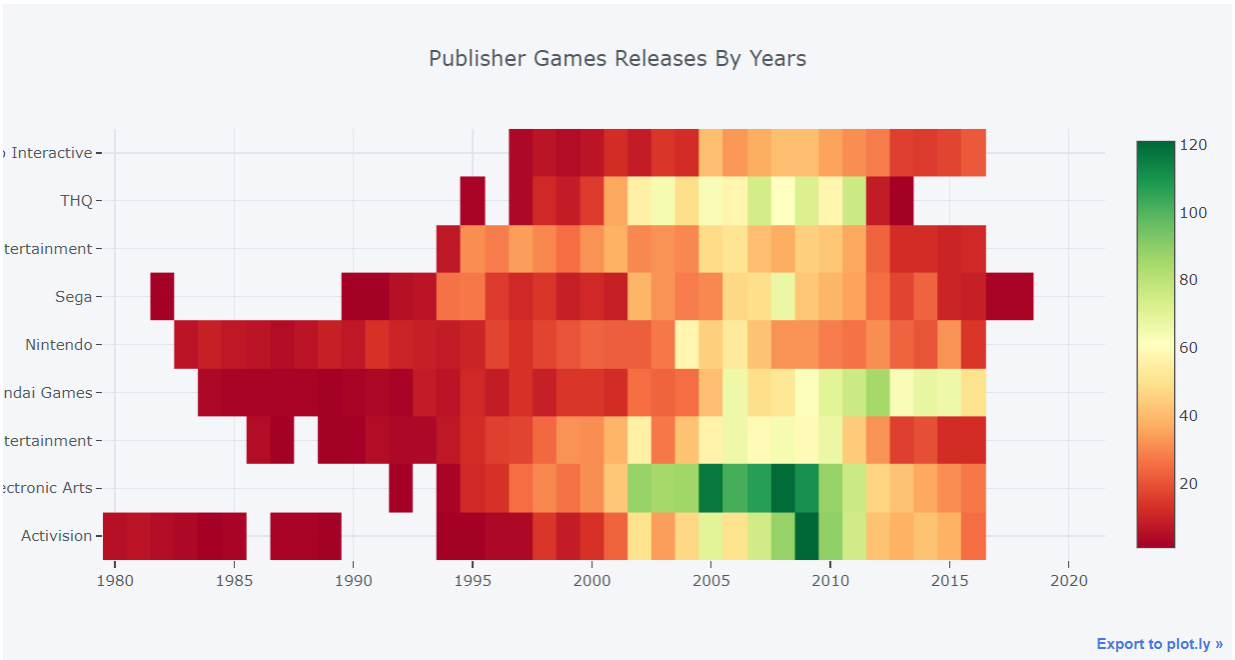
	Name	Platform	Year	Genre	Publisher	NA	EU	JP	Other	Global	Critic_Score	Critic_Count	U
count	16448	16448	16448.000000	16448	16416	16448.000000	16448.000000	16448.000000	16448.000000	16448.00000	7983.000000	7983.000000	7
unique	11429	31	NaN	12	579	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	Need for Speed: Most Wanted	PS2	NaN	Action	Electronic Arts	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	12	2127	NaN	3308	1344	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	2006.488996	NaN	NaN	0.263965	0.145895	0.078472	0.047583	0.53617	68.994363	26.441313	
std	NaN	NaN	5.877470	NaN	NaN	0.818286	0.506660	0.311064	0.187984	1.55846	13.920060	19.008136	
min	NaN	NaN	1980.000000	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.01000	13.000000	3.000000	
25%	NaN	NaN	2003.000000	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.06000	60.000000	12.000000	
50%	NaN	NaN	2007.000000	NaN	NaN	0.080000	0.020000	0.000000	0.010000	0.17000	71.000000	22.000000	
75%	NaN	NaN	2010.000000	NaN	NaN	0.240000	0.110000	0.040000	0.030000	0.47000	79.000000	36.000000	
max	NaN	NaN	2020.000000	NaN	NaN	41.360000	28.960000	10.220000	10.570000	82.53000	98.000000	113.000000	

