CCT COLLEGE

Machine Learning

Assessment

Dublin

2021

GROUP :

Adrian de Miranda Gomes - 2021261
Gabriela Fernandez Perez - 2021305
Nuno Alfredo Ribeiro Teixeira de Almeida - 2021310

VIDEO GAMES SALES ANALYSIS

Work developed in the analysis of a Dataset and use of the models to obtain the first general grade of the Machine Learning discipline of tHDip Data Analytics - FT - Sept 2021 at CCT course.

Lectures:Dr. Muhammad Iqbal

Dublin

2021

ILLUSTRATION LIST

APPENDIX LIST

SUMMARY

1. INTRODUCTION

The research is based on a global analysis of video game sales on the major platforms in the market, which was collected directly from the Kaggle platform. The research was prompted by projections for the video game business in 2020, which has been one of the fastest growing industries in recent years and is predicted to expand even faster in the future. It is estimated that the global games market could be worth $200 billion by the end of 2023, although the market has declined slightly in 2020 (New Zoo site).

According to the analysis carried out in the work and the research already published, it is believed that there may be more than 30,000 games available for download in the Steam Games Library, the largest and most popular platform for digital game distribution worldwide. (PC Gamer site)

Since 2018, the global games market has always been highly rated, over US$ 100 billion, depending on the segment. In March 2021, with 35.39 million copies sold. Animal Crossing New Horizons is in second place with 32.63 million copies sold. (Statista site).

The work to be developed will lead us to solve the objectives and explain the metrics to achieve them, according to the Machine Learning Models. The purpose of this analysis is to find and train a model to predict what the European sales of a potential sale would have been based on other variables in this study.

2. METHODOLOGY

The aim of the project is to present an exploratory analysis of video game sales, which were best sold in Japan, Europe, America and on a global scale. As well as what are the best-selling games on the main platforms. Sales in Europe were determined as the most important variable analysed in the work.

Its main rationale is that the games market has stood out in the world economic scenario, in addition to being very important in the technological environment and artificial intelligence. The approach an algorithm learns to become more accurate in its predictions is how traditional machine learning is often classified. supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning are the four basic techniques. The algorithm that data scientists use is determined by the sort of data they wish to predict.

According to Burns (2021) when a machine learning model is complex, explaining how it works might be difficult. Because it's crucial for the business to explain how each choice was taken, data scientists in some vertical businesses must employ fundamental machine learning models.

To perform steps and develop the work, we used academic knowledge from the Data Preparation, Statistical Techniques, and Machine Learning disciplines. We also used external web research and libraries available in Python language to have illustrations and perform functions in the Data Set used. As a result, a Jupyter Notebook file was created, as well as an external infographic to represent each participant's performance and a Report that clarifies the entire process and analysis.

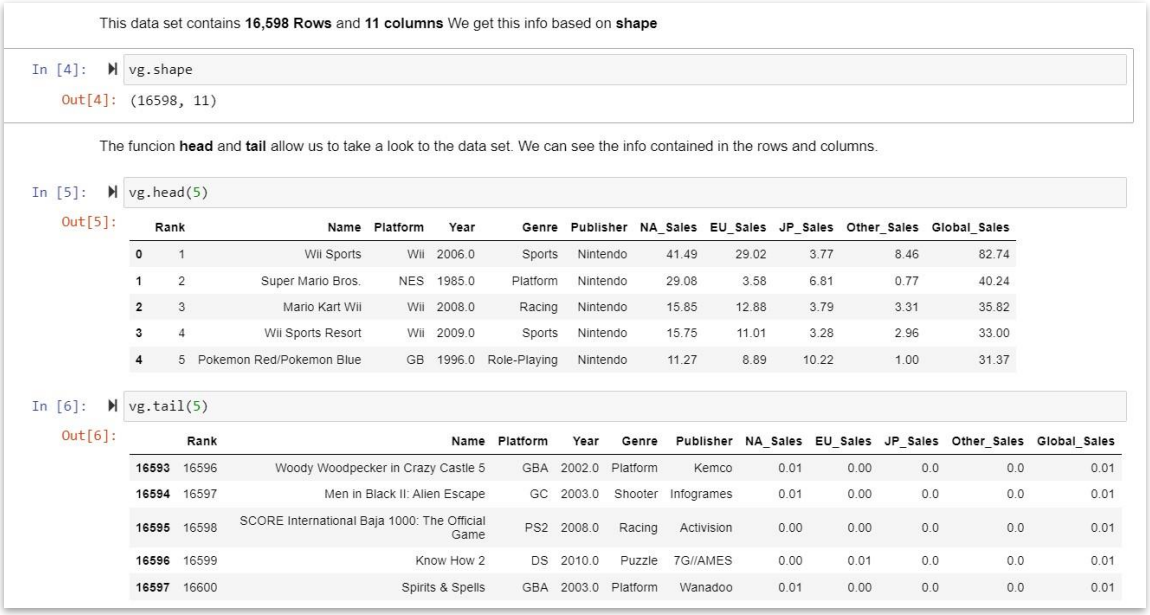
The methodology we used in this dataset includes different approaches, including an EDA (Exploratory Data Analysis) where combining different code functions (explained below) we determined the dataset, dealing with missing values, or zero values, etc.

3. RESULTS

3.1 DATA UNDERSTANDING

According to the descriptive analysis of the dataset in general, it contains 16,598 rows and 11 columns. This is characterized by the columns: Rank, Name, Platform, Year, Genre, Publisher, North American Sales, European Sales, Japan Sales, Other Sales Locations, and Global Sales.

Figure 1 – Size and details about Data Set;



This data set contains **16,598 Rows** and **11 columns** We get this info based on **shape**

```
In [4]: vg.shape
Out[4]: (16598, 11)
```

The funcion **head** and **tail** allow us to take a look to the data set. We can see the info contained in the rows and columns.

```
In [5]: vg.head(5)
```

Out[5]:

	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

```
In [6]: vg.tail(5)
```

Out[6]:

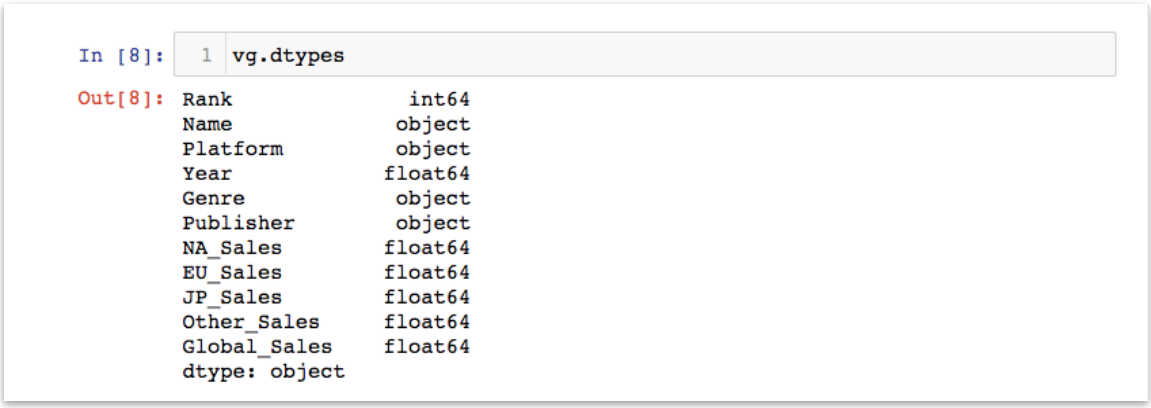
	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.0	0.0	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.0	0.0	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.0	0.0	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.0	0.0	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.0	0.0	0.01

