**VIDEO GAMES SALES**

**PROJECT SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD FOR THE DEGREE OF**

**BACHELORS OF COMPUTER APPLICATION**

SUBMITTED BY

**ADITYA DEY**

**(2075003)**

***Under The Guidance Of***

**Mr. Sudhansu Shekhar Patra**

**(School Of Computer Application)**



**School Of Computer Applications**

**Kalinga Institute Of Industrial Technology**

**[Deemed to be University]**

**April 2023**

**CERTIFICATE OF ORIGINALITY**

This is to certify that the project report entitled “**VIDEO GAME SALES”**

submitted to School of Computer Application, KIIT University in partial fulfilment of the requirement for the award of the degree of **BACHELORS OF COMPUTER APPLICATIONS (BCA)**, is an authentic and original work carried out by **Mr. Aditya Dey** with **Roll no. 2075003** and **Regd. No. 20417376847** under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University for the fulfilment of the requirements of any course of study.

**Signature of the Student: Signature of the Guide**

Date: ……………….. Date:…………………

**School of Computer Application**

**KIIT Deemed to be University University, Bhubaneswar**

**CERTIFICATE**

This is to certify that the project work entitled **Video Game Sales** Submitted By **Aditya Dey** bearing roll no. **2075003,** is an authentic and original Work.

**Signature Signature**

**(Internal Examiner) (External Examiner)**

**Date.................. Date..................**

**DECLARATION**

I , **Aditya Dey** , roll no **2075003** do hereby declare that the project report entitled **“Video Game Sales”** submitted to School of Computer Application, KIIT University, Bhubaneswar for the award of the degree of **BACHELORS OF COMPUTER APPLICATION (BCA)** , is an authentic and original work carried out by me from 1st Jan 2023 to 1st May 2023 under the guidance of **Mr. Sudhansu Shekhar Patra.**

**Signature of the Student**

**Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ACKNOWLEDGMENT**

I would like to express my gratitude to everyone for their support and assistance in the completion of this Data Science project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. My team members, **Aayush Kumar Panda** and **Rohan Behera**, for their hard work, dedication, and expertise that contributed to the success of this project.

I express my warm thanks to **Mr. Sudhansu Shekhar Patra** for their support and guidance at **KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT) DEEMED TO BE UNIVERSITY**

**Date\_\_\_\_\_\_\_\_\_\_ Signature of the Student**

**ADITYA DEY Place\_\_\_\_\_\_\_\_\_\_ ROLL\_NO: 2075003**

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **SL NO** | **CONTENT** | **PAGE NO** |
| **1** | 1. **Introduction**   1.1 What is DataScience  1.2 Data Science Life Cycle  1.3 Different steps involved while handling a dataset | **10-14** |
| **2** | 1. **Dataset**   2.1 Video Game Sales Dataset  2.2 About Dataset  2.3 Potential angles to consider while handling sales dataset. | **15-17** |
| **3** | 1. **Steps**   **3.1 Data Preprocessing**  3.1.1 Import the libraries  3.1.2. Load the dataset  3.1.3. Inspect the dataframe  **3.2 Data cleaning**  3.2.1. Fill NAN values  3.2.1.1 Fill NAN values of “Publisher”  3.2.1.2 Fill NAN values of “Year”  **3.3 Data Analysis and Visualization**  3.3.1 Using Describe () Function  3.3.2 Top 20 highest grossing Games  3.3.3 Best selling game in NA  3.3.4 Most common year for video game release  3.3.5 Top categories of games sold  3.3.6 Game genre and their profit  3.3.7 Year releasing most games  3.3.8 Video games global sales trend by year  3.3.9 Highest sales price Globally  3.3.10 Overview sales comparison between Genre  3.3.11 top 20 Publisher  3.3.11.1 Top Publisher each year  3.3.12 Total revenue by region  3.3.13 Sales from 1980-2020  3.3.14 Most played game globally  3.3.15 Scatterplot 1  3.3.16 Scatterplot 2  3.3.17 Pairplot Graph | **17**  **17-22**  **22-27**   1. **49** |
| **4** | 1. **Prediction**   **4.1 Set the dependent and independent variable**  4.1.1 Converting Independent and dependent variables  4.1.2 Split the dataset into training and testing sets  **4.2 Algorithms in predictions**  4.2.1 Decision Tree  4.2.1.1 Fit the model  4.2.1.2 Make Predictions  4.2.1.3 Calculate score of model  4.2.1.4 Calculate accuracy  4.2.2 Linear Regression  4.2.2.1 Fit the model  4.2.2.2 Make Predictions  4.2.2.3 Calculate score of model  4.2.2.4 Calculate accuracy  **4.3 Store the results**  **4.4 Confusion Matrix**  4.4.1 Confusion Matrix of Decision Tree  4.4.1.1 Convert Predictions into binary classes  4.4.1.2 Create binary classification target variable  4.4.1.3 Compute confusion matrix for decision tree  4.4.2 Confusion Matrix of Linear Regression  4.4.2.1 Convert Predictions into binary classes  4.4.2.2 Compute confusion matrix for Linear Regression  4.4.3 Conclusion of overall prediction | **50**  **50**  **51-54**  **54**  **54-57** |
| **5** | 1. **Limitation of Video game sales dataset** | **59** |
| **6** | 1. **Future scope of video game sales** | **60** |
| **7** | 1. **Conclusion** | **61** |
| **8** | 1. **Bibliography** | **62** |

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Title** | **Page no** |
| 1 | Over all view point of DataScience | 11 |
| 2 | Lifecycle of DataScience | 12 |
| 3 | Methodology used in reprocessing | 18 |
| 4 | Import Libraries | 18 |
| 5 | Read the dataset | 20 |
| 6 | Inspect the data frame | 21 |
| 7 | Column wise null count | 23 |
| 8 | Column wise null count in percentage | 23 |
| 9 | Fill NAN values of publisher | 24 |
| 10 | Fill missing values of publisher | 25 |
| 11 | Count rows with unknown values | 25 |
| 12 | Filling missing values of year | 26 |
| 13 | Convert data type to integer | 26 |
| 14 | Checking all NAN values filled or not | 27 |
| 15 | Checking all NAN values filled or not in percentage | 27 |
| 16 | Using Describe function | 28 |
| 17 | Top 20 Highest grossing games | 28 |
| 18 | Top 20 Highest grossing games output | 29 |
| 19 | Best selling game in North America | 29 |
| 20 | Most common year for video game release | 30 |
| 21 | Top 10 categories of games sold | 30 |
| 22 | Output of top 10 categories of games sold | 31 |
| 23 | Genre that generated highest profit | 31 |
| 24 | Top genre Produced | 32 |
| 25 | Year release most games | 33 |
| 26 | Visual representation of year release most games | 33 |
| 27 | Game produced each year | 34 |
| 28 | Visual representation of game produced each year | 34 |
| 29 | Global sales trend by year | 35 |
| 30 | Global sales for each year | 35 |
| 31 | Visual representation of global sales for each year | 36 |
| 32 | Platform with highest sales price globally | 36 |
| 33 | Highest sales price globally | 36 |
| 34 | Plotting Graph of top ten highest sales globally for all the platforms | 37 |
| 35 | Platform vs Global sales | 37 |
| 36 | Sales comparison between genre | 38 |
| 37 | Plotting graph of sales comparison between genre | 38 |
| 38 | Heat map of sales comparison between genre | 39 |
| 39 | Making a precise graph of sales comparison between genre | 39 |
| 40 | Output of sales comparison between genre | 40 |
| 41 | Visual representation of sales comparison between genre | 40 |
| 42 | Top 20 publisher | 41 |
| 43 | Visual representation top 20 publisher | 41 |
| 44 | Top publisher each year | 42 |
| 45 | Top publisher count plotting graph | 43 |
| 46 | Visual representation top publisher count | 43 |
| 47 | Total revenue by region | 43 |
| 48 | Pie chart of total revenue by region | 44 |
| 49 | Sales from year 1980-2020 | 45 |
| 50 | Visuals of sales from year 1980-2020 | 45 |
| 51 | Most played video game | 46 |
| 52 | Analyzing the most popular video games | 46 |
| 53 | Most loved games over the years | 47 |
| 54 | How Jp\_sales distributed on each rank | 47 |
| 55 | How global\_sales distributed on each rank | 48 |
| 56 | Pairplot comparison between all numerical column | 49 |
| 57 | Converting dependent and independent variables | 50 |
| 58 | Split the data into training and testing dataset | 50 |
| 59 | Decision Tree (Fit the model) | 51 |
| 60 | Decision Tree (make prediction) | 51 |
| 61 | Decision Tree (score of model) | 52 |
| 62 | Decision Tree (accuracy of the model) | 52 |
| 63 | Linear regression (fit the model) | 53 |
| 64 | Linear regression (make prediction) | 53 |
| 65 | Linear regression (score of model) | 53 |
| 66 | Linear regression (accuracy of the model) | 54 |
| 67 | Storing the result | 54 |
| 68 | Convert prediction into binary classes(DT) | 55 |
| 69 | Create binary classification target variable (DT) | 55 |
| 70 | Compute confusion matrix for Decision Tree | 55 |
| 71 | Convert prediction into binary classes (LR) | 56 |
| 72 | Compute confusion matrix for LR | 57 |

**1.INTRODUCTION**

**1.1 What is Data Science?**

Data science is an interdisciplinary field that involves the extraction of insights and knowledge from data, using a variety of statistical, mathematical, and computational methods. It combines techniques from statistics, machine learning, and computer science to make sense of complex data sets and provide solutions to a wide range of real-world problems.

The primary goal of data science is to extract actionable insights and knowledge from data to make informed decisions. This process involves collecting and analyzing data, building models, and making predictions or recommendations based on the results.

Data science is used in a variety of fields, including business, healthcare, finance, and social sciences.

Data science requires a combination of technical and non-technical skills. Technical skills include programming, data visualization, statistical analysis, and machine learning, while non-technical skills include critical thinking, problem-solving, and communication.

In summary, data science is a field that utilizes statistical and computational methods to extract insights and knowledge from complex data sets to solve real-world problems.

In today's era of digitalization, data is constantly generated at an unprecedented rate, and analyzing this data has become an essential component of decision-making in various fields. The goal of our project is to use data science techniques to extract insights and patterns from a large dataset.

Recent years have also witnessed a massive growth in the field of Internet of Things(IOT),which alone produces 90% of the data that is being generated. Approximately 2.5 quintillion bytes of data is generated every day.Sources of such data are:

* Sensors used in shopping malls to collect shopper’s information
* Posts, Pictures and videos shared on social media platforms
* Text, pictures and videos in our smartphones and tablets
* Purchase transactions completed through e-commerce websites

This data is known is big data. Companies that generate huge amounts of data always look at possibilities to analyse and utilize the data.

Data science encompasses skills from multiple disciplines like statistics ,mathematics, business domain and machine learning, to name a few.