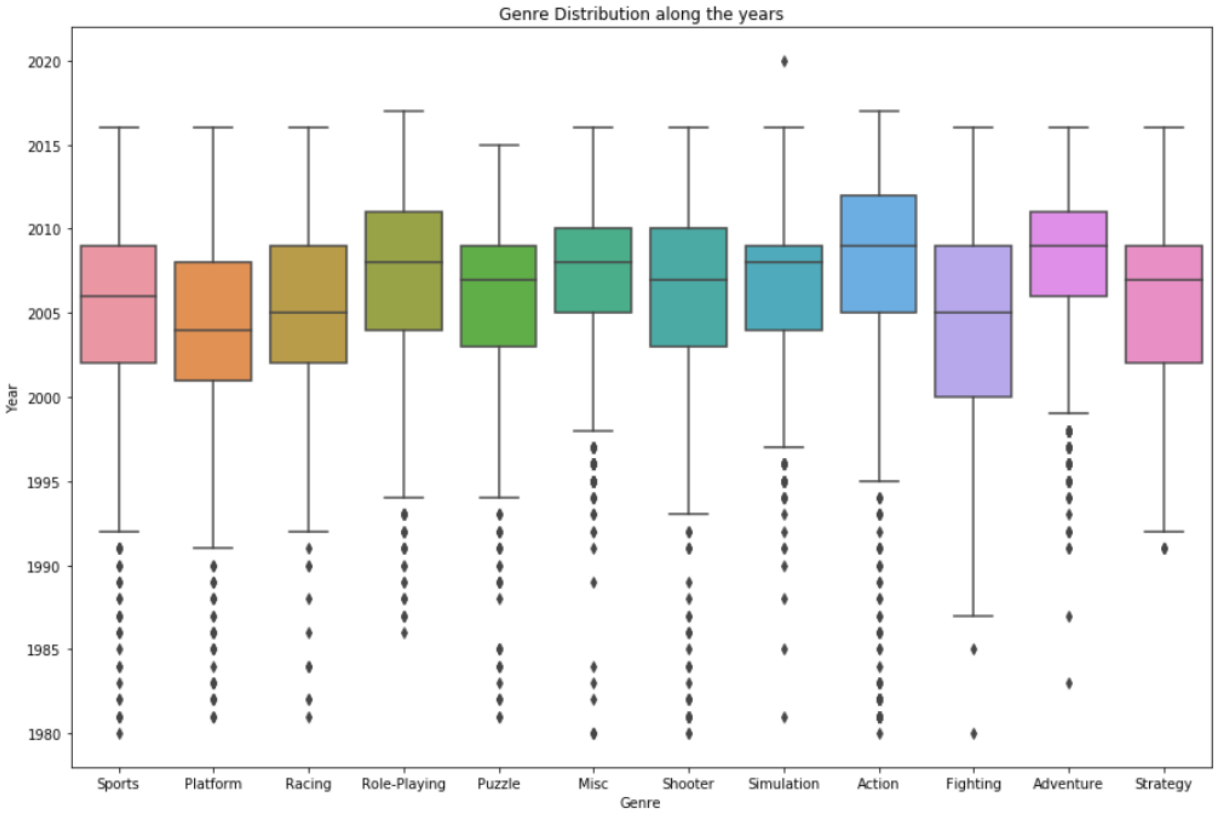
Data exploration and analysis

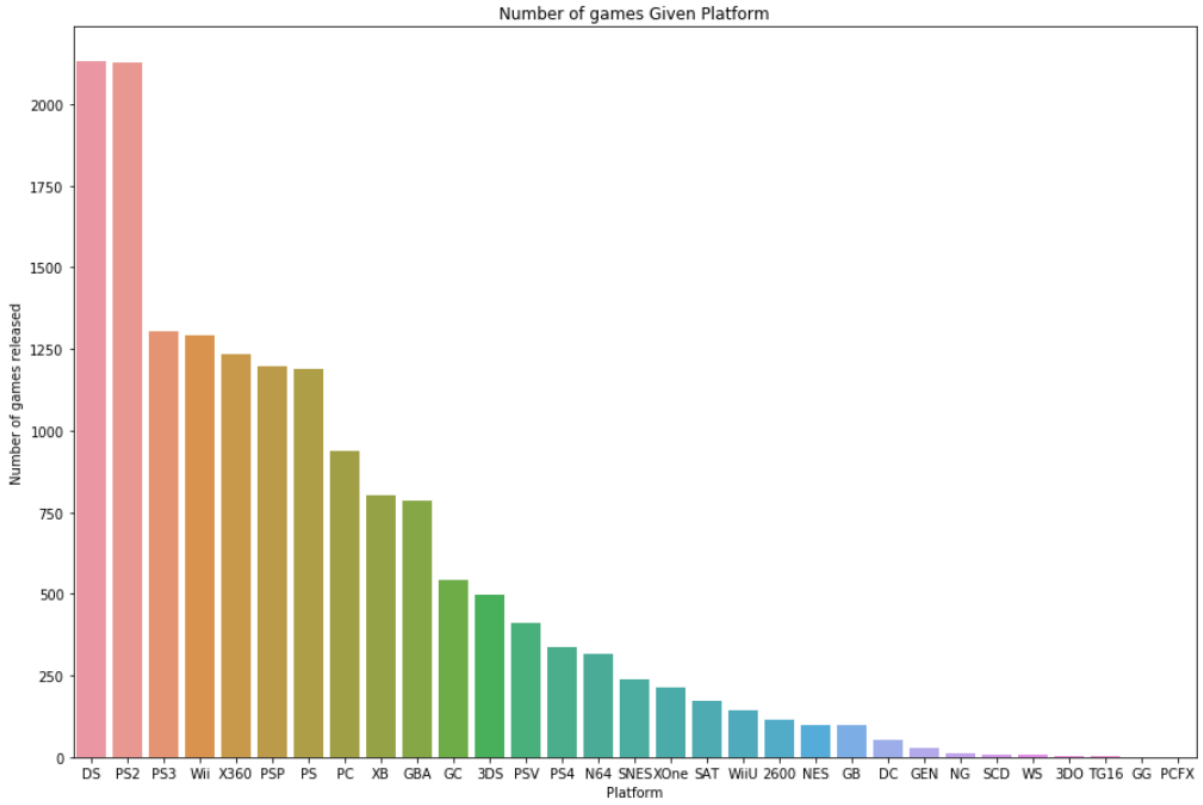
Below are few of the plots from EDA: -




Sales per year by Genre
Action
Adventure
Fighting
Misc
Platform
Puzzle
Racing
Role-Playing
Shooter
Simulation
Sports
Strategy
27.77 Action
1999
100
50
0
1980
1990
2000
2010
2020


Publisher Games Releases By Years
Interactive
THQ
tertainment
Sega
Nintendo
ndai Games
tertainment
ctronic Arts
Activision
1980
1985
1990
1995
2000
2005
2010
2015
2020
120
100
80
60
40
20
Export to plot.ly »


Genre Distribution along the years
Year
2020
2015
2010
2005
2000
1995
1990
1985
1980
Sports
Platform
Racing
Role-Playing
Puzzle
Misc
Shooter
Simulation
Action
Fighting
Adventure
Strategy
Genre


Number of games Given Platform
Number of games released
2000
1750
1500
1250
1000
750
500
250
0
DS
PS2
PS3
Wii
X360
PSP
PS
PC
XB
GBA
GC
3DS
PSV
PS4
N64
SNES
XOne
SAT
WiiU
2600
NES
GB
DC
GEN
NG
SCD
WS
3DO
TG16
GG
PCFX
Platform

Outliers Maintenance

From, the output above, we can see: -

- There are high outliers in sales columns (NA, EU, JP, Other, Global) and User_Count column.