Figure 2 – Information about Data Set;

```
In [4]: 1 vg.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.4+ MB
```

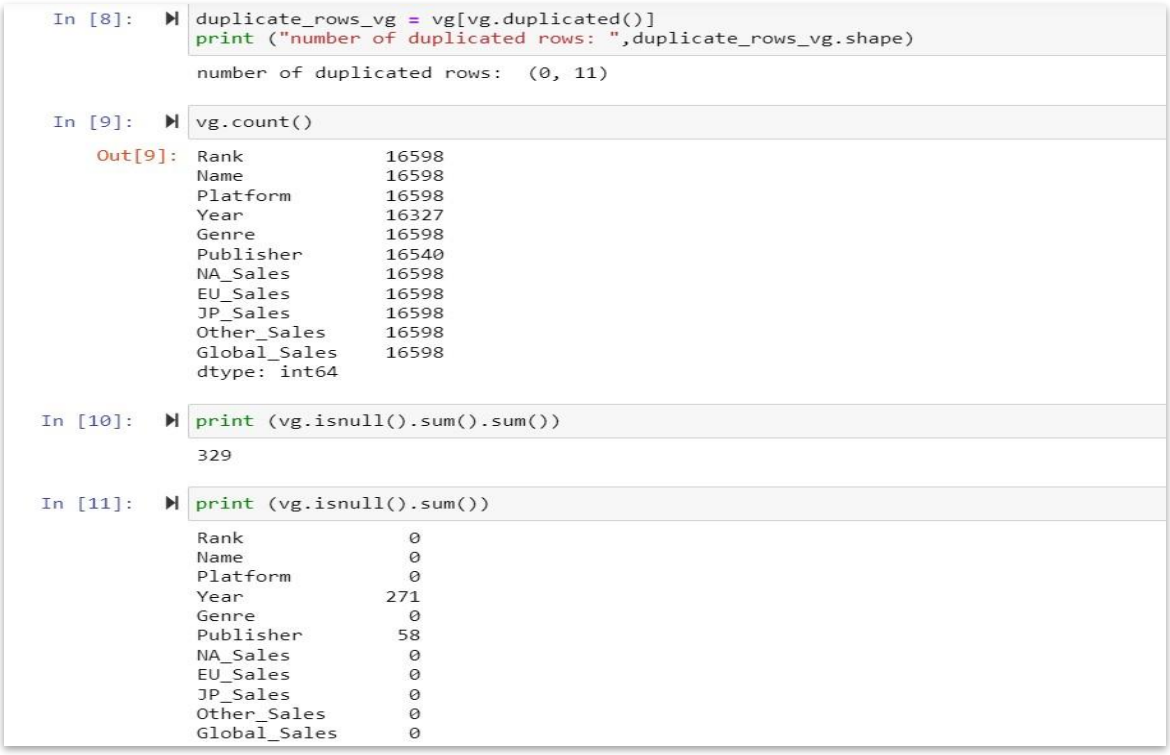
This dataset consists on different types of data, such as categorical (objects/text), numbers (integer/int) and float (decimal numbers). Further, the objects will need to be encoded as Machine Learning models cannot work with text.

Figure 3 – Using df.dtypes code;



Once we have an overview, we prepare and clean the entire dataset. We looked for duplicate rows, which this dataset does not have, but found missing values in the Year (271) and Publisher (58) variables. As the Pandas library was used, which allows automatically returning all cells with blank or Na values as NaN (Sullivan, 2018), but in the codename "Count" it can indicate if there is any missing value, since this function shows how many attributes there are in each observation. (Pandas, 2018).

Figure 4 – Information and Missing Data Variables;



Since we knew Publisher was not an interesting variable for our project, we decided to remove it and replace all missing Year values with the median. Once we deal with missing values, we find zero values in some variables, which include the most important one for our case of study (European Sales: 5712). In this case is essential to standardize our data in order to perform a good performance in the analysis.

Figure 5 – Information about Data Set;

```
In [13]:
vg.dropna(subset=["Publisher"],inplace=True)
vg["Year"].fillna(vg["Year"].median(),inplace=True)
vg["Year"]=vg["Year"].astype("int64")

In [14]:
print (vg.isnull().sum())

Rank            0
Name            0
Platform        0
Year            0
Genre           0
Publisher       0
NA_Sales        0
EU_Sales        0
JP_Sales        0
Other_Sales     0
Global_Sales    0
dtype: int64
```

Figure 6 – Numbers of Zero Values per variable;

```
In [15]:
print ("Number of rows with 0 values for each variable")
for col in vg.columns:
    missing_rows = vg.loc[vg[col] == 0].shape[0]
    print (col + ": " + str(missing_rows))

Number of rows with 0 values for each variable
Rank: 0
Name: 0
Platform: 0
Year: 0
Genre: 0
Publisher: 0
NA_Sales: 4476
EU_Sales: 5712
JP_Sales: 10411
Other_Sales: 6433
Global_Sales: 0
```

Once our data was prepared, we perform a small EDA (Exploratory Data Analysis). In this stage we look for the list of publisher and genres that s the most frequently-occurring element. To have a better understanding of the dataset, we look for the 5 number summary which consist in the Smallest Data Value, the First Quartile, Median, Third Quartile and the Largest Value (Britannica, Gregersen, 2021) of the dataset, we also wanted to know this information with the objects. It is important to know these values because they represent a very specific part of the data set, e.g. the mean identifies the centre of the data set (how it is located), the quartiles give an idea of how the data is spread and by the max and min we can get the range of observations. (Government of Canada, 2002)

Figure 7 – The most frequently-occurring element with Value_Counts ;

```
In [18]: vg["Publisher"].value_counts()

Out[18]: Electronic Arts                1351
         Activision                      975
         Namco Bandai Games              932
         Ubisoft                         921
         Konami Digital Entertainment    832
                                         ...
         Kando Games                       1
         White Park Bay Software           1
         2D Boy                            1
         FunSoft                           1
         UEP Systems                       1
         Name: Publisher, Length: 578, dtype: int64

In [20]: vg["Genre"].value_counts()

Out[20]: Action          3309
         Sports          2343
         Misc            1712
         Role-Playing    1486
         Shooter         1308
         Adventure       1282
         Racing          1248
         Platform         884
         Simulation       863
         Fighting         846
         Strategy         678
         Puzzle           581
         Name: Genre, dtype: int64
```

Figure 8 – Describe function with all variables;

```
In [21]: vg.describe()
```

Out[21]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16540.000000	16540.000000	16540.000000	16540.000000	16540.000000	16540.000000	16540.000000
mean	8294.197642	2006.414510	0.265079	0.146883	0.077998	0.048191	0.538426
std	4790.703200	5.788794	0.817929	0.506129	0.309800	0.188879	1.557424
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4143.750000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8292.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12440.250000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.480000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

```
In [22]: vg.describe(include=object)
```

Out[22]:

	Name	Platform	Genre	Publisher
count	16540	16540	16540	16540
unique	11442	31	12	578
top	Need for Speed: Most Wanted	PS2	Action	Electronic Arts
freq	12	2159	3309	1351