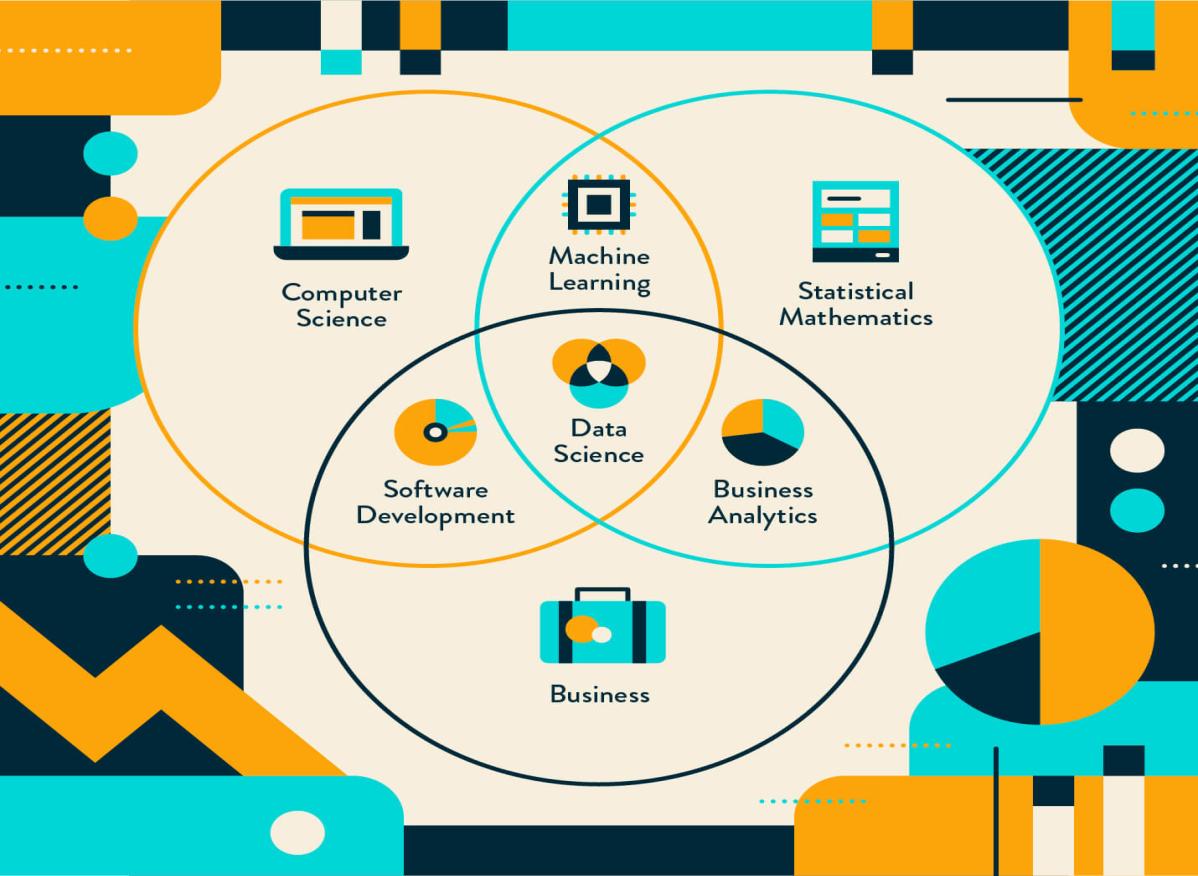


Figure 1: Overall view point of data Science

The dataset we are working with includes information from a variety of sources, including online surveys, social media, and public data repositories. We will leverage various tools and techniques, such as statistical analysis, machine learning, and data visualization, to uncover hidden patterns and relationships within the data.

The insights gained from our analysis will be useful for decision-makers in various industries, including marketing, healthcare, and finance. For example, we may identify demographic trends and preferences that can be used to inform marketing strategies or reveal patterns in medical data that can be used to improve patient outcomes.

We are excited to embark on this project and share our findings with the broader community.

The dataset we have used is a large and complex dataset that requires sophisticated tools and techniques to extract meaningful insights.

**1.2 Data Science Life Cycle**

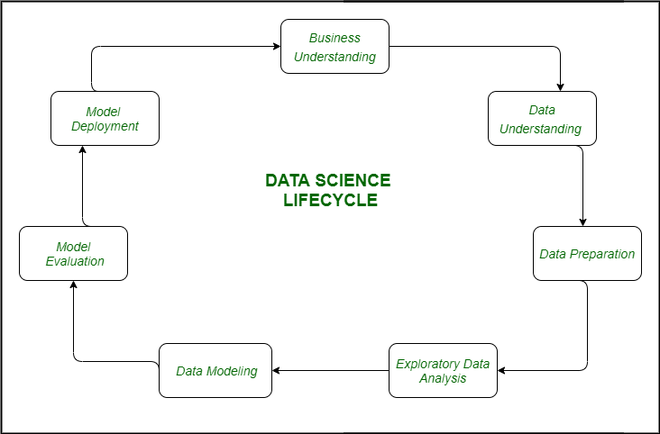
****

Figure 2: Lifecycle of Data Science

* **Business Understanding:**

The complete cycle revolves around the enterprise goal. It is extraordinarily essential to apprehend the commercial enterprise goal sincerely due to the fact that will be your ultimate aim of the analysis. After desirable perception only we can set the precise aim of evaluation that is in sync with the enterprise objective. You need to understand if the customer desires to minimize savings loss, or if they prefer to predict the rate of a commodity, etc.

* **Data Understanding:**

After enterprise understanding, the subsequent step is data understanding. This includes a series of all the reachable data. Here you need to intently work with the commercial enterprise group as they are certainly conscious of what information is present, what facts should be used for this commercial enterprise problem, and different information. This step includes describing the data, their structure, their relevance, their records type. Explore the information using graphical plots.

* **Preparation of Data:**

Next comes the data preparation stage. This consists of steps like choosing the applicable data, integrating the data by means of merging the data sets, cleaning it, treating the lacking values through either eliminating them or imputing them, treating inaccurate data through eliminating them, additionally test for outliers the use of box plots and cope with them. Constructing new data, derive new elements from present ones. Format the data into the preferred structure, eliminate undesirable columns and features. Data preparation is the most time-consuming but arguably the most essential step in the complete existence cycle. Your model will be as accurate as your data.

* **Exploratory Data Analysis:**

This step includes getting some concept about the answer and elements affecting it, earlier than constructing the real model. Distribution of data inside distinctive variables of a character is explored graphically the usage of bar-graphs, Relations between distinct aspects are captured via graphical representations like scatter plots and warmth maps. Many data visualization strategies are considerably used to discover each and every characteristic individually and by means of combining them with different features.

* **Data Modelling:**

Data modelling is the coronary heart of data analysis. A model takes the organized data as input and gives the preferred output. This step consists of selecting the suitable kind of model, whether the problem is a classification problem, or a regression problem or a clustering problem. After deciding on the model family, amongst the number of algorithms amongst that family, we need to cautiously pick out the algorithms to put into effect and enforce them. We need to tune the hyperparameters of every model to obtain the preferred performance. We additionally need to make positive there is the right stability between overall performance and generalizability.

* **Model Evaluation:**

Here the model is evaluated for checking if it is geared up to be deployed. The model is examined on an unseen data, evaluated on a cautiously thought out set of assessment metrics. We additionally need to make positive that the model conforms to reality. If we do not acquire a quality end result in the evaluation, we have to re-iterate the complete modelling procedure until the preferred stage of metrics is achieved. Any data science solution, a machine learning model, simply like a human, must evolve, must be capable to enhance itself with new data, adapt to a new evaluation metric. We can construct more than one model for a certain phenomenon, however, a lot of them may additionally be imperfect. The model assessment helps us select and construct an ideal model.

* **Model Deployment:**

The model after a rigorous assessment is at the end deployed in the preferred structure and channel. This is the last step in the data science life cycle. Each step in the data science life cycle defined above must be laboured upon carefully. If any step is performed improperly, and hence, have an effect on the subsequent step and the complete effort goes to waste. For example, if data is no longer accumulated properly, you’ll lose records and you will no longer be constructing an ideal model. If information is not cleaned properly, the model will no longer work. If the model is not evaluated properly, it will fail in the actual world. Right from Business perception to model deployment, every step has to be given appropriate attention, time, and effort.

**1.3 Different steps involved while handling a dataset:**

**Process of building a predictive model using pandas.**

* **Import the required libraries:**

You will need to import the pandas library along with the libraries for the specific machine learning algorithm you plan to use. For example, if you plan to use scikit-learn, you will need to import sklearn.

* **Load the dataset:**

Load the dataset into a pandas dataframe using the read\_csv() function. You can also load data from other sources such as Excel, SQL, and JSON.

* **Explore the dataset:**

Get an overview of the dataset using functions like head(), tail(), info(), describe(), etc.

These functions will help you understand the structure, types, and values of the dataset.

* **Preprocess the data:**

Preprocessing is an important step to clean and transform the data for analysis. Some common preprocessing steps include handling missing values, encoding categorical variables, scaling numeric features, and removing outliers.

* **Split the data into training and testing sets:**

Split the dataset into training and testing sets. The training set is used to build the model, while the testing set is used to evaluate the performance of the model.

* **Select the algorithm:**

Select the appropriate machine learning algorithm for your problem. For example, if you have a classification problem, you can use algorithms like Logistic Regression, Naive Bayes, Decision Trees, or Random Forests.

* **Train the model:**

Train the model on the training set using the fit() function.

* **Evaluate the model:**

Evaluate the performance of the model on the testing set using metrics like accuracy, precision, recall, and F1 score.

* **Predict using the model:**

Use the predict() function to make predictions on new data.

* **Save the model:**

Save the trained model to a file using the joblib library, so that you can use it later.

**2.DATASET**

**2.1 VIDEO GAME SALES DATASET**

Video games have become a global phenomenon, with millions of people across the world playing games on various platforms such as consoles, PCs, and mobile devices. The video game industry has grown significantly in recent years, with sales figures reflecting the growing popularity of this form of entertainment.

One of the positive aspects of video games is their ability to bring people together, regardless of their location. Multiplayer games allow players from all over the world to connect and compete or collaborate, building friendships and communities along the way. In this way, video games can help to break down barriers and promote social interaction.

Video games can also help to improve cognitive skills such as problem-solving, spatial awareness, and decision-making, making them a valuable tool for education and personal development.

The global sales figures for video games are a testament to their popularity and the positive impact they can have on individuals and communities. The industry has seen significant growth in recent years, with sales reaching over $180 billion in 2020 alone.

This success has led to increased innovation and creativity within the industry, with developers continually pushing the boundaries of what is possible in gaming.

In summary, video games have become a global phenomenon that brings people together. The continued success of the video game industry reflects the positive impact that games can have on individuals and communities worldwide.

**2.2 About Dataset**

The following model involves to predict the sales outcome of a video game title in a given period of time. It tries to forecast the copies sold in the observed regions across the world, predominantly from North America, Europe, Japan and etc…

Scope of the problem addressed is how video game publishers can predict the value and profitability from making the sales in video games, ranging from the features that affect the sales towards the audience liking,

for example, how attractive the name is, what popular platforms do players use at that time, along with the important factor of genre which determines the users' liking on a certain game aspect and observing the global sales overtime to show which region does a specific video game have the high rating for the database worldwide - region varies due to the predominance of income, status and access to technology.

In this project, it will adopt the **linear regression** and **decision tree** method to be applied in the imported dataset.

The **Video Game Sales** dataset contains a list of video games with sales greater than 100,000 copies.

It was generated by a scrape of [**vgchartz.com**](http://www.vgchartz.com/)**.**

**Fields include**

**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)

**EU\_Sales -** Sales in Europe (in millions)

**JP\_Sales -** Sales in Japan (in millions)

**Other\_Sales -** Sales in the rest of the world (in millions)

**Global\_Sales -** Total worldwide sales.

There are **16,598** records. 2 records were dropped due to incomplete information

**2.3 Here are some potential topics or angles to consider when writing about your video games sales dataset:**

* **Top-selling games:**

You could examine which video games have sold the most copies, both overall and within specific time periods or genres. You could also look at the factors that may have contributed to their success, such as marketing campaigns, positive reviews, or strong word-of-mouth.

* **Sales by platform:**

Another approach is to analyze sales data by platform, such as console, PC, or mobile. This could reveal trends in gaming preferences across different demographics or regions, as well as the impact of new hardware releases on sales.

* **Regional sales:**

You could also explore how video game sales vary by region, both globally and within specific countries. This could include examining differences in platform preferences, cultural factors that influence game selection, or variations in marketing strategies.

* **Genre trends:**

You could investigate which video game genres are currently most popular, or have seen the biggest growth in sales over time. This could involve analyzing data on specific game titles, as well as broader trends in player preferences.

* **Sales predictions:**

Finally, you could use your dataset to make predictions about future video game sales, based on historical data and other relevant factors. This could include forecasting the success of upcoming releases, or predicting how market changes (such as the rise of cloud gaming or new platform releases) might impact sales trends

1. **STEPS**
   1. **Data Preprocessing**

**Data preprocessing** in a dataframe refers to the series of steps that are taken to clean, transform and prepare the raw data in the dataframe for analysis.