They might be useful for training as they indicate bestseller games, but for now I am going to remove them and maybe add them later. The below function can be used to remove outliers present in the data set. A data entry is called an outlier if: -

- value < Q1 - 3 * IQR
- value > Q3 + 3 * IQR

where,

- Q1 - First Quartile
- Q3 - Thrid Quartile
- IQR - Inter-quartile range

```
def rm_outliers(df, list_of_keys):
    df_out = df
    for key in list_of_keys:

        # Calculate first and third quartile
        first_quartile = df_out[key].describe()["25%"]
        third_quartile = df_out[key].describe()["75%"]

        # Interquartile range
        iqr = third_quartile - first_quartile
        removed = df_out[(df_out[key] <= (first_quartile - 3 * iqr)) |
                    (df_out[key] >= (third_quartile + 3 * iqr))]
        df_out = df_out[(df_out[key] > (first_quartile - 3 * iqr)) &
                    (df_out[key] < (third_quartile + 3 * iqr))]

    return df_out, removed
```

Data after removing outliers is given below: -

```
data, rmvd_global = rm_outliers(data, ["Global"])
data.describe()
```

	Year	NA	EU	JP	Other	Global	Critic_Score	Critic_Count	User_Score	User_Count	A
count	15401.000000	15401.000000	15401.000000	15401.000000	15401.000000	15401.000000	7286.000000	7286.000000	6747.000000	6747.000000	15401.0000
mean	2006.592624	0.144688	0.072628	0.047301	0.024357	0.289258	67.779028	24.518117	7.079976	111.325033	11.4073
std	5.758078	0.210709	0.131408	0.130786	0.050152	0.346918	13.612120	17.194878	1.511031	406.635191	5.7580
min	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000	13.000000	3.000000	0.000000	4.000000	-2.0000
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000	59.000000	11.000000	6.300000	9.000000	8.0000
50%	2007.000000	0.070000	0.020000	0.000000	0.010000	0.150000	70.000000	20.000000	7.400000	21.000000	11.0000
75%	2010.000000	0.190000	0.080000	0.030000	0.030000	0.380000	78.000000	34.000000	8.200000	61.000000	15.0000
max	2020.000000	1.670000	1.580000	1.650000	1.180000	1.690000	98.000000	106.000000	9.700000	10665.000000	38.0000

From, the above we can see that nearly half of the entries in the User score are not present. For this purpose, I have made two models. The first one does not have scored data, while the second one takes the weighted score coming the User score and the critic score. The first model follows One-hot encoding scheme, while the second one uses Ordinal Encoding scheme. In both the models, the initial procedure is the same.