The most popular Video Game in the whole dataset is Need for Speed: Most Wanted. PSP2 (Figure 11). is the most popular Platform and the most popular Genre is Action (Figure 10). We could get this information based on the mode function which

determine the most repeated value, whereas the mean is the average element in the data set. (Stapel, 2021)

Figure 9 – Using Mode Code for all variables;

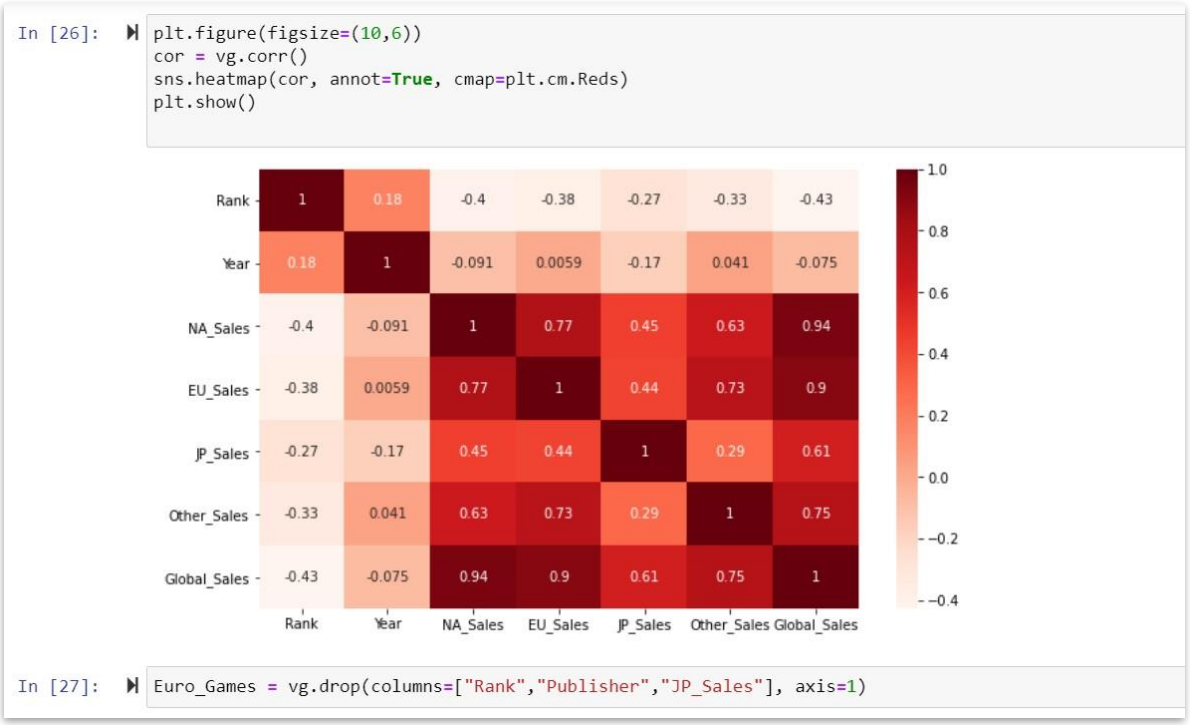
```
In [142]:
name_mode =vg["Name"].mode()
print ("- Most Popular Video Game: ",name_mode)
print()
Pmode =vg["Platform"].mode()
print ("- Most Popular Platform: ",Pmode)
print()
Yearmode =vg["Year"].mode()
print ("- Year: ",Yearmode)
print()
Genremode =vg["Genre"].mode()
print ("- Most Popular Genre: ",Genremode)
print()
Pumode =vg["Publisher"].mode()
print ("- Most Popular Publisher: ",Pumode)
print()

- Most Popular Video Game:  0    Need for Speed: Most Wanted
dtype: object

- Most Popular Platform:  0    PS2
dtype: object

- Year:  0    2007
dtype: int64

- Most Popular Genre:  0    Action
dtype: object

- Most Popular Publisher:  0    Electronic Arts
dtype: object
```

Figure 10 – Most Popular Video Game;

3.2 VISUALISATION

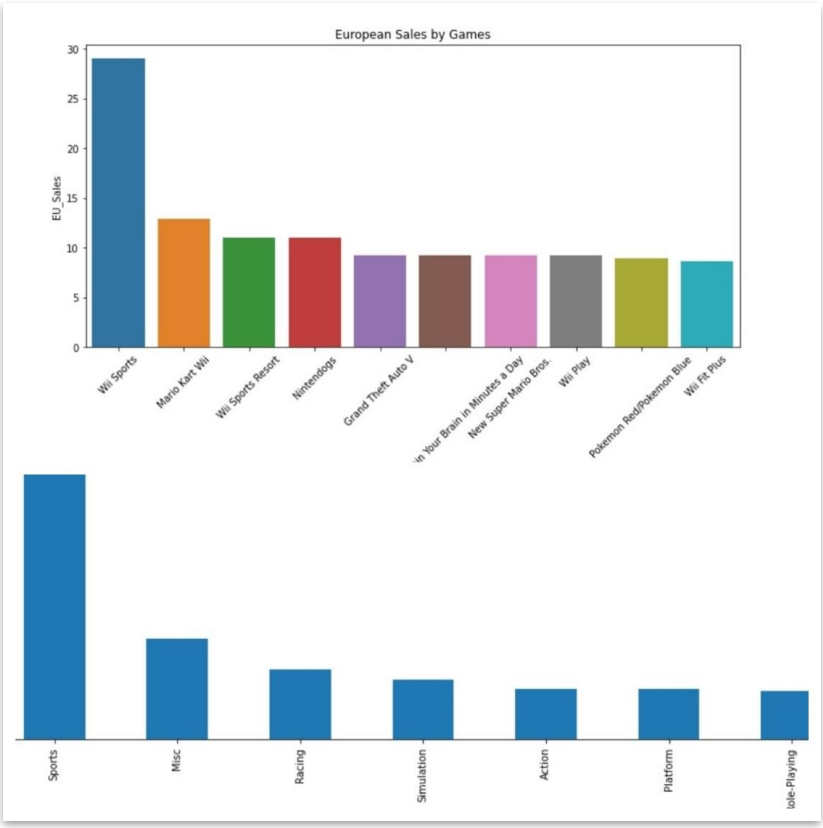
Subsequently, data preparation for the models is essential. With all this, we had enough information to determine which variables will not be used in our analysis. To determine the variables, we considered the correlation graph, and it concluded that according to the percentage of correlation between, e.g., Euro sales, North America sales and Other sales around 70%. Indicating that the variables "Rank", "Publisher", "JP Sales" won't fit to work in the model.