This involves several techniques such as **removing duplicates, handling missing values, standardizing or normalizing the data, encoding categorical variables, scaling numeric variables,** and more.

Data preprocessing is an essential step in data analysis and machine learning because it ensures that the data is of high quality, consistent, and in a format that can be easily analyzed. By performing data preprocessing on a dataframe, you can ensure that the data is suitable for the analysis or modelling tasks you plan to perform.

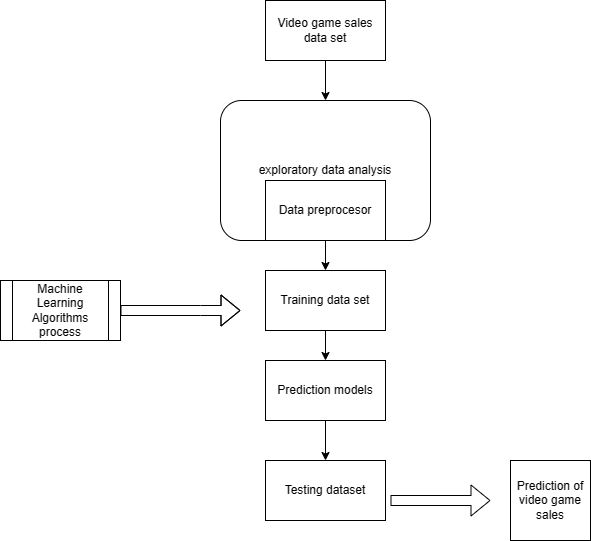
**Methodology used in data preprocessing:**

Figure 3: Methodology used in data preprocessing

* + 1. **Import The Libraries**

Importing libraries is necessary when working with dataframes and other tools in Python because it allows you to access the functions and methods provided by those libraries, which in turn enables you to perform data analysis and manipulation tasks.

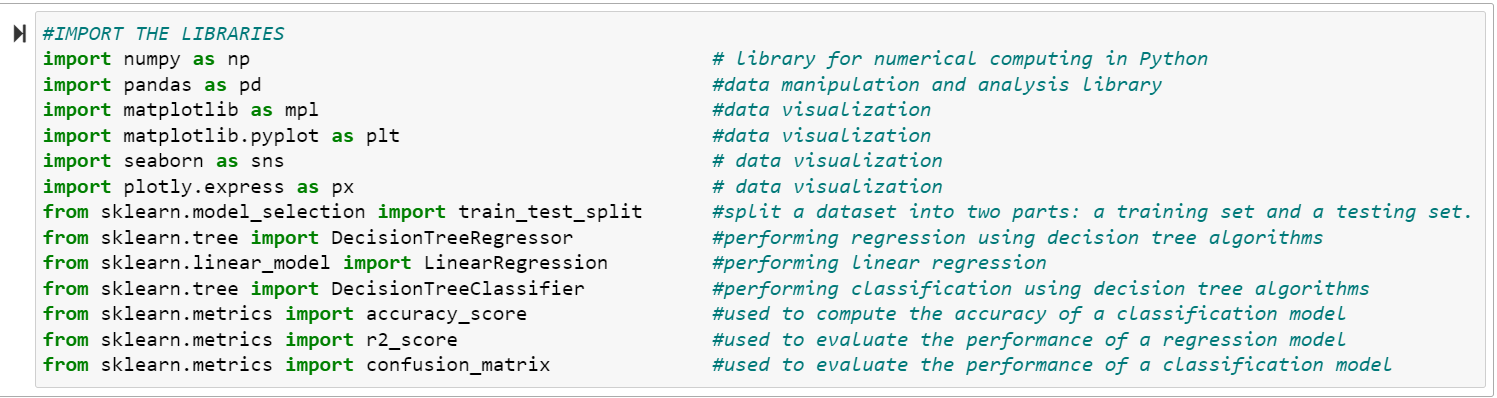
****

Figure 4: Import Libraries

* **Import numpy as np :**

Numpy is a popular numerical computing library in Python that **provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays**. By importing numpy, we can use its functions and methods in our code.

* **Import pandas as pd:**

Pandas is a popular data manipulation library in Python that **provides support for working with data frames, series, and various file formats such as CSV, Excel, and SQL**. By importing pandas, we can use its functions and methods in our code.

* **Import matplotlib as mpl:**

Matplotlib is a popular visualization library in Python that **provides support for creating a wide range of plots such as line, scatter, bar, and histogram charts.** By importing matplotlib, we can use its functions and methods in our code.

It is often used in combination with **import matplotlib.pyplot as plt** to make plotting more concise.

* **Import seaborn as sns:**

Seaborn is a popular data visualization library in Python that **provides support for creating various types of statistical plots such as scatter plots, line plots, histograms, and heatmaps.**

* **Import plotly.express as px:**

Plotly Express is a high-level data visualization library in Python that **provides support for creating various types of plots such as scatter plots, line plots, bar charts, and 3D plots.**

* **from sklearn.model\_selection import train\_test\_split:**

Scikit-learn (sklearn) is a popular machine learning library in Python that **provides various tools for machine learning tasks such as classification, regression, and clustering**. The train\_test\_split function from the model\_selection module is used to split a dataset into two parts - **a training set and a testing set** - which can be used to train and evaluate a machine learning model.

* **from sklearn.tree import DecisionTreeRegressor**  :

The DecisionTreeRegressor class from the tree module is **used to create a decision tree regression model,** which can be used to make predictions based on a set of input features.

* **from sklearn.linear\_model import LinearRegression :**

The LinearRegression class from the linear\_model module is **used to create a linear regression model**, which can be used to make predictions based on a set of input features.

* **from sklearn.metrics import accuracy\_score :**

The accuracy\_score function from the metrics module is **used to calculate the accuracy of a classification model.**

* **from sklearn.metrics import r2\_score :**

The r2\_score function is a commonly **used metric for evaluating the performance of regression models**. It calculates the coefficient of determination, which measures how well the model fits the data. The score ranges from 0 to 1, where **1 indicates a perfect fit and 0 indicates a poor fit.**

* **from sklearn.metrics import confusion\_matrix :**

The confusion\_matrix function is **used to create a confusion matrix**, which is a **table that summarizes the performance of a classification model**. It compares the predicted values of a model with the true values and shows the number of true positives, false positives, true negatives, and false negatives

.**3.1.2 Load The Dataset**

Loading a dataset in a dataframe means reading the data from a source such as a CSV file, SQL database, or Excel sheet, and creating a dataframe object in memory to hold the data.

A dataframe is a two-dimensional table-like data structure in which each column can have a different data type. It is a very popular data structure in Python for data analysis because it allows for easy manipulation and analysis of data.

In Python, the Pandas library provides functions and classes to work with dataframes. To load a dataset into a dataframe using Pandas, you typically use the read\_csv(), read\_excel(), or read\_sql() function depending on the source of the data.

**Here in our project we have used a .csv file.**

**Source code:**



Figure 5: Read the data set

**CODE:**

**vg=pd.read\_csv("vgales.csv")**

**vg**

It seems that you are reading a CSV file "vgsales.csv" and storing it in a Pandas DataFrame called "vg". The file is likely related to video game sales data, given the variable name "vgsales".

**The table shown is our dataset.**

* **Why it is not showing the entire dataset but showing the top 5 and bottom 5 datas?**
* This default behavior is intended to provide a quick preview of the DataFrame's structure and content without overwhelming the user with too much information at once.

**3.1.3 Inspect The Dataframe**

**Code:**

Vg.shape

**Output:**

(16598, 11)

The **.shape** attribute of a Pandas DataFrame **returns a tuple representing the dimensions(Number of rows and colums) of the DataFrame**, with the first element representing the number of rows and the second element representing the number of columns.

**Here 16598 shows the number of rows and 11 shows the number of column.**

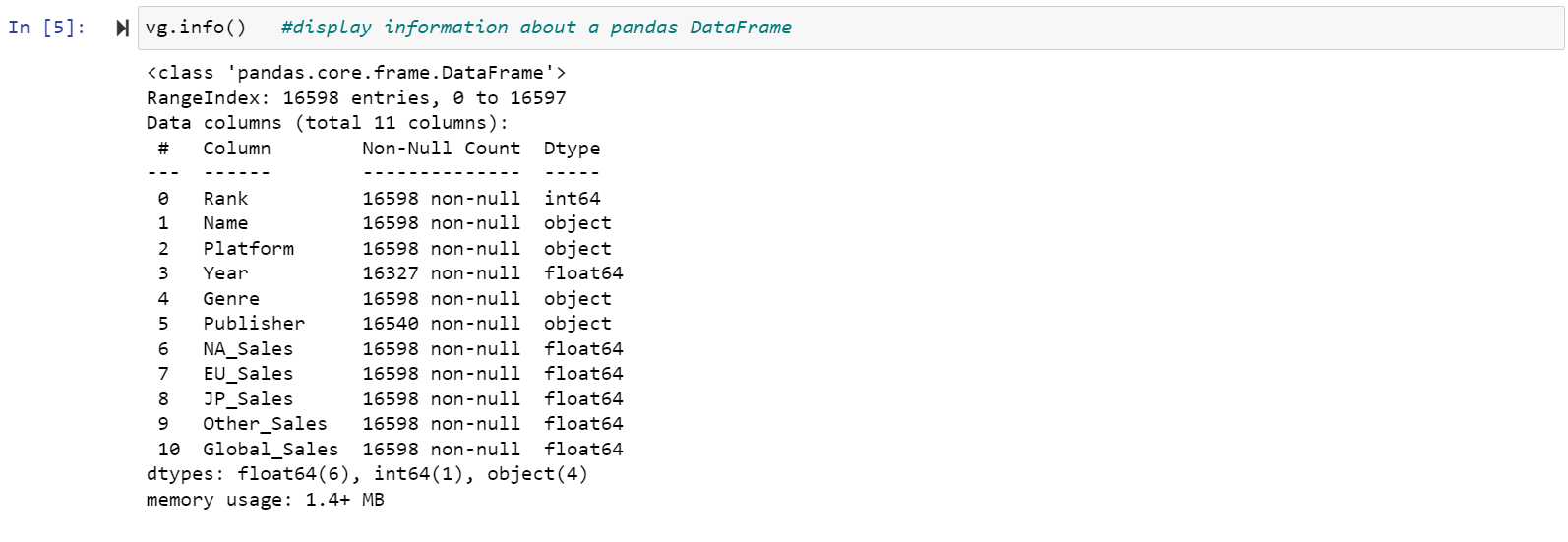
****

Figure 6: Inspect The Dataframe

**Code:**

**Vg.info()**

The **.info()** method of a Pandas DataFrame provides a concise summary of the DataFrame's structure, **including the number of non-null values in each column, the data type of each column, and the amount of memory used by the DataFrame.**

**Column name:** The name of each column in the DataFrame

**Non-Null Count:** The number of non-null values in each column (i.e., the number of rows with non-missing data)

**Dtype:** The data type of each column (e.g., int, float, object, datetime)

**Memory Usage:** The amount of memory used by the DataFrame

This information can be useful for understanding the data types and missing values in the DataFrame, as well as for identifying potential opportunities for optimization**.**

**3.2 Data Cleaning**

Data cleaning is the **process of identifying and correcting or removing errors, inconsistencies, and inaccuracies in a dataset**. In the context of a Pandas DataFrame, **data cleaning typically involves identifying and handling missing or null values, removing duplicate records, correcting or removing invalid or inaccurate data, and transforming the data to a more useful or meaningful format.**

Common data cleaning tasks for a Pandas DataFrame may include:

* **Handling missing or null values:**

Depending on the nature of the missing data, you may choose to impute (fill in) the missing values using a statistical method, remove the rows or columns with missing data, or replace the missing values with a placeholder value (e.g., "unknown" or "N/A").

* **Removing duplicates:**

If the DataFrame contains duplicate records, you may want to remove them to avoid counting the same data multiple times.