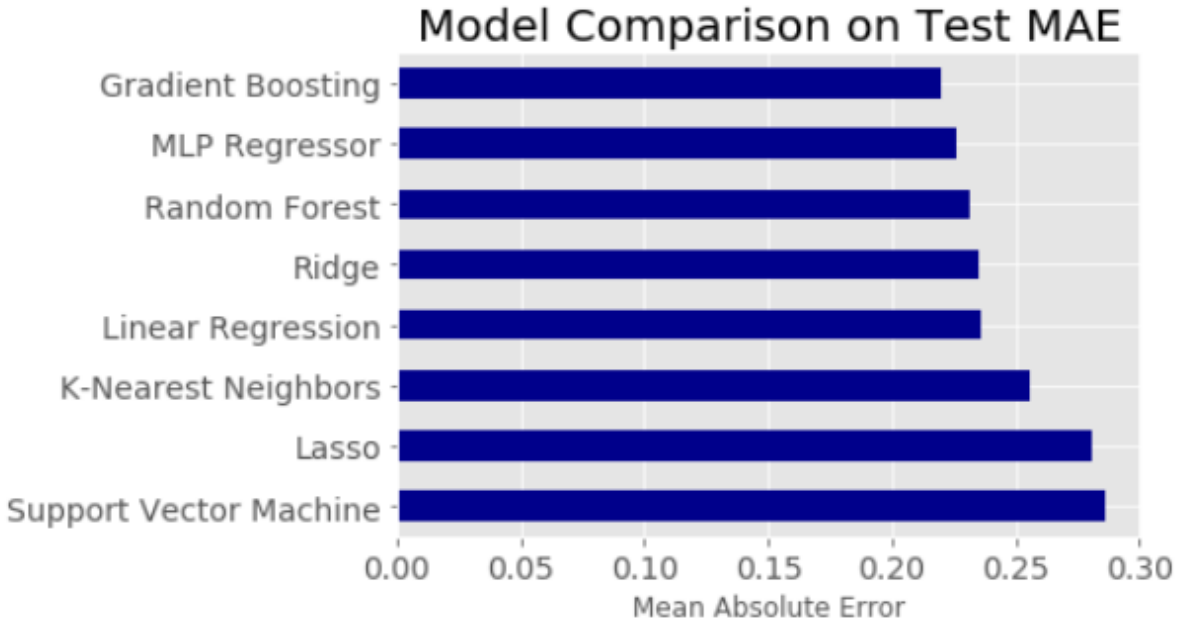
Model 1: -

Encoding

```
import category_encoders as ce

# Numeric columns
numeric_subset = scored.select_dtypes("number").drop(columns=["NA", "EU", "JP", "Other", "Year"])

# Categorical column
categorical_subset = scored[["Platform", "Genre", "Rating"]]

# One hot encoding
encoder = ce.one_hot.OneHotEncoder()
categorical_subset = encoder.fit_transform(categorical_subset)

# Column binding to the previos numeric dataset
features = pd.concat([numeric_subset, categorical_subset], axis = 1)

# Find correlations with the score
correlations = features.corr()["Global"].dropna().sort_values()
```

Prediction Models

The data was split into training (80%) and testing (20%) set. For each of the machine learning models listed above we calculated the mean absolute error and the respective results are shown below: -



From the above, it can be seen that Gradient Boosting is the most appropriate model for the dataset out of the 8 models chosen. The error obtained was 0.2197. We then fine tune the hyper parameters to further try to decrease the error.

Hyperparameters tuning

The selected parameters for the tuning are shown below: -

```
hyperparameter_grid = {"loss": ["ls", "lad", "huber"],
                       "max_depth": [2, 3, 5, 10, 15],
                       "min_samples_leaf": [1, 2, 4, 6, 8],
                       "min_samples_split": [2, 4, 6, 10],
                       "max_features": ["auto", "sqrt", "log2", None]}
```

The results of hyperparameter by using the randomized search is shown below: -

```
random_cv.best_estimator_
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='huber', max_depth=15,
             max_features='log2', max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=8, min_samples_split=6,
             min_weight_fraction_leaf=0.0, n_estimators=100,
             n_iter_no_change=None, presort='auto', random_state=42,
             subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0,
             warm_start=False)
```

After which we use grid search to find optimal value of the n_estimators parameter.

```
grid_search.best_estimator_
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='huber', max_depth=15,
             max_features='log2', max_leaf_nodes=None,
             min_impurity_decrease=0.0, min_impurity_split=None,
             min_samples_leaf=8, min_samples_split=6,
             min_weight_fraction_leaf=0.0, n_estimators=50,
             n_iter_no_change=None, presort='auto', random_state=42,
             subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0,
             warm_start=False)
```

Model Results

Finally the errror value is calculated for the updated model.