Figure 11 – Correlation Variables;



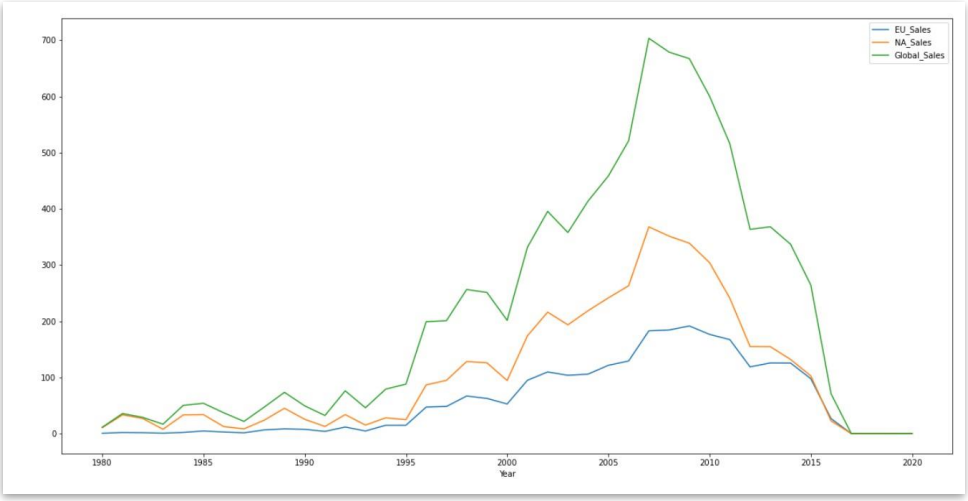
A graph was generated describing the 10 best games by the "Name" column listed in the dataset. This ranking brings a reference between the Name column and Sales from Europe. This indicates that 'Wii Sports' and 'Mario Kart Wii' were the top sellers. In another case, we use a plot. bar to show the correlation between "Genre" and "EU_Sales", and understand which Genre is more frequent in European sales.

Figure 12– Most Sale Game and Genre by European Sales;



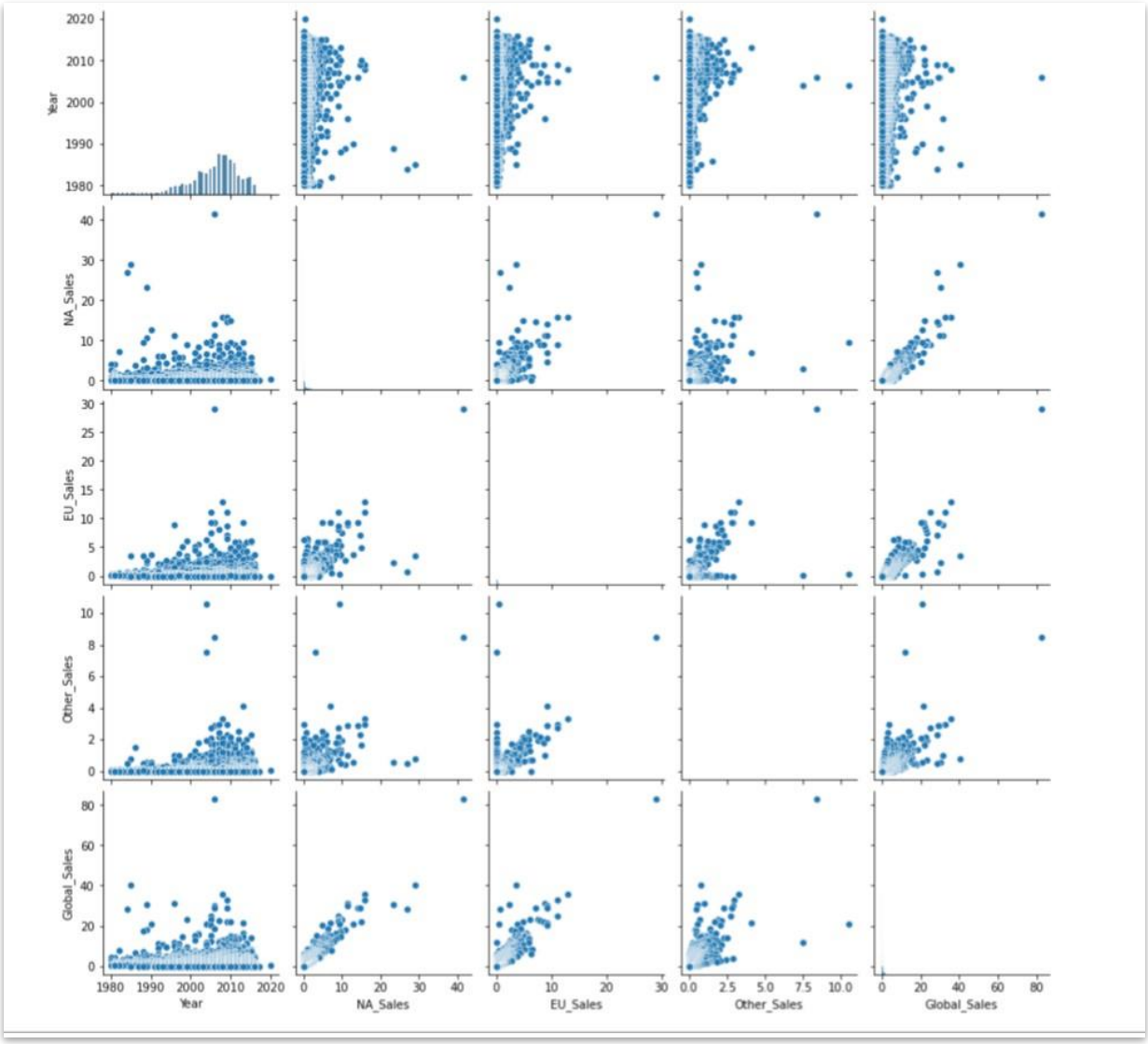
The chart below shows the years that were considered in the analysis, in which the period between 2000 and 2014 was the one with the highest sales results for the Games in Europe. However, during the same period, it appears that those in NA sales were even higher than when compared to Euro sales. European sales peaked around 2009. In light of global sales, it is noted that there has been a growing increase from 1995 to 2010.

Figure 13– European, Global and North American Sales from 1980 to 2020;



The following chart was created to show the marginal distribution of data in each column, graphically representing different variables in the rows and columns. According to the pattern, a grid of axes will be created so that each numeric variable in the data is shared between the y axes in a single row and the x axes in a single column. So that we were able to identify the sales of each region about the year and among others.

Figure 14–Scatterplot;



4. DISCUSSION

4.1 MODELING

According to the objective of the project which is to determine which variables affect Sales in Europe, how it behaves and the forecast of Sales in Europe about the Year, Platform, Gender, American sales and other sales. For this, we explore some models of machine learning. Beforehand, we need to understand what the machine learning model means.

Machine learning is a field of Artificial Intelligence (AI) based on the premise that computers can learn from data, recognize patterns and make judgments with very little human intervention (SAS Institute, 2021). Models can be used for classification, which means they can predict labels, while regression predicts quantity (Brownlee, 2017).

Machine Learning is divided into *Supervised learning, Semi-supervised and Unsupervised Learning.*

Supervised Learning: The algorithm is trained using a label, this means that the machine learns by example. Here the trainer knows the desire output and then the algorithm needs to find the method to determine how to find this “correct answer”. (APD, 2019).

The algorithm is trained until it achieves a good accuracy in the following things: Identifying the patron of the data; Learning from observations; Making predictions.

Semi-Supervised Learning: The algorithm is trained by a mix of labelled and unlabelled data.

Unsupervised Learning: This method does not need a label. The “correct answer” is not given, tries to discover the dataset internal structure, by similarities it organizes the data and it finally compress it.

The good model cannot suffer from *Underfitting* or *Overfitting*. When a model is Overfitting, it is simply when the training data gives an excellent result but does not perform well on the test data. It turns out that it is possible that the model learned the existing training relationships very well and then only memorized and when receiving the information from the predictor variables in the test data, the model tries to apply the same memorized rules, but with different data, this rule has no validity and performance is affected. It is common to hear that in this scenario the trained model has no generalization. When the model is *Underfitting*, the performance in the training itself is already considered poor, so this model can already be disregarded for the analysis. (McQuaid, 2021)

The models used in the research were **Linear Regression, KNN, SVR and SVM**.