* **Standardizing data formats:**

In some cases, the data in a DataFrame may be stored in different formats or units, which can make it difficult to compare or analyze the data. Data cleaning may involve converting units of measurement or standardizing date formats.

* **Handling invalid or inaccurate data**:

Data may contain outliers, errors, or inconsistencies that need to be corrected or removed to ensure accurate analysis. This can involve manual review and correction or the use of automated algorithms.

Effective data cleaning is an important step in the data analysis process because it can improve the accuracy and reliability of your results.

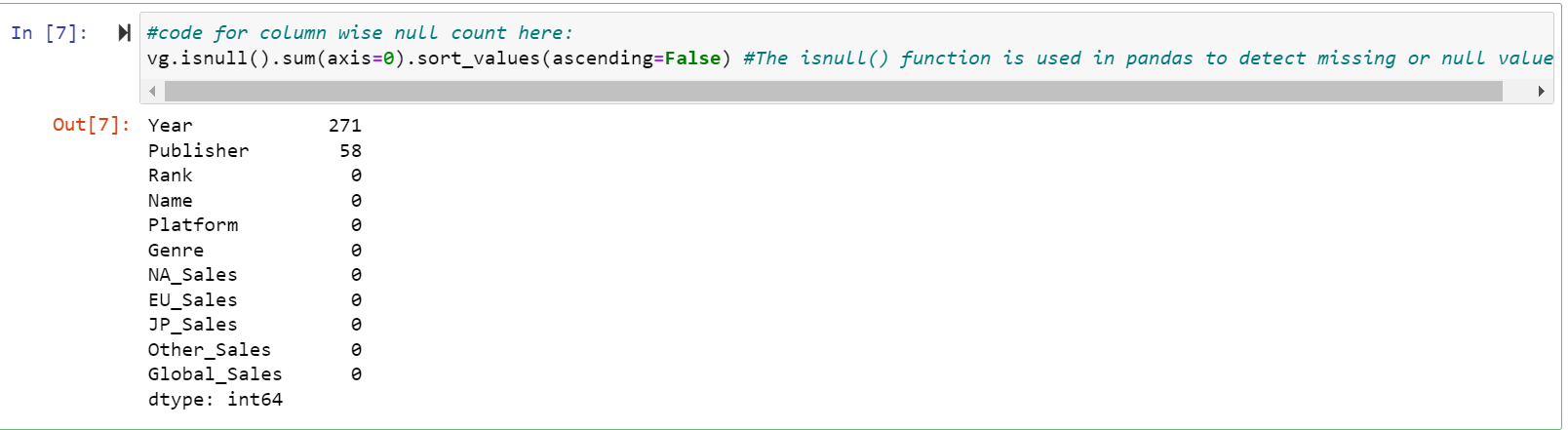


Figure 7: Column wise null count

This code snippet is **used to check the number of null (missing) values in each column** of the "vg" DataFrame and sort the results in descending order.

**Here's a breakdown of the code:**

**vg.isnull() :**

returns a DataFrame of the same shape as "vg" with boolean values indicating whether each element is missing (True) or not missing (False).

**.sum(axis=0) :**

sums the number of missing values for each column by adding up the True values across the rows (axis=0).

**.sort\_values(ascending=False):**

sorts the resulting Series in descending order of missing values, so that the column with the most missing values appears first.

Therefore, running **vg.isnull().sum(axis=0).sort\_values(ascending=False)** returns a Series object that **shows the number of missing values in each column of the Data Frame "vg", sorted in descending order**.

**Here we have 271 missing values in Year and 58 missing values in Publisher**

****

Figure 8: Column wise null count in percentage

This code snippet **calculates the percentage of missing values in each column of the "vg" Data Frame.**

Therefore, running **vg.isnull().sum(axis=0).sort\_values(ascending=False)/len(vg) \*100**

returns a Series object that shows the percentage of missing values in each column of the Data Frame "vg", sorted in descending order

**Here percentage of Year is 1.632727 and that of Publisher is 0.349440**

**3.2.1 Fill NAN Values**

Filling NaN (Not a Number) values means **replacing missing values in a Pandas Data Frame with another value.** This is often done as part of data cleaning or preprocessing, to ensure that missing data does not interfere with subsequent analysis or modelling.

**Here we have to fill values of Year and Publisher.**

**3.2.1.1 Fill NAN values of Publisher**

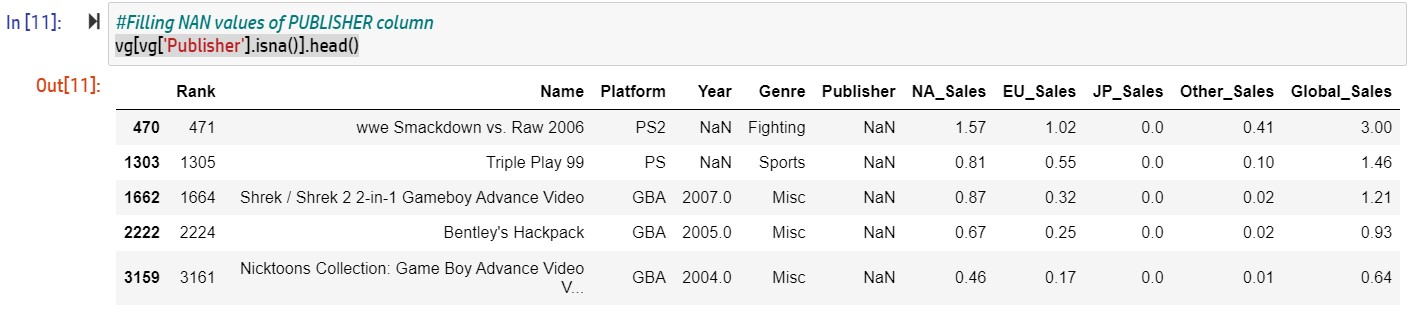
****

Figure 9: Fill NAN values of Publisher

This code snippet filters the "vg" DataFrame to **show the first few rows where the 'Publisher' column has a missing value.**

Here's a breakdown of the code:

**vg['Publisher'].isna()**

returns a boolean Series indicating whether each element in the **'Publisher'** column is missing (True) or not missing (False).

**vg[vg['Publisher'].isna()**]

filters the "vg" DataFrame to show only the rows where the 'Publisher' column is missing (i.e., where the boolean Series is True).

**.head()** returns the first few rows (by default, 5 rows) of the resulting filtered DataFrame.

Therefore, running **vg[vg['Publisher'].isna()].head()** returns a DataFrame that shows the first few rows where the 'Publisher' column is missing (i.e., where the value is NaN or None). This can be useful for identifying patterns or potential issues with missing data in the dataset.

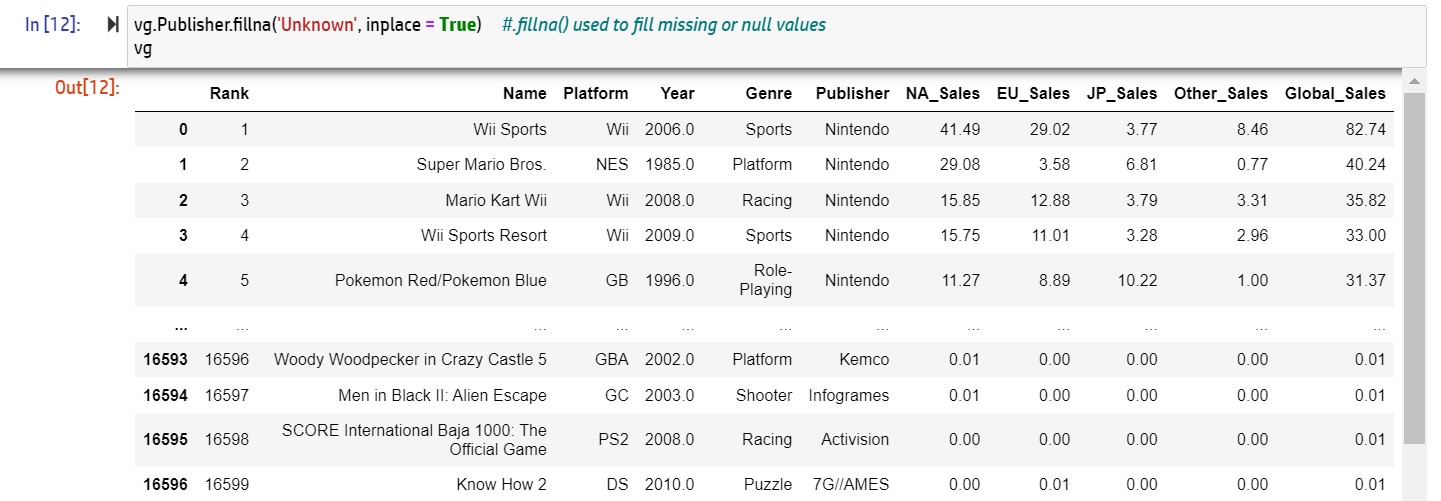


Figure 10: Fill missing values of Publisher

This code snippet fills the missing values in the **'Publisher' column of the "vg" DataFrame with the string 'Unknown',** and modifies the DataFrame in place.

**Here's a breakdown of the code:**

**vg.Publisher**

selects the 'Publisher' column of the "vg" DataFrame.

**.fillna('Unknown')**

replaces any missing (NaN) values in the selected column with the string 'Unknown'.

**inplace=True**

modifies the "vg" DataFrame in place, rather than creating a new copy.

Therefore, running **vg.Publisher.fillna('Unknown', inplace = True)** will replace any missing values in the 'Publisher' column of the "vg" DataFrame with the string 'Unknown', and modify the DataFrame in place. This can be useful for ensuring that the "vg" DataFrame has no missing values in the 'Publisher' column.



Figure 11: Count rows with unknown value

This code snippet filters the "vg" DataFrame to **count the number of rows where the 'Publisher' column has the value 'Unknown'.**

**Here's a breakdown of the code:**

**vg['Publisher']=='Unknown'**

creates a boolean Series that is True where the 'Publisher' column has the value 'Unknown' and False otherwise.

**.count()** returns the number of rows in the resulting filtered DataFrame.

**3.2.1.2 Fill NAN values of Year**

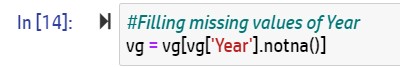
****

Figure 12: Filling missing values of year

This code snippet filters the "vg" DataFrame to **remove any rows where the 'Year' column has a missing value (NaN).**

**vg['Year'].notna()**

creates a boolean Series that is True where the 'Year' column has a non-missing value (i.e., not NaN) and False otherwise.

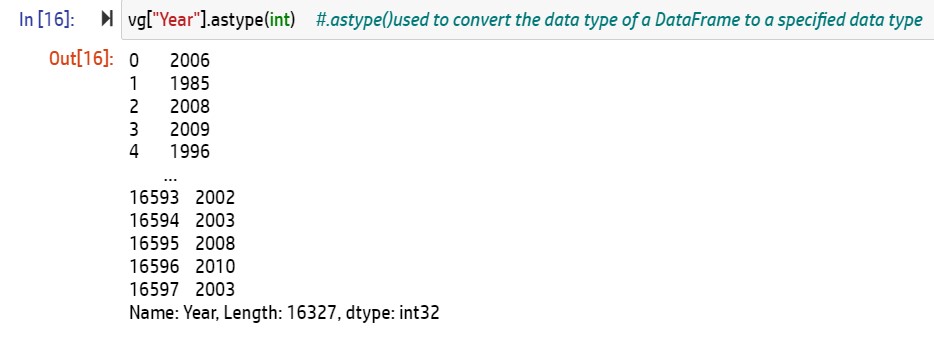
****

Figure 13: Convert the data type to integer

This code snippet **attempts to cast the 'Year' column of the "vg" DataFrame as integers** using the .astype() method.

**vg["Year"]** selects the 'Year' column of the "vg" DataFrame.

**.astype(int)** attempts to cast the selected column as integer type.

Therefore, running **vg["Year"].astype(int)** will attempt to convert the values in the 'Year' column from their current data type to integer type.