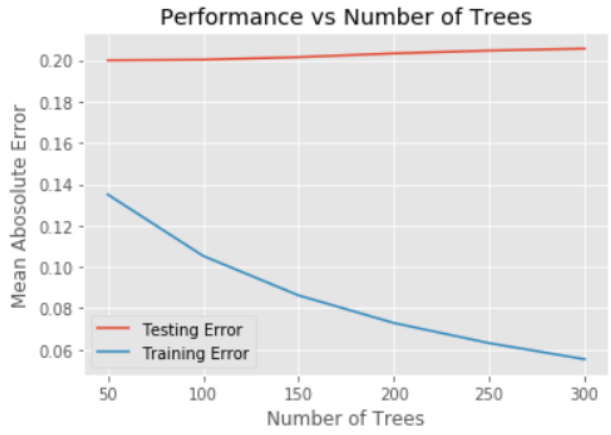
```
basic_final_model = grid_search.best_estimator_
basic_final_pred = basic_final_model.predict(X_test)
basic_final_mae = mae(Y_test, basic_final_pred)
print("Final model performance on the test set: MAE = {:.04f}.".format(basic_final_mae))
```

```
Final model performance on the test set: MAE = 0.2095.
```

The reason for such low error drop is because the data has been overfit as shown in the graph below: -

```
results = pd.DataFrame(grid_search.cv_results_)

plt.plot(results["param_n_estimators"], -1 * results["mean_test_score"], label = "Testing Error")
plt.plot(results["param_n_estimators"], -1 * results["mean_train_score"], label = "Training Error")
plt.xlabel("Number of Trees"); plt.ylabel("Mean Abosolute Error"); plt.legend();
plt.title("Performance vs Number of Trees");
```

executed in 174ms, finished 14:29:15 2019-04-07



This is the end of Model 1. Further attempts can be made to reduce the error and optimize this model in near future.

Model 2:

Weighted Scoring

There are new columns that are added to the Dataset named as “Grouped Platform”. This attributes group all the famous platforms together into a single entity. For example, Playstation entity for all PS, PS2, PSP, PS3, etc. Next, I create my own Weighted score Rating and Developer score rating. To do this, we find percent of all games created by each developer, then calculate cumulative percent starting with devs with the least number of games. Finally, I divide them into 10 groups (10% each). Higher rank means more games developed. The formula that is used for calculating Weighted score is ((User score * 10 * User count) + (Critic score * Crictic count))/ (User count + Critic count). This is done for all the entries in the dataset.

```
# One weighted score value including all scores and counts field.
scored["Weighted_Score"] = (scored["User_Score"] * 10 * scored["User_Count"] +
                            scored["Critic_Score"] * scored["Critic_Count"]) / (scored["User_Count"] + scored["Critic_Count"])

# Dataframe having developers arranged based on their frequency
devs = pd.DataFrame({"dev": scored["Developer"].value_counts().index,
                     "count": scored["Developer"].value_counts().values})

# Mean scoring datafram based on the weighted score
m_score = pd.DataFrame({"dev": scored.groupby("Developer")["Weighted_Score"].mean().index,
                        "mean_score": scored.groupby("Developer")["Weighted_Score"].mean().values})

# Creating merging the mean_score and developer dataframes and then sorting the resultant into ascending order
devs = pd.merge(devs, m_score, on="dev")
devs = devs.sort_values(by="count", ascending=True)

# Percentage of all games created by each developer and storing it in form of cumulative fashion
devs["percent"] = devs["count"] / devs["count"].sum()
devs["top%"] = devs["percent"].cumsum() * 100

# Dividing them into 10 groups
n_groups = 10
devs["top_group"] = (devs["top%"] * n_groups) // 100 + 1
devs["top_group"].iloc[-1] = n_groups
devs
```

A general check can be performed before we move on to the encoding of the data, as shown below: -

```
data["Critic_Score"].fillna(0.0, inplace=True)
data["Critic_Count"].fillna(0.0, inplace=True)
data["User_Score"].fillna(0.0, inplace=True)
data["User_Count"].fillna(0.0, inplace=True)
data = data.join(devs.set_index("dev")["top_group"], on="Developer")
data = data.rename(columns={"top_group": "Developer_Rank"})
data["Developer_Rank"].fillna(0.0, inplace=True)
data["Rating"].fillna("None", inplace=True)
```