- **Linear Regression**

Linear regression has examined a set of predictor variables that perform well in predicting a dependent variable. That is, regression estimates are used to explain the relationship between a dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent variable and one

independent variable is defined by the formula y = c + b * x, where y = estimated score of the dependent variable, c = constant, b = regression coefficient and x = score in the independent variable. (Müller, Guido, 2017)

- **KNN (K-Nearest Neighbours)**

The K-Nearest Neighbours (KNN) algorithm is a type of supervised machine learning algorithm. KNN is considered easy to implement in its most basic form, and still performs quite complex classification tasks. It is a lazy learning algorithm as it does not have a specialized training phase. Instead, it uses all the data for training while classifying a new data point or instance. KNN is a non-parametric learning algorithm, meaning that it assumes nothing about the underlying data. This is an extremely useful feature since most real-world data doesn't follow any theoretical assumptions. There is makes the KNN algorithm much faster than others that require training, eg SVM, linear regression, etc. The KNN algorithm does not work well with large dimensional data and categorical resources, as with a large number of dimensions, it is difficult for the algorithm to calculate the distance in each dimension. (Müller, Guido, 2017)

- **SVM**

This method increases the dimensionality of the resource space. This model works with more than one dimension that is segregated by a line and supports two more lines that are called Supported Vectors that can give rise to classification errors, but this margin can be controlled to obtain better precision. (Iqbal, 2019). This misclassification reduces the overfitting problem as it allows some observations to be on the wrong side of the hyperplane (Amat, 2017).

- **SVR**

This method is a little bit different from SVM, as it is a regression model that works for continuous values instead of Classification as SVM. It works with the same parameters as SVM. (Bhattacharyya, 2018)

Once we understood what is Machine Learning and what does each Model specifically does, we can start explaining the code and the results we got with it. The first step was importing the libraries to perform all the models, we also remove variables that are not useful to answer our question, such as Name of videogame, Global or other Sales and rename this data as *Model Games*. The second step was very important, encode de data, this is because two of our variables where objects, and machines cannot work with text, we use *Label Encoder* to encode our variables and be able to perform our models.

Figure 15– Dropping and Encoder Dataset;

```
In [42]: Modelgames = Euro_Games.drop(columns=["Name","EU_Sales","Global_Sales"], axis=1)

In [43]: Modelgames.head()
```

Out[43]:

	Platform	Year	Genre	NA_Sales	Other_Sales
0	Wii	2006	Sports	41.49	8.46
1	NES	1985	Platform	29.08	0.77
2	Wii	2008	Racing	15.85	3.31
3	Wii	2009	Sports	15.75	2.96
4	GB	1996	Role-Playing	11.27	1.00

```
In [44]: le = LabelEncoder()
Modelgames['Platform'] = le.fit_transform(Modelgames['Platform'].astype('str'))
Modelgames['Genre'] = le.fit_transform(Modelgames['Genre'].astype('str'))
Modelgames['Year'] = le.fit_transform(Modelgames['Year'].astype('str'))
Modelgames.head()
```

Out[44]:

	Platform	Year	Genre	NA_Sales	Other_Sales
0	26	26	10	41.49	8.46
1	11	5	4	29.08	0.77
2	26	28	6	15.85	3.31
3	26	29	10	15.75	2.96
4	5	16	7	11.27	1.00

Preparing data for the model includes Scaler to standardize the data. We determine our dependent variable (y) which always depends on the question we want to answer. Since we want to know what affects video game sales in Europe, we selected EU Sales as the dependent variable. And we just have to determine that X and y will be used for training and testing.

Figure 16– Using Scaler and performing Variables X and Y;

Performing our independente variable Y we will use our dependent variable "European Sales"

```
In [45]: y = Euro_Games['EU_Sales']

In [46]: X= Modelgames

In [47]: scaler = StandardScaler()
scaler.fit(Modelgames)
Modelgames = scaler.transform(Modelgames)

In [48]: X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.30)
```

Now we are ready to perform our Machine Learning Models.

First, we start by dividing the data by 30%. Our first model was linear regression, whose training accuracy was 70% and 64% in the test.

Figure 17– Linear Regression Model (30%);

```
In [48]: X_train, X_test, y_train, y_test = train_test_split (X, y, test_size=0.30)

In [49]: reg = LinearRegression()
         reg.fit(X_train,y_train)

Out[49]: LinearRegression()

In [50]: y_pred = reg.predict(X_test)

In [51]: Acc_reg30 = reg.score(X_train, y_train)
         acc_reg30 = reg.score(X_test, y_test)
         print ('Train Accuracy : {:.2f}%'.format(Acc_reg30*100))
         print ('Test Accuracy : {:.2f}%'.format(acc_reg30*100))

         Train Accuracy : 70.45%
         Test Accuracy : 64.59%
```

Note: The previous hypothesis we were trying to prove was that Year, Genre and Platform could influence European Sales. However, when we ran that model the accuracy was very low (Please see figure below). Taking that into account we chose to try adding the variables that in the heatmap showed more correlation. i.e. Other Sales and North American Sales. Once we added these features to the model, the accuracy of our models improved significantly. This demonstrates that Sales are a better predictor than the other features we chose at the beginning. See appendix see *Appendix E and F* for more information about application.

The second to be performed was KNN, with a training accuracy: 76.61% and a testing accuracy of 55.82%. KNN looks for the data points that are most similar to the others, we expected this model to be more accurate as we thought it would group the data into families that suggest a better sale (Figure 19). This also suggests an excessive Overfiting of the analysis. The SVR model was also tested with 56.77% training accuracy and 60.00% testing accuracy, possibly with Underfiting.

Figure 18– K-nearest Neighbors Algorithm Model (30%);

```
In [56]: neigh = KNeighborsRegressor()
         neigh.fit(X_train, y_train)

Out[56]: KNeighborsRegressor()

In [57]: pred_neigh = neigh.predict(X_test)

In [58]: Acc_neigh30 = neigh.score(X_train, y_train)
         acc_neigh30 = neigh.score(X_test, y_test)
         print ('Train Accuracy : {:.2f}%'.format(Acc_neigh30*100))
         print ('Test Accuracy : {:.2f}%'.format(acc_neigh30*100))

         Train Accuracy : 78.33%
         Test Accuracy : 49.06%
```

Figure 19– Support Vector Machine Model (30%);

```
In [63]: model_svr = SVR()
         model_svr.fit(X_train, y_train)

Out[63]: SVR()

In [64]: pred_svr = model_svr.predict(X_test)

In [65]: Acc_svr30 = model_svr.score(X_train, y_train)
         acc_svr30 = model_svr.score(X_test, y_test)
         print ('Train Accuracy : {:.2f}%'.format(Acc_svr30*100))
         print ('Test Accuracy : {:.2f}%'.format(acc_svr30*100))

         Train Accuracy : 67.18%
         Test Accuracy : 40.61%
```

We also divided into 20% and 10%, with the precision of: Linear Regression 20% training result 74% and test 34%. KNN 20% training result 85% and test result in 56%. SVR 20% training results in 62% and test 60%. Linear Regression 10% training result 68% and test result from 73%. KNN 10% training result from 80% and test result 86%. SVR 10% training result 65% and test result from 69%.