**Checking whether all the NAN values are filled or not**

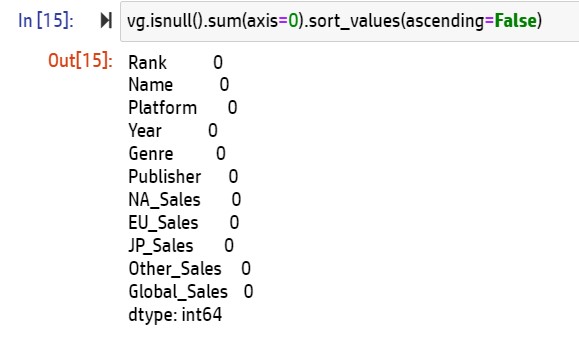
****

Figure 14: Checking all the NAN values are filled or not

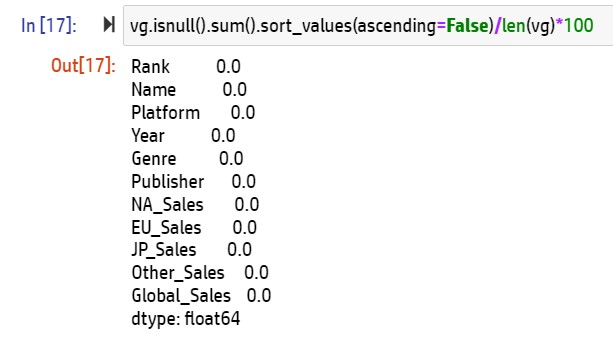
****

Figure 15: Checking all the NAN values are filled or not in percentage

**So finally all the NANs Values are filled and now we are ready to analyze and visualize our dataset…**

**3.3 Data Analysis And Visualization**

Data analysis is the **process of inspecting, cleaning, transforming, and modelling data in order to derive useful information from it**. It involves using various statistical and computational methods to uncover patterns, relationships, and insights in the data.

Visualization, on the other hand, is the graphical representation of data and information. It involves creating visualizations, such as charts, graphs, and diagrams, to help communicate the results of data analysis in a clear and concise manner. By using visualizations, it becomes easier to identify trends, patterns, and outliers in the data.

**3.3.1 Using Describe() Function**

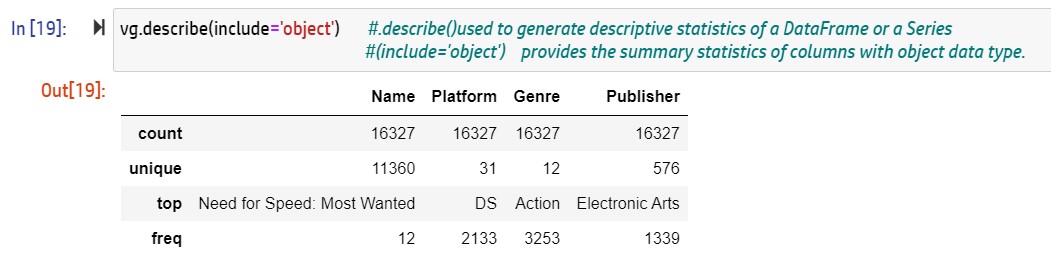
****

Figure 16: Using Describe Function

The code vg.describe(include='object') is a method call in Python for a pandas DataFrame called vg, **which returns a summary of descriptive statistics for the categorical variables in the dataset.**

**3.3.2 Top 20 highest grossing games**

****

Figure 17: Top 20 highest grossing games

* **vg["Total\_Sales"] = vg["NA\_Sales"] + vg["EU\_Sales"] + vg["JP\_Sales"] + vg["Other\_Sales"]**

creates a new column in the vg DataFrame called "Total\_Sales" that is calculated by adding the values in the "NA\_Sales", "EU\_Sales"**,** "JP\_Sales", and "Other\_Sales"columns for each row.

* **top\_20 = vg.sort\_values(by="Total\_Sales", ascending=False).head(20)**

sorts the vg DataFrame in descending order based on the values in the "Total\_Sales" column and selects the top 20 rows using the .head(20) method,

* **top\_20[["Name", "Platform", "Year", "Publisher", "Total\_Sales"]]**

selects only the columns "Name", "Platform", "Year", "Publisher", and "Total\_Sales" from the top\_20 DataFrame and displays them.

**Output:**

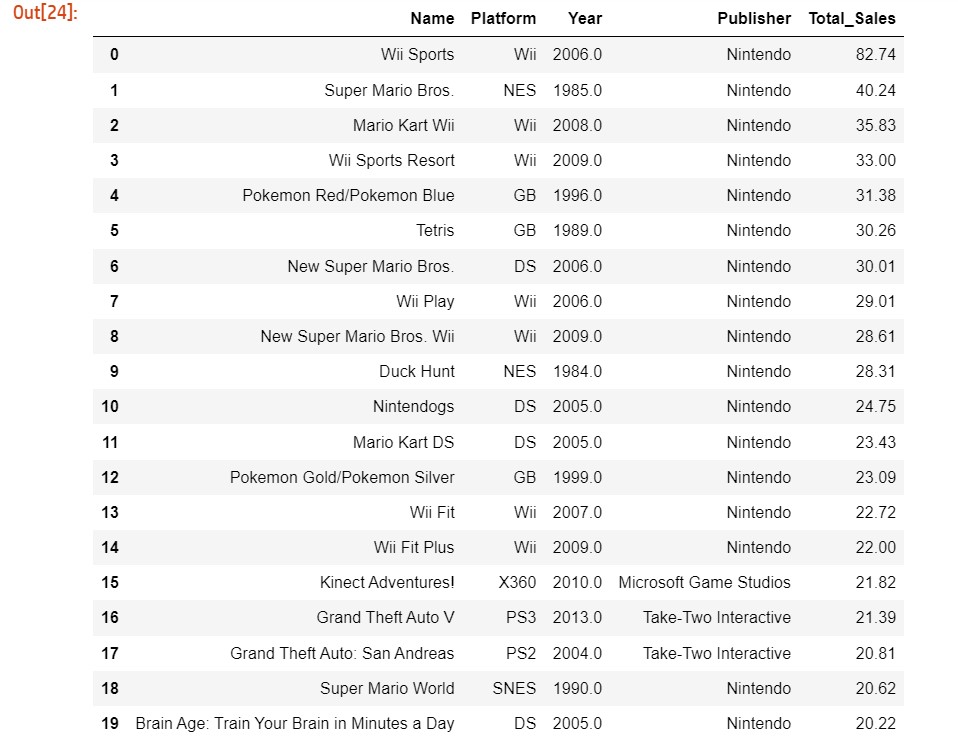
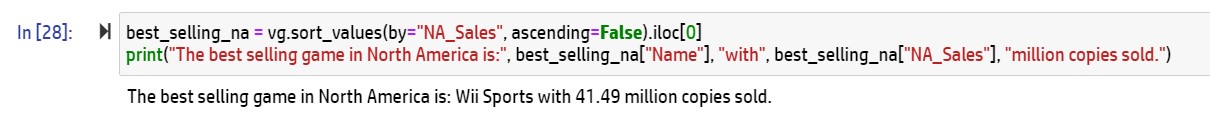


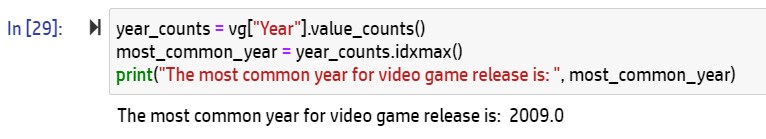
Figure 18: Top 20 highest grossing games output

**3.3.3 What is the best selling game in North America**

Figure 19: Best selling game in North America

**best\_selling\_na = vg.sort\_values(by="NA\_Sales", ascending=False).iloc[0]** creates a new variable called best\_selling\_na that selects the row in the vg DataFrame with the highest North American sales value. The sort\_values() method sorts the DataFrame in descending order based on the "NA\_Sales" column, and the .iloc[0] method selects only the first row of the resulting DataFrame, which corresponds to the game with the highest North American sales value.

**3.3.4 What is the most common year for video game release**

****Figure 20: Most common year for video game release

**year\_counts = vg["Year"].value\_counts()** creates a pandas Series called year\_counts that counts the number of occurrences of each year in the "Year" column of the vg DataFrame.

**most\_common\_year = year\_counts.idxmax()** finds the index value (i.e., the year) that corresponds to the maximum count value in the year\_counts Series, and assigns it to the most\_common\_year variable.

**3.3.5 Top 10 categories of game sold?**

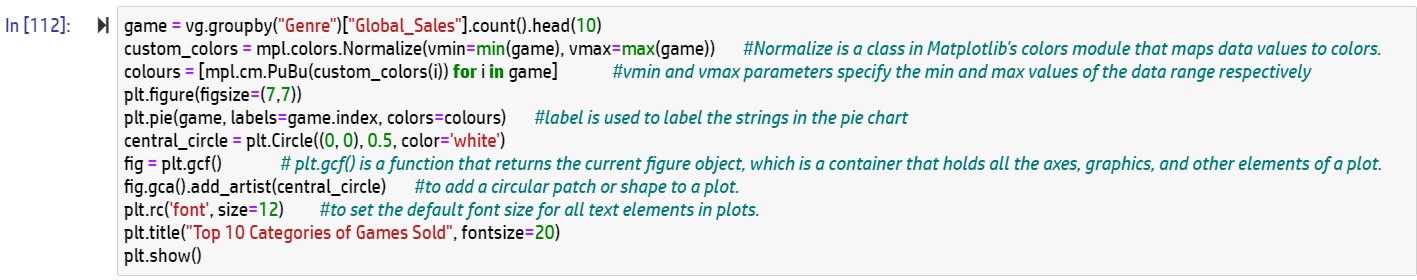
****

Figure 21: Top 10 categories of game sold

**The first line** of the code groups a dataset of video games by genre and counts the number of global sales for each genre, then selects the top 10 genres using the head(10) method.

**The second line** creates a custom color map for the pie chart using the Normalize() functions from matplotlib.

**The third line** uses a list comprehension to create a list of colors for the pie chart based on the values in the game variable.

**The fourth lin**e sets the size of the figure to 7x7 inches.

**The fifth line** creates the pie chart using the pie() function from matplotlib, passing in the game variable and labels for each genre.