executed in 22ms, finished 14:29:16 2019-04-07

Enconding

Now I will do the same things as I did in the basic model, except for using Ordinal encoding for categorical values instead of OneHot.

```
# Select the numeric columns
numeric_subset = data.select_dtypes("number").drop(columns=["NA", "EU", "JP", "Other", "Year"])

# Select the categorical columns
categorical_subset = data[["Grouped_Platform", "Genre", "Rating"]]

mapping = []
for cat in categorical_subset.columns:
    tmp = scored.groupby(cat).median()["Weighted_Score"]
    mapping.append({"col": cat, "mapping": [x for x in np.argsort(tmp).items()]})

encoder = ce.ordinal.OrdinalEncoder()
categorical_subset = encoder.fit_transform(categorical_subset, mapping=mapping)

# Join the two dataframes using concat. Axis = 1 -> Column bind
features = pd.concat([numeric_subset, categorical_subset], axis = 1)

# Find correlations with the score
correlations = features.corr()["Global"].dropna().sort_values()
```

Next I follow the same procedure after Enconding in the basic model and fit various features in the dataset and then finally calculated the updated error value based on the parameters.

Model Results

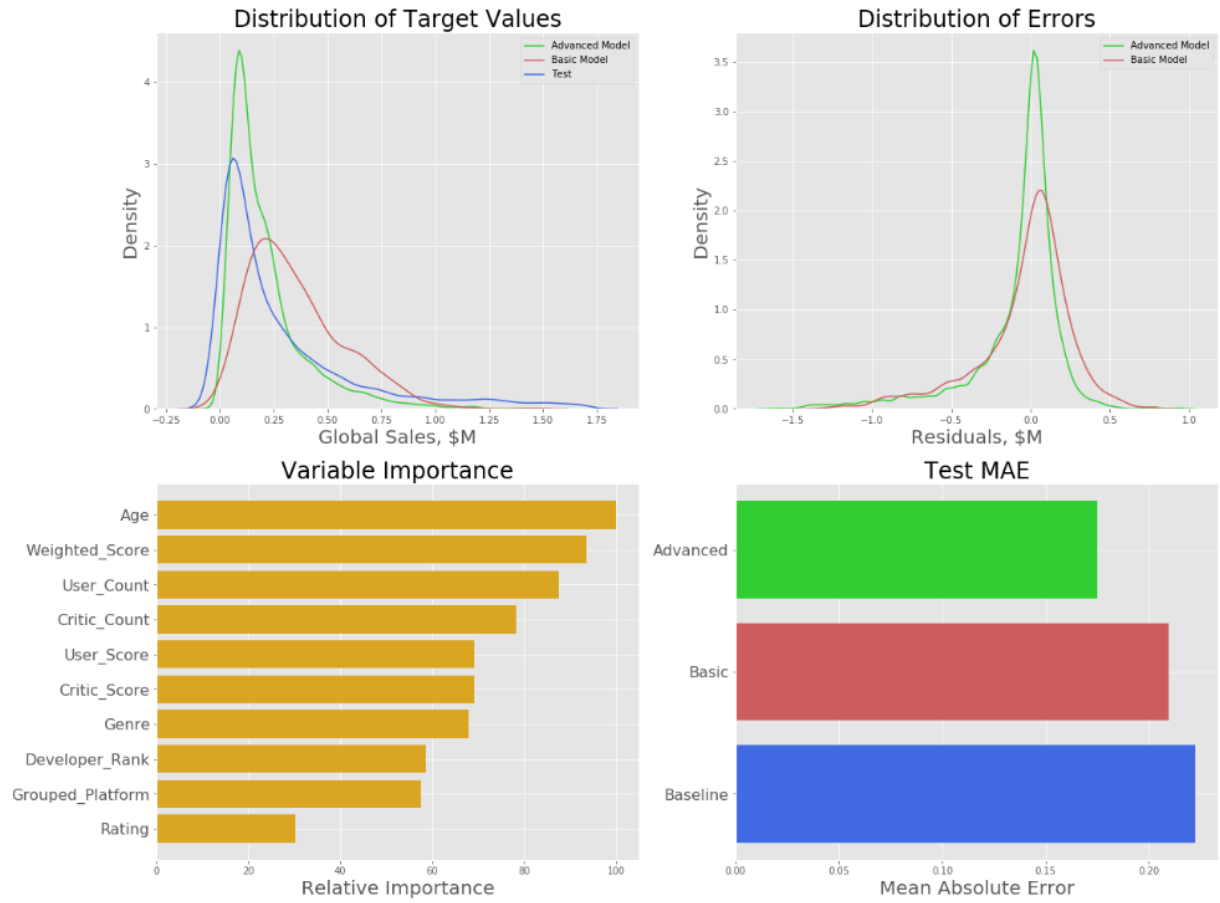
```
# Getting the final model error
final_model = grid_search.best_estimator_
final_pred = final_model.predict(features_test)
final_mae = mae(target_test, final_pred)
print("Final model performance on the test set: MAE = {:.04f}.".format(final_mae))
```

```
Final model performance on the test set: MAE = 0.1757.
```