4.2 CROSS VALIDATION

The cross-validation method, also known as K-fold, consists of dividing the dataset into k parts, using k-1 parts for training and the rest for testing, doing this k times. It is a widely used technique for evaluating the performance of machine learning models.

In the case of this analysis, we divided it into 5 parts, as each part tests a model with a different part calculating the metric chosen to evaluate the model, in which at the end of the process we will have k measures of the chosen evaluation metric, with which it can be calculated the mean and standard deviation, so we got the score.mean() scores of:

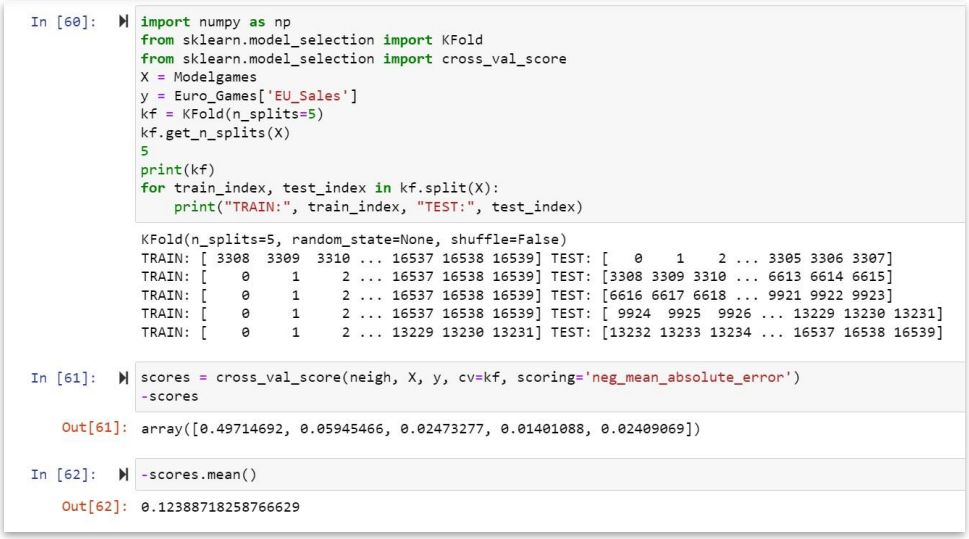
Linear Regression: 0.0977, KNN: 0.123, SVR: 0.140,

Figure 20– CROSS VALIDATION in Linear Regression Model(30%);

```
In [53]: import numpy as np
         from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score
         X = Modelgames
         y = Euro_Games['EU_Sales']
         kf = KFold(n_splits=5)
         kf.get_n_splits(X)
         5
         print(kf)
         for train_index, test_index in kf.split(X):
             print("TRAIN:", train_index, "TEST:", test_index)

         KFold(n_splits=5, random_state=None, shuffle=False)
         TRAIN: [ 3308  3309  3310 ... 16537 16538 16539] TEST: [   0    1    2 ... 3305 3306 3307]
         TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [3308 3309 3310 ... 6613 6614 6615]
         TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [6616 6617 6618 ... 9921 9922 9923]
         TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [ 9924  9925  9926 ... 13229 13230 13231]
         TRAIN: [    0     1     2 ... 13229 13230 13231] TEST: [13232 13233 13234 ... 16537 16538 16539]

In [54]: scores = cross_val_score(reg, X, y, cv=kf, scoring='neg_mean_absolute_error')
         -scores

Out[54]: array([0.32888116, 0.06701275, 0.0399531 , 0.02905079, 0.02368214])

In [55]: -scores.mean()

Out[55]: 0.09771598680450606
```

4.2.1 KNN

Figure 21– CROSS VALIDATION in KNN Model(30%);



```
In [60]:
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
X = Modelgames
y = Euro_Games['EU_Sales']
kf = KFold(n_splits=5)
kf.get_n_splits(X)
5
print(kf)
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)

KFold(n_splits=5, random_state=None, shuffle=False)
TRAIN: [ 3308  3309  3310 ... 16537 16538 16539] TEST: [   0    1    2 ... 3305 3306 3307]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [3308 3309 3310 ... 6613 6614 6615]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [6616 6617 6618 ... 9921 9922 9923]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [ 9924  9925  9926 ... 13229 13230 13231]
TRAIN: [    0     1     2 ... 13229 13230 13231] TEST: [13232 13233 13234 ... 16537 16538 16539]

In [61]:
scores = cross_val_score(neigh, X, y, cv=kf, scoring='neg_mean_absolute_error')
-scores

Out[61]: array([0.49714692, 0.05945466, 0.02473277, 0.01401088, 0.02409069])

In [62]:
-scores.mean()

Out[62]: 0.12388718258766629
```

SVR

Figure 22– CROSS VALIDATION in SVR Model(30%);

```
In [71]:
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
X = Modelgames
y = Euro_Games['EU_Sales']
kf = KFold(n_splits=5)
kf.get_n_splits(X)
5
print(kf)
for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)

KFold(n_splits=5, random_state=None, shuffle=False)
TRAIN: [ 3308  3309  3310 ... 16537 16538 16539] TEST: [   0    1    2 ... 3305 3306 3307]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [3308 3309 3310 ... 6613 6614 6615]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [6616 6617 6618 ... 9921 9922 9923]
TRAIN: [    0     1     2 ... 16537 16538 16539] TEST: [ 9924  9925  9926 ... 13229 13230 13231]
TRAIN: [    0     1     2 ... 13229 13230 13231] TEST: [13232 13233 13234 ... 16537 16538 16539]

In [72]:
scores = cross_val_score(model_svr, X, y, cv=kf, scoring='neg_mean_absolute_error')
-scores

Out[72]: array([0.49549811, 0.05149179, 0.04183394, 0.05239614, 0.06197776])

In [73]:
scores.mean()

Out[73]: -0.14063954552402858
```

Since the CV makes it possible to find out if your model is overfitting in your training data, ie when training and test scores are not close to each other and Underfitting if the training score is far from being perfect score (which is 1.0 for r2). We conclude that our analyzes on the KNN and SVR models are underfitting and LR overfitting.