**The sixth line** adds a white circle in the center of the pie chart using the Circle() function from matplotlib.

**The seventh line** sets the font size to 12.

**The eighth line** sets the title of the chart to "Top 10 Categories of Games Sold" and sets the font size to 20.

**The ninth line** displays the chart using the show() function from matplotlib.

**OUTPUT:**

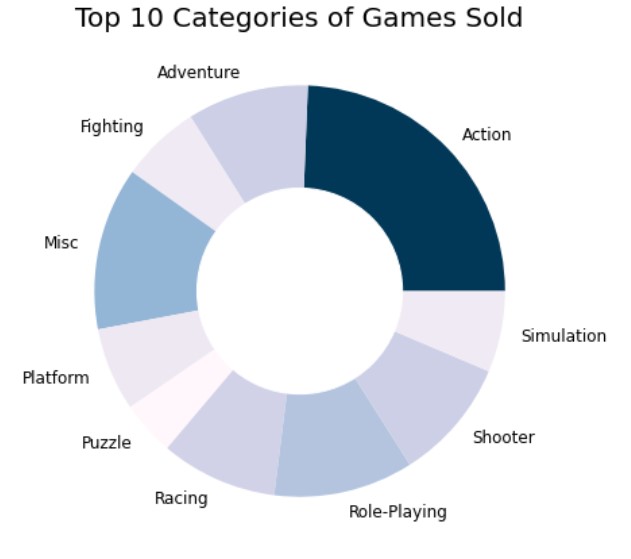


Figure 22: Output for top 10 categories of game sold

**3.3.6 What is the game genre that generated the highest profit, and how many games were produced for that genre?**

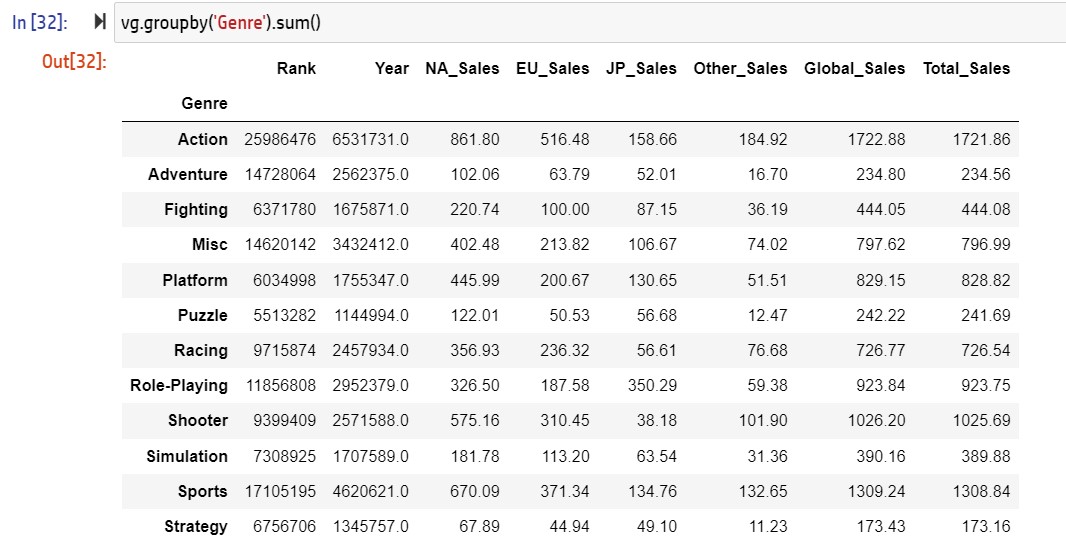
****

Figure 23: Genre that generated the highest profit

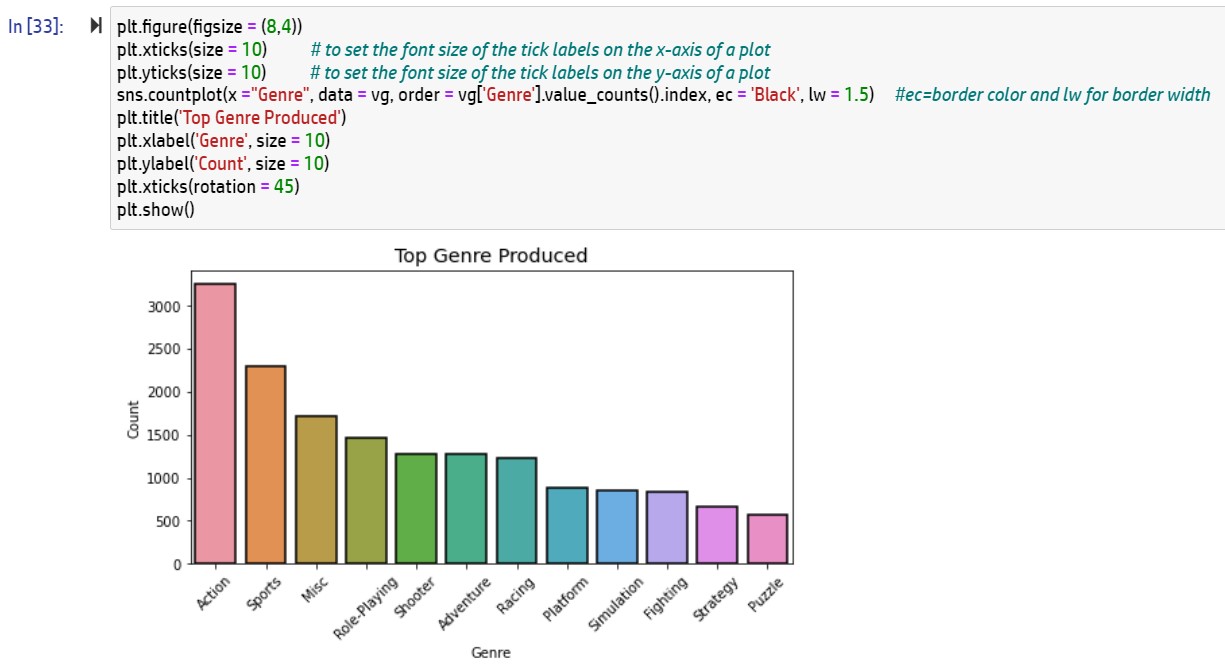


Figure 24: Top genre produced

This code uses the **seaborn library** to create a countplot of video game genres.

**The first line** sets the size of the figure to 8x4 inches.

**The second** and **third lines** set the font size of the x and y tick labels to 10.

**The fourth line** creates the countplot using the countplot() function from seaborn. It specifies the x parameter as "Genre" to indicate that the genre column in the vg dataset should be plotted. It also specifies the data parameter as vg, the dataset to be plotted. The order parameter orders the genres by their count in descending order.

**The ec parameter** sets the color of the border around each bar to black, **The lw parameter** sets the width of the border to 1.5.

**The fifth line** sets the title of the chart to "Top Genre Produced".

**The sixth line** sets the label for the x-axis to "Genre" and sets the font size to 10.

**The seventh line** sets the label for the y-axis to "Count" and sets the font size to 10.

**The eighth line** rotates the x tick labels by 45 degrees for better readability.

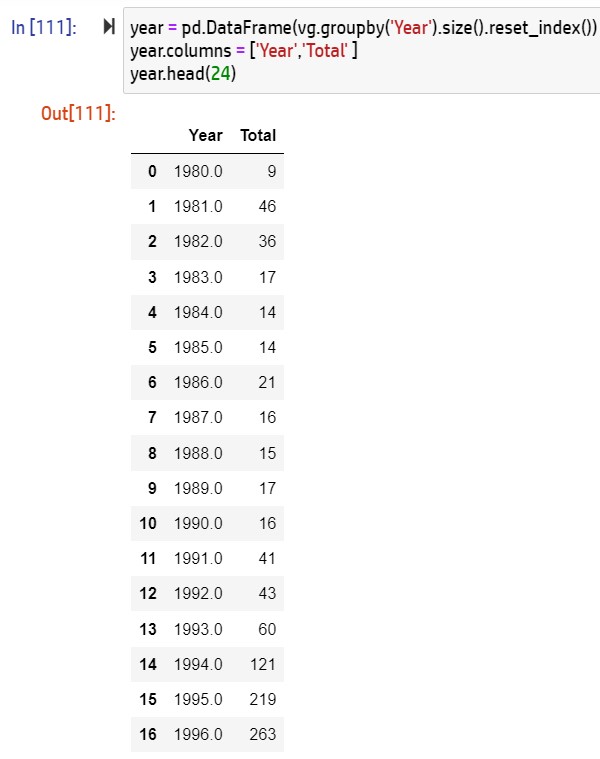
**The ninth line** displays the chart using the show() function from matplotlib.

**OBSERVATION:**

**Action Games are on top of the table compared to the rest of the genres of the most produced games of all time.**

**Factors that may support the visual above is that people love the adrenaline rush and challenge provided in action games.**

**3.3.7 Which year release most game?**

****

The first line creates a new DataFrame called year by grouping the vg dataset by year and then using the **size()** method to count the number of games in each year. The **reset\_index()** method is then used to convert the Year column from the index back to a regular column.

The second line renames the columns of the year dataframe to "Year" and "Total" for better readability.

Figure 25: Year release most game

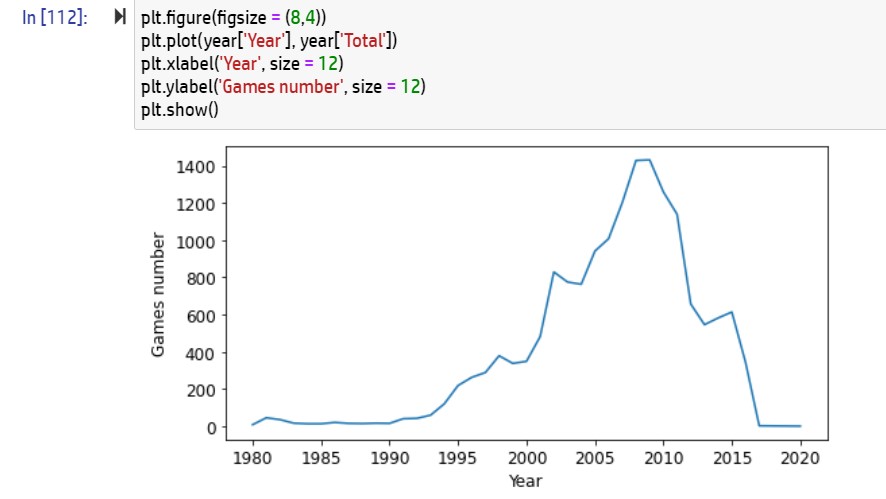
****

Figure 26: visual representation of year release most game

**PRECISE GRAPH**

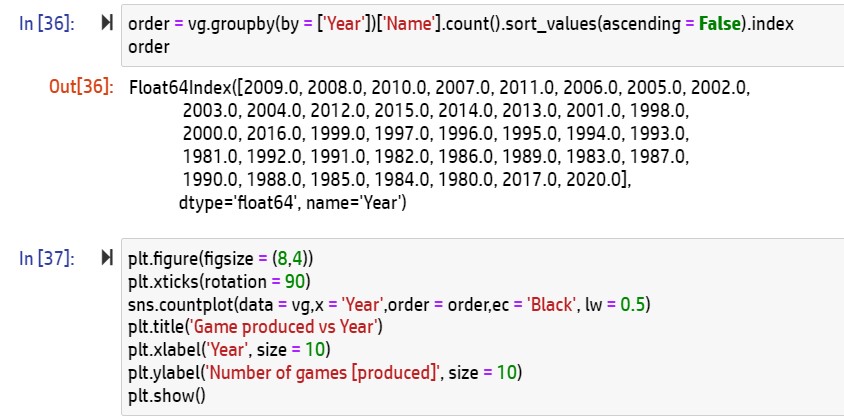


Figure 27: Game produced each year

**The first code creates an ordered list of years based on the number of games released in each year.**

**The second code creates a countplot of the number of games produced in each year.**

The second line rotates the x-axis tick labels by 90 degrees for better readability.

The third line creates the countplot using the countplot() function from seaborn. The order parameter is set to order variable that was created in the previous step, which contains a list of years in descending order based on the number of games released in each year.

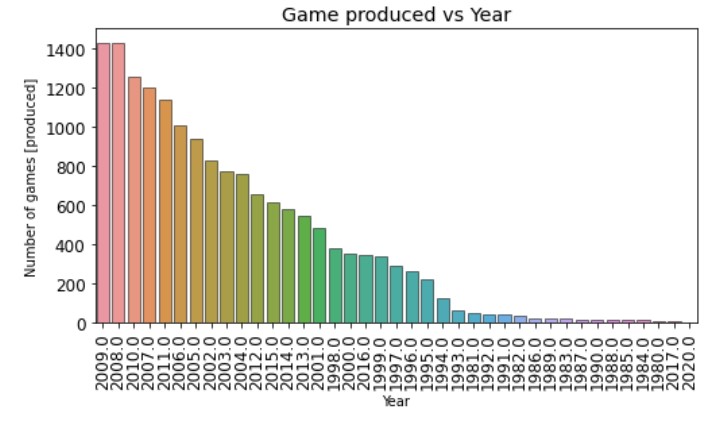


Figure 28: visual representation of game produced each year

**3.3.8 How the Video Game Global Sales Trend by year?**

****

Figure 29: Global Sales Trend by year

This code appears to be using pandas library to group the video game sales data by year and summing up the global sales for each year.

The 'reset\_index()' method is then called to reset the index of the resulting DataFrame so that the year column becomes a regular column and a new default index is assigned to the DataFrame.