The final updated error value is 0.1757, which much lesser than the initial model. Hence this model is much better than the last model and is much preffered over the other model.

6. RESULTS

The below plot best summarizes how the models are different from each other, along with which parameters to use to get the best results while training the model.



There are many conclusions that can be drawn from EDA. Few of them are listed below: -

- Wii sports have the most sales so far
- EU had max sales in 2009
- Since 1996 to 2013 NA was leading in sales
- Activison was one of the oldest publishers but recently EA and Nintendo and Namo Bandai games have released most of the games
- 70 was the critic score most of the times (276 games) while the highest score was 98

The most important features that affect the model are Age, Weighted Score, User Count, Critic Count, User Score and Critic Score.

The best machine learning model suitable for the data is Gradient Boosting model and the minimum error achieved was 0.1757.

7. CONCLUSION AND FUTURE WORK

We were successfully able to perform the exploratory data analysis on the dataset and gain valuable inferences from the data. From that were able to further encode and then train different machine learning algorithms and create different models on the dataset. We were then able to reduce the error respectively and get a final accuracy of 0.1757. Since, Model 1 was overfitting the data, future work main involve to work on that model and remove the overfitting condition. Model 2 can be further more refined to reduce the error. Introduction of new machine learning algorithms other than the mentioned ones, might be able to give better results. These all can be considered as future work.

8. REFERENCES

- Kaggle (https://www.kaggle.com/datasets)
- Wikipedia (https://en.wikipedia.org/wiki/Video_game)
- NOMIKOS, Petros M. (ed.). 2016 IEEE Conference on Computational Intelligence and Games. 2016 IEEE Conference on Computational Intelligence and Games. 2016.
- SIWEK, STEPHEN E. Video Games in the 21st Century: The 2017 Report [online]. 2017 [visited on 2017-04-12]. Available from: http : / / www . theesa . com / wp - content / uploads / 2017 / 02 / ESA _ EconomicImpactReport_Design_V3.pdf.
- MCCOURT, J. Year-end DEG Home Entertainment Spending [online]. 2016 [visited on 2017-04-12]. Available from: http://degonline. org/wp-content/uploads/2016/01/External_2015- Year- endDEG-Home-Entertainment-Spending-1-5-2016.pdf.
- TAMASSIA, Marco; RAFFEY, William; SIFAZ, Rafet; DRACHENX, Anders; ZAMBETTA, Fabio; HITCHENS, Michael. Predicting Player Churn in Destiny: A Hidden Markov Models Approach to Predicting Player Departure in a Major Online Game. In: NOMIKOS, Petros M. (ed.). 2016 IEEE Conference on Computational Intelligence and Games. 2016, p. 325.
- SHAKER, Noor; ABOU-ZLEIKHA, Mohamed. Transfer Learning for Cross-Game Prediction of Player Experience. In: NOMIKOS, Petros M. (ed.). 2016 IEEE Conference on Computational Intelligence and Games. 2016, p. 209.

****