5. CONCLUSION/RECOMMENDATIONS

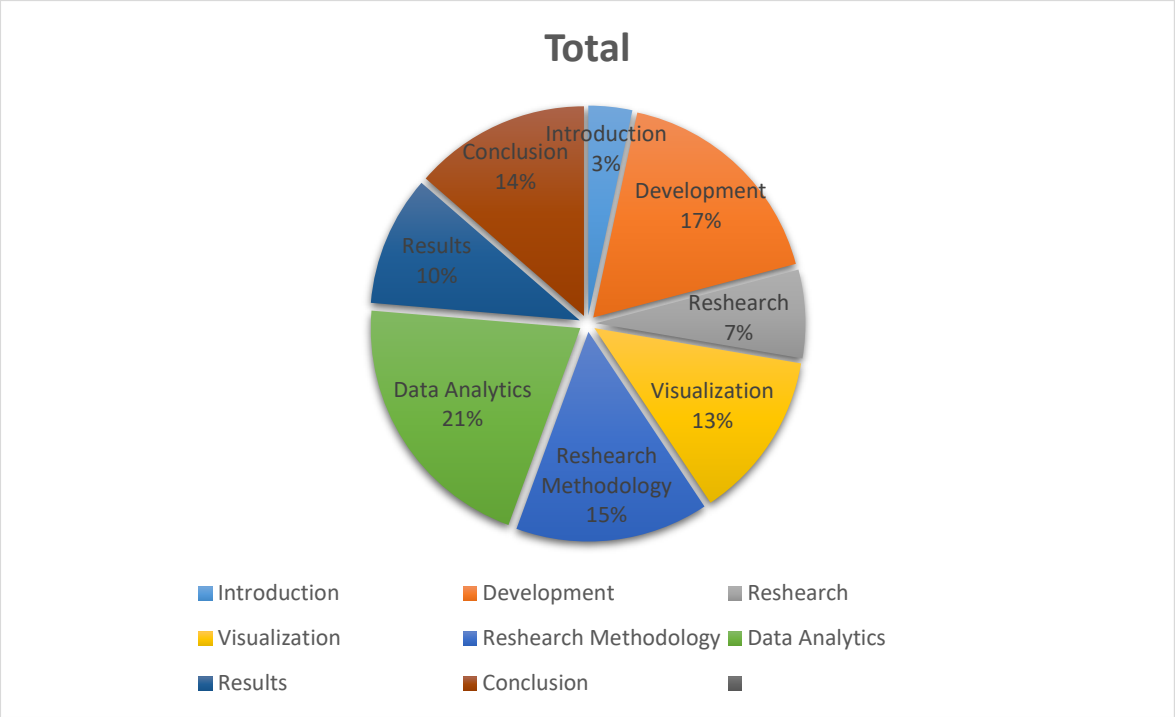
The work was developed to study the variable Europe and Sales, in order to use it in possible ways to forecast the future. In conclusion, we can see several results weighted in the research and confirmed in the jupyter notebook file of the most important and determinant variables so that machine learning models can be positive, sales. The global sale among all the sales variables was the one that obtained the best results in the machine learning models with high accuracy around 89% in the linear regression, 81% KNN. However, this variable was removed from this study because it obtains too many related values that could lead to an unrealistic result. According to the tests performed, we can also note that the variables used are effective in the chosen models.

The models performed with X= European Sales and X= Plataform, Genre, Year, North America and Other Sales after all the Data Cleaning and Data Understanding. Used to understand the Test accuracy around 70% in the models applied in that Report. Note that this results may change every time the notebooks is run bc the random sample for testing and training changes every time.

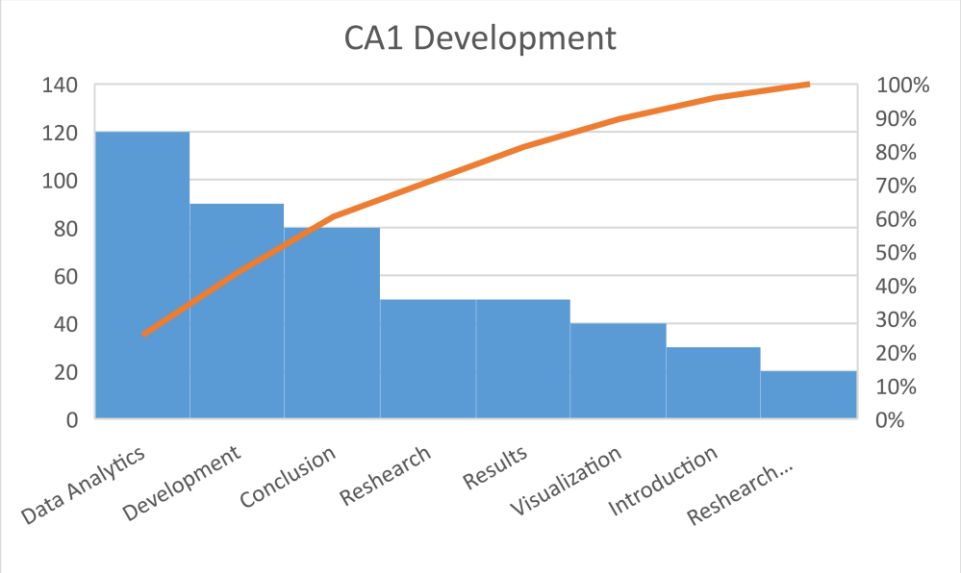
So it can be understood that to reduce the variables we can maintain the relationships between North America and Other Sales and apply it in a future study of Predication in Machine Learning Models.

6. APPENDICES

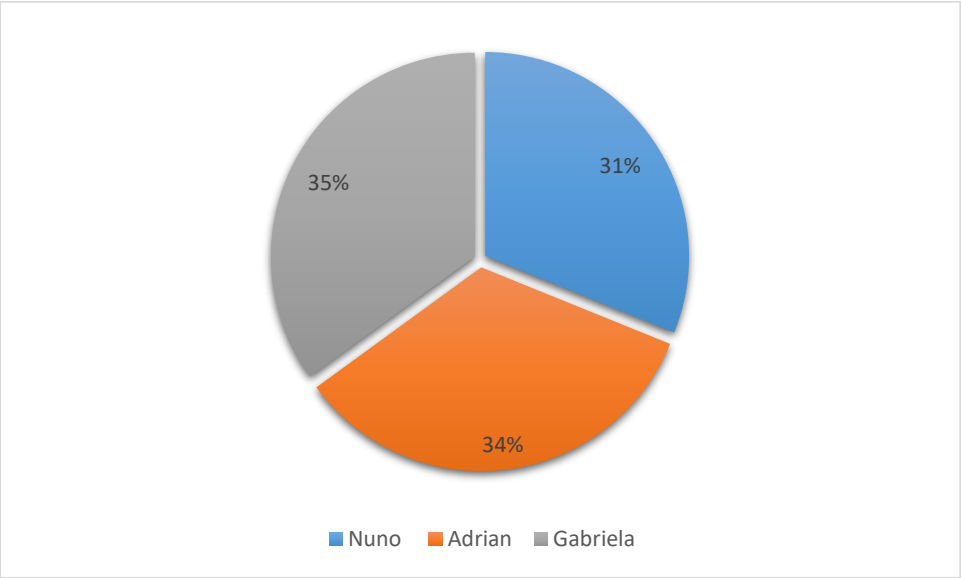
Appendix A– Pie chart on percentage of developed parts;



Appendix B– Linear graph of hours worked and Completion of work by category;



Appendix C– Pie chart on total in percentage of hours worked by each component;



Appendix D– Data Frame created to show the Performance hours by Components Group;

```
In [159]: df
```

Out[159]:

	Developement part	Performance Nuno	Performance Gabriela	Performance Adriam
0	Introduction	30	12	10
1	Development	90	90	90
2	Reshearch	10	30	25
3	Visualisation	40	90	120
4	Research Methodology	10	40	120
5	Data Analysis	120	120	80
6	Results	50	60	45
7	Conclusion	80	80	50

Appendix E– Linear Regression Mode (30%) without North American and Global Sales;

Linear Regression 30%

In [124]:
```
reg = LinearRegression()
reg.fit(X_train,y_train)
```

Out[124]:
```
LinearRegression()
```

In [125]:
```
y_pred = reg.predict(X_test)
```

In [126]:
```
Acc_reg30 = reg.score(X_train, y_train)
acc_reg30 = reg.score(X_test, y_test)
print ('Train Accuracy : {:.2f}%'.format(Acc_reg30*100))
print ('Test Accuracy : {:.2f}%'.format(acc_reg30*100))
```

```
Train Accuracy : 0.21%
Test Accuracy : 0.31%
```