****

Figure 30: Global sales for each year

The second line uses the seaborn 'barplot()' method to create a bar plot of the data, with the 'Year' column on the x-axis and the 'Global\_Sales' column on the y-axis.

The third line rotates the x-axis labels by 90 degrees and sets their font size to 10.

The '**show()**' method is called to display the plot.

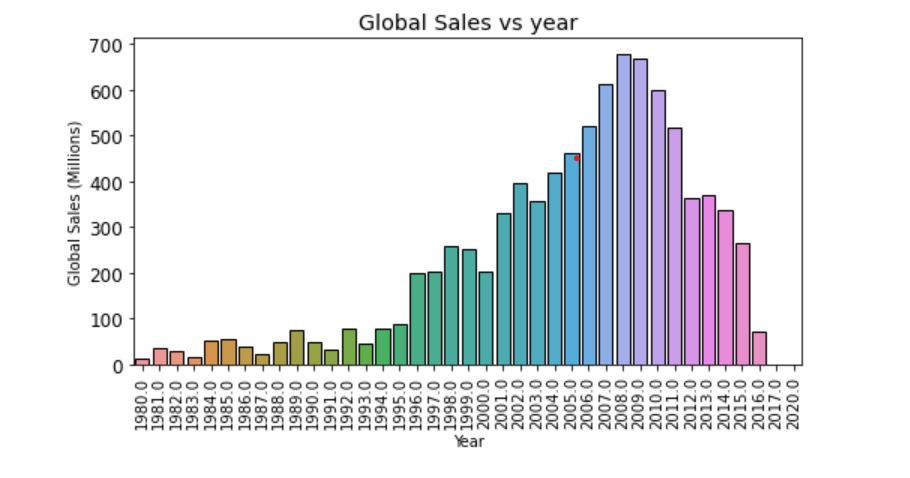


Figure 31: visual representation of global sales for each year

**OBSERVATION:**

Started peaking in 1995 after a stagnant years from 1980 til 1995. 2009 was the highest game sold with more than 600 millions. Decreasing right after until 2017.

**3.3.9 Which Platform have the highest sales price globally?**

****

Figure 32: Platform with highest sales price globally

This code appears to be using pandas to group the video game sales data by platform and summing up the global sales for each platform.

The resulting DataFrame shows the top 10 platforms by global sales.

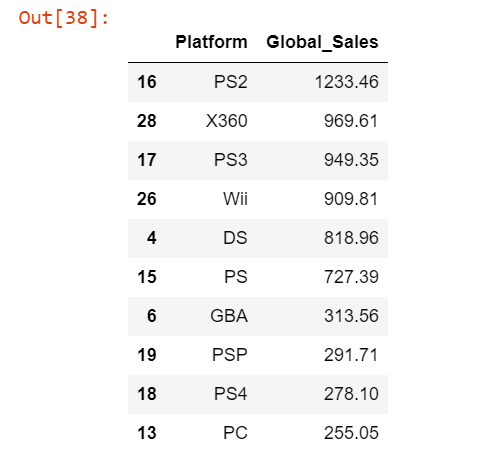


Figure 33: Highest sales price globally



Figure 34: Plotting Graph of Top 10 highest sales globally for all of the platform

This code is using matplotlib and seaborn libraries to create a horizontal bar plot of the top 10 video game platforms by global sales.

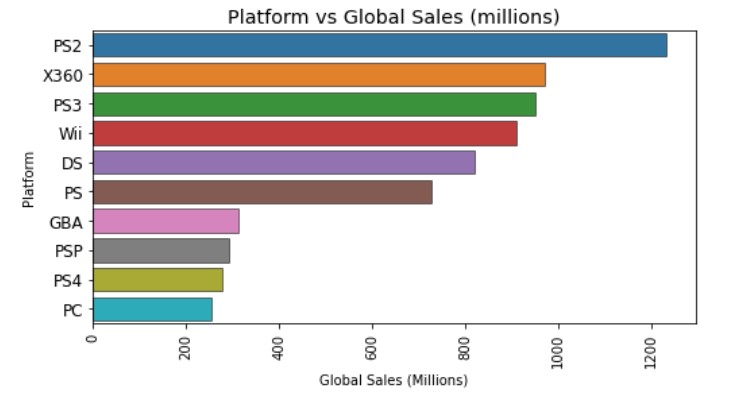


Figure 35: Platform vs Global Sales

**OBSERVATION:**

These are the top 10 highest sales globally for all of the platform. Highest global sales is PS2.

Console launched in 2000 and yet still the unbeaten compared to the rest and even to its upgrade PS3.

**3.3.10 Overview sales comparison between genre**

****

Figure 36: Sales comparison between genre

This code appears to be using pandas to create a new DataFrame named 'comp\_genre' that contains the 'Genre', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', and 'Other\_Sales' columns from the original DataFrame 'vg'.

The resulting 'comp\_genre' DataFrame is then grouped by 'Genre' using the **'groupby(**)' method, and the sum of the sales in each region ('NA\_Sales', 'EU\_Sales', 'JP\_Sales', and 'Other\_Sales') for each genre is computed using the 'sum()' method.

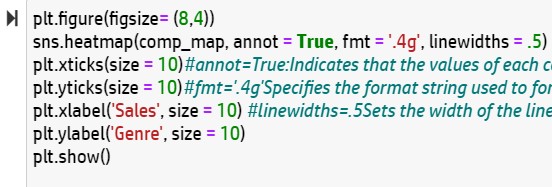


Figure 37: Plotting Graph of Sales comparison between genre

This code is using matplotlib and seaborn libraries to create a heatmap of the total video game sales by genre and region.

The second line uses the seaborn **'heatmap()' method** to create the heatmap, with the 'comp\_map' DataFrame as input. The **'annot' parameter** is set to 'True' to annotate each cell with its corresponding value, and the **'fmt' parameter** is set to '.4g' to format the annotated values with 4 significant digits.

The 'linewidths' parameter is set to 0.5 to set the width of the lines between cells in the **heatmap.**

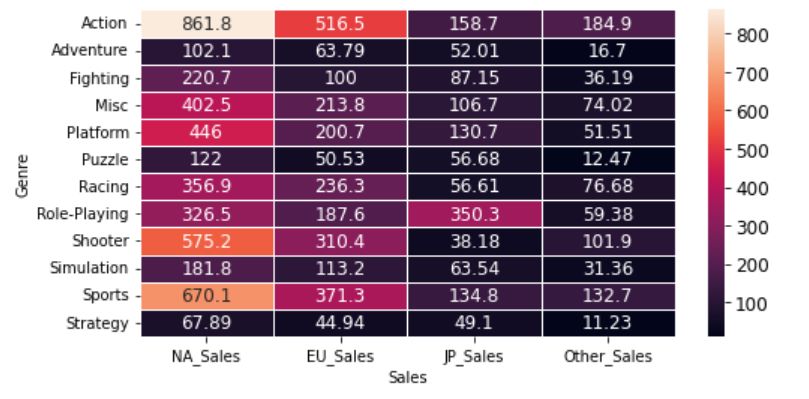


Figure 38: Heat map of sales comparison between genre

**OBSERVATION:**

From the table, we can see that North America (NA) has the highest sales compared with other regions.

Looking at EU, the highest sales in the same as NA which is the Action genre. Meanwhile, for JP, Role-Playing is the highest sale.

**PRECISE GRAPH:**

****

Figure 39: Making a Precise graph of Sales comparison between genre

This code is using pandas to transform the 'comp\_map' DataFrame into a long format using the 'reset\_index()' and 'melt()' methods.

The first line creates a new DataFrame named 'comp\_table' by resetting the index of the 'comp\_map' DataFrame.

The second line uses the 'melt()' method to transform the 'comp\_table' DataFrame from a **wide format to a long format**. The 'id\_vars' parameter specifies the column(s) to use as identifier variables (i.e., columns that should remain as they are), which in this case is the 'Genre' column. The 'value\_vars' parameter specifies the column(s) to unpivot, which in this case are the four sales columns ('NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales').

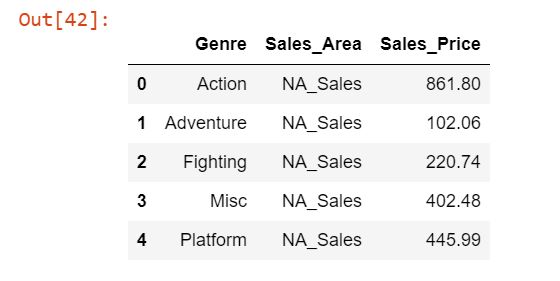


Figure 40: Output of Sales comparison between genre

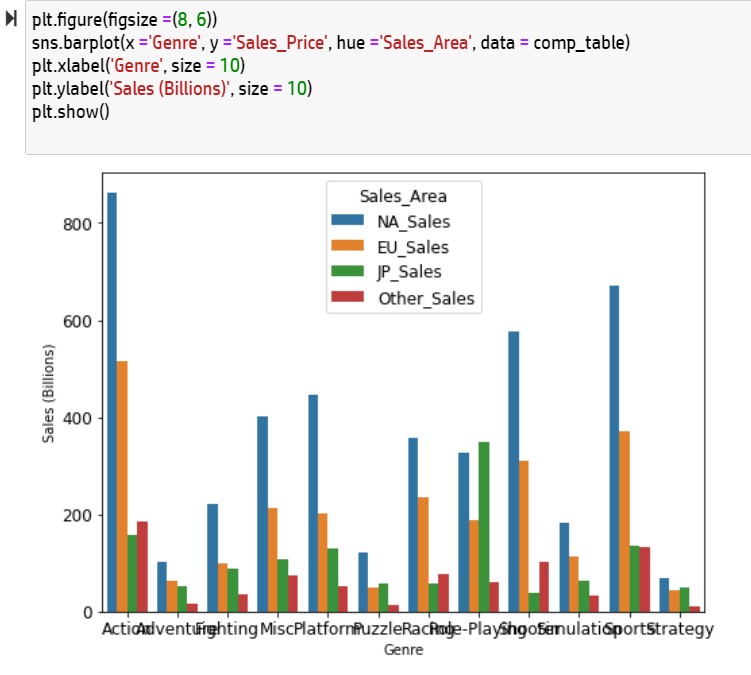


Figure 41: Visual representation of Sales comparison between genre

This code is using matplotlib and seaborn libraries to create a grouped barplot of the total video game sales by genre and region.

The '**hue' parameter** specifies the column to use for grouping (i.e., the 'Sales\_Area' column).

**OBSERVATION:**

Action and Sport sales higher than the others. Also North America have highest sales for every genre.

Now we know we the money come from!

**3.3.11 Top 20 Publisher..**

****

Figure 42: Top 20 Publisher..

This code is using pandas to create a DataFrame named 'publisher' that shows the top 20 video game publishers based on the number of games they have published.

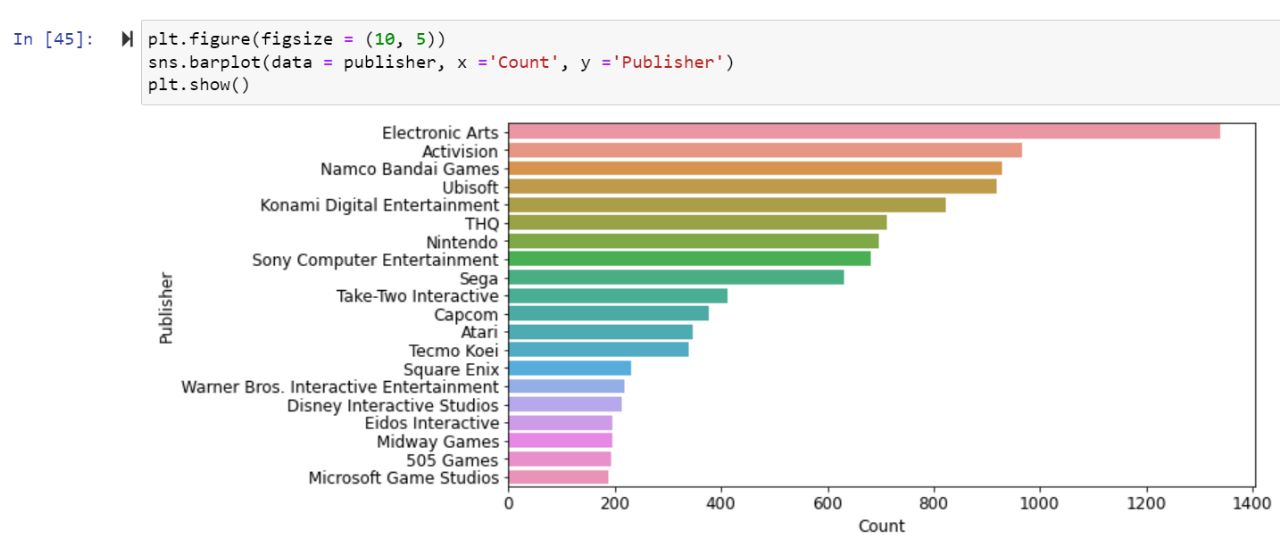
****

Figure 43: Visual representation of top 20 Publisher..