In [127]:
```
print('Mean squared error: %.2f'% mean_squared_error(y_test, y_pred))
print('Variance Score: %.2f'% r2_score(y_test, y_pred))
```

```
Mean squared error: 0.20
Variance Score: 0.00
```

Appendix F– K-nearest Neighbors Algorithm Model (30%) without North American and Global Sales;

KNN 30%

In [129]:
```
neigh = KNeighborsRegressor()
neigh.fit(X_train, y_train)
```

Out[129]:
```
KNeighborsRegressor()
```

In [130]:
```
pred_neigh = neigh.predict(X_test)
```

In [131]:
```
Acc_neigh30 = neigh.score(X_train, y_train)
acc_neigh30 = neigh.score(X_test, y_test)
print ('Train Accuracy : {:.2f}%'.format(Acc_neigh30*100))
print ('Test Accuracy : {:.2f}%'.format(acc_neigh30*100))
```

```
Train Accuracy : 8.24%
Test Accuracy : -17.51%
```

In [132]:
```
print('Mean squared error: %.2f'% mean_squared_error(y_test, pred_neigl
print('Variance Score: %.2f'% r2_score(y_test, pred_neigh))
```

```
Mean squared error: 0.24
Variance Score: -0.18
```

REFERENCES

3.1. Cross-validation: evaluating estimator performance (no date) *scikit-learn*. Available at: https://scikit-learn/stable/modules/cross_validation.html (Accessed: 1 December 2021).

3.3. Metrics and scoring: quantifying the quality of predictions (no date) *scikit-learn*. Available at: https://scikit-learn/stable/modules/model_evaluation.html (Accessed: 1 December 2021).

Afonja, T. (2018) *Kernel Functions*, *Medium*. Available at: https://towardsdatascience.com/kernel-function-6f1d2be6091 (Accessed: 19 November 2021).

APD, R. (2019) '¿Cuáles son los tipos de algoritmos del machine learning?', *APD España*, 4 April. Available at: https://www.apd.es/algoritmos-del-machine-learning/ (Accessed: 18 November 2021).

Bhattacharyya, I. (2018) 'Support Vector Regression or SVR'.

Bose, A. (2021) *Cross Validation — Why & How*, *Medium*. Available at: https://towardsdatascience.com/cross-validation-430d9a5fee22 (Accessed: 1 December 2021).

Brownlee, J. (2017) 'Difference Between Classification and Regression in Machine Learning', *Machine Learning Mastery*, 10 December. Available at: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/ (Accessed: 19 November 2021).

Brownlee, J. (2020) 'Repeated k-Fold Cross-Validation for Model Evaluation in Python', *Machine Learning Mastery*, 2 August. Available at: https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/ (Accessed: 1 December 2021).

Burns, Ed (2021) 'Machine Learning'. Avaiable at https://searchenterpriseai.techtarget.com/definition/machine-learning-ML .(Accessed 17 November 2021)

Chen, C., Härdle w., Unwin A., (2008) Handbook of Data Visualization. Berlin,Heidelberg: Springer-Verlag.

Duca, A.L. (2021) *How to Check if a Classification Model is Overfitted using scikit-learn*, *Medium*. Available at: https://towardsdatascience.com/how-to-check-if-a-classification-model-is-overfitted-using-scikit-learn-148b6b19af8b (Accessed: 2 December 2021).

Government of Canada, S.C. (2002) *Statistics: Power from Data! Five-number summary*. Available at: https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214877-eng.htm (Accessed: 2 December 2021).

Gregersen, E. and Britannica (2021) *five-number summary | statistics*, *Encyclopedia Britannica*. Available at: https://www.britannica.com/science/five-number-summary (Accessed: 4 November 2021).

Iqbal, M. (2019) 'Gaussian Naïve Bayes and Support Vector Machine.'

Jain, D. (2020) *Cross-validation using KNN*, *Medium*. Available at: https://towardsdatascience.com/cross-validation-using-knn-6babb6e619c8 (Accessed: 2 December 2021).

Java (2021) *Unsupervised Machine learning - Javatpoint*, *www.javatpoint.com*. Available at: https://www.javatpoint.com/unsupervised-machine-learning (Accessed: 2 December 2021).

Jones, T. (2018) 'Supervised learning models', *IBM Developer*, 26 February. Available at: https://developer.ibm.com/articles/cc-supervised-learning-models/ (Accessed: 18 November 2021).

New Zoo. Available at: (https://newzoo.com/insights/articles/newzoo-games-market-numbers-revenues-and-audience-2020-2023/) (Accessed 30 November 2021).

McQuaid, D. (2019) 'What is Exploratory Data Analysis'.

Mode (2016) *Pandas .values_count() & .plot()* | *Python Analysis Tutorial - Mode*, *Mode Resources*. Available at: https://mode.com/python-tutorial/counting-and-plotting-in-python/ (Accessed: 4 November 2021).

Müller, A.C. Guido, S. (2017) Introduction to Machine Learning with Python, A Guide for Data Scientists. USA. O'Reilly.

NG, R. (no date) *Cross-Validation*, *ritchieng.github.io*. Available at: http://www.ritchieng.com/machine-learning-cross-validation/ (Accessed: 2 December 2021).

Pandas documentation. Available at:https://pandas.pydata.org/pandas-docs/stable/index.html (Accessed 30 November 2021).

PC Gamer. Available at: https://Steam now has 30,000 games | PC Gamer (Accessed 30 November 2021).

Pyle, D. (1999) Data Preparation for Data Mining. San Francisco, USA. Morgan Kaufmann.

Ray, S. (2018) 'Cross Validation | Cross Validation In Python & R', *Analytics Vidhya*, 3 May. Available at: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/ (Accessed: 2 December 2021).

SAS Institute (2021) *Machine Learning: What it is and why it matters*. Available at: https://www.sas.com/en_in/insights/analytics/machine-learning.html (Accessed: 2 December 2021).

Saxena (2020) '8 Categorical Data Encoding Techniques to Boost your Model in Python!', *Analytics Vidhya*, 13 August. Available at: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/ (Accessed: 2 December 2021).

sklearn.model_selection.cross_val_score (no date) *scikit-learn*. Available at: https://scikit-learn/stable/modules/generated/sklearn.model_selection.cross_val_score.html (Accessed: 1 December 2021).

Smith, G. (2017) *Video Game Sales*. Available at: https://kaggle.com/gregorut/videogamesales (Accessed: 19 November 2021).

Stapel, E. (2021) *Mean, Median, Mode, and Range*, *Purplemath*. Available at: https://www.purplemath.com/modules/meanmode.htm (Accessed: 2 December 2021).

Statista. Available at: https://statista.com/statistics/552623/number-games-released-steam/(Accessed 30 November 2021).

Sullivan, J. (2018) *Data Cleaning with Python and Pandas: Detecting Missing Values*, *Medium*. Available at: https://towardsdatascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b (Accessed: 6 November 2021).

Taylor, C., M. S., M. and B. A., M. (2018) *What Is the 5 Number Summary in Statistics?*, *ThoughtCo*. Available at: https://www.thoughtco.com/what-is-the-five-number-summary-3126237 (Accessed: 19 November 2021).

Witten I. H., Frank E. (2005) Data Mining: Practical Machine Learning Tools and Techniques (Second Edition). Northern Ireland, UK. Morgan Kaufmann.