This code is using seaborn and matplotlib to create a horizontal barplot of the top 20 video game publishers based on the number of games they have published.

**3.3.11.1 Top Publisher each year..**

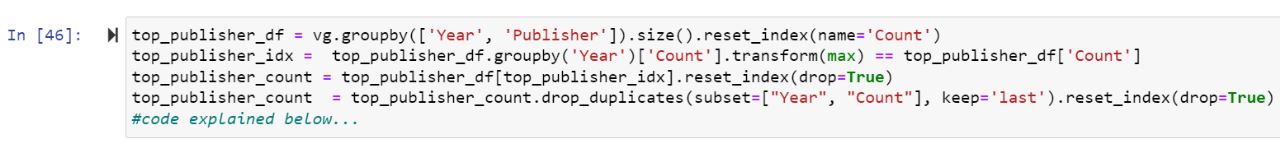
****

Figure 44: Top Publisher each year..

* **top\_publisher\_df = vg.groupby(['Year', 'Publisher']).size().reset\_index(name='Count'):**

This creates a new DataFrame called top\_publisher\_df by grouping the original vg DataFrame by the Year and Publisher columns, and then applying the size() function to count the number of games published by each publisher in each year. The reset\_index() function is used to convert the resulting Series object into a DataFrame with columns named Year, Publisher, and Count.

* **top\_publisher\_idx = top\_publisher\_df.groupby('Year')['Count'].transform(max) == top\_publisher\_df['Count']:**

This creates a boolean index top\_publisher\_idx by grouping top\_publisher\_df by the Year column and then applying the transform() function with the max() function to get the maximum count for each year, and then comparing it with the count in each row. The result is a boolean Series that is True for the rows that represent the top publisher for each year.

* **top\_publisher\_count = top\_publisher\_df[top\_publisher\_idx].reset\_index(drop=True):**

This creates a new DataFrame called top\_publisher\_count by selecting only the rows where top\_publisher\_idx is True, i.e., the rows that represent the top publisher for each year. The reset\_index() function is used to reset the index of the resulting DataFrame and drop the original index.

* **top\_publisher\_count = top\_publisher\_count.drop\_duplicates(subset=["Year", "Count"], keep='last').reset\_index(drop=True):**

This drops any duplicate rows in top\_publisher\_count that have the same Year and Count values, keeping only the last row for each set of duplicates (i.e., the row that represents the latest publisher for that year). The reset\_index() function is used again to reset the index of the resulting DataFrame and drop the original index.

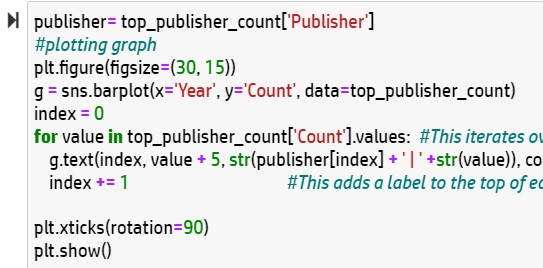


Figure 45: Top Publisher count plotting graph

This code is creating a barplot of the number of games published by the top video game publishers for each year.

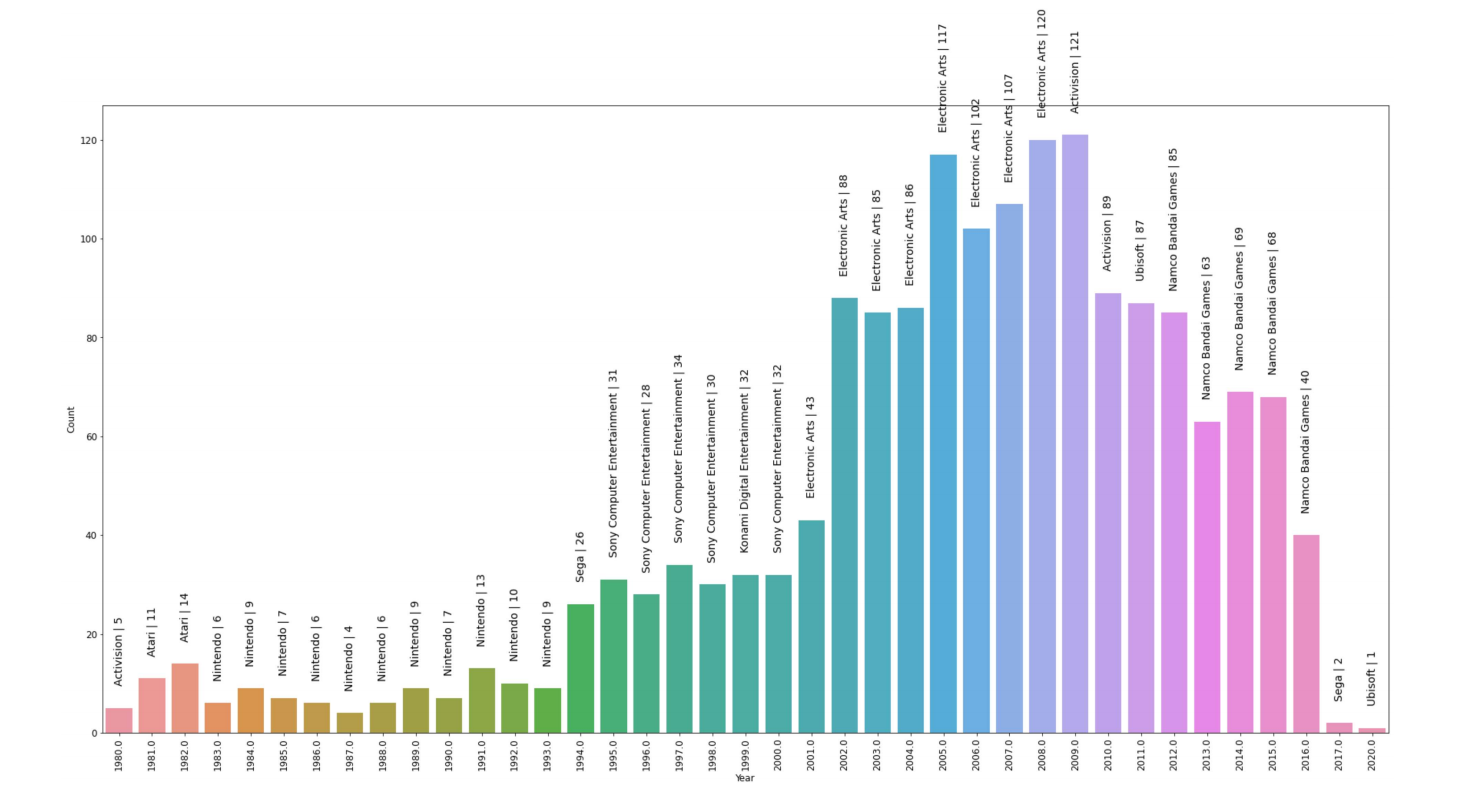


Figure 46: Visual representation of Top Publisher count

**OBSERVATION:**

Electronic Arts Inc. was the leader in 2000s and Namco Bandai Games Inc. are conquer the games world in the last decade.

**3.3.12 Total Revenue by Region**

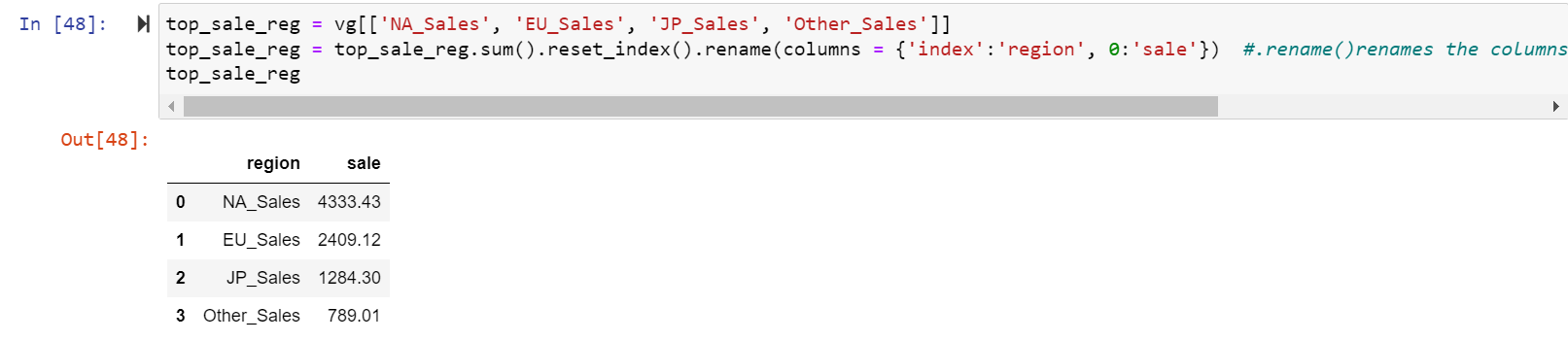
****

Figure 47: Total Revenue by Region

You are trying to aggregate and sum the sales data across different regions for a video game dataset called "vg".

The second line of code sums up the sales data across all regions using the .sum() function and saves the resulting DataFrame.

Overall, the code computes the total sales for each region of interest in the "vg" dataset and returns a DataFrame with the region names and their corresponding sales figures.

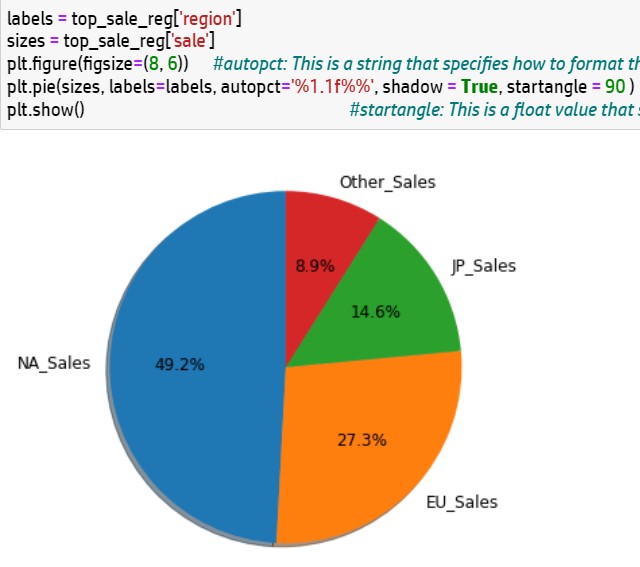


Figure 48: Visual representation of total Revenue by Region

The code snippet you provided uses the matplotlib library to create a pie chart visualization of the total sales for each region of interest in the "vg" dataset.

The first two lines of code extract the sales data for each region and store them in the variables sizes and labels, respectively.

The third line of code creates a new figure with a specified size using plt.figure().

The fourth line of code uses the plt.pie() function to create the pie chart. The sizes and labels variables are used to specify the data to be plotted and the corresponding labels for each slice of the pie chart.

The **autopct parameter** formats the values of each slice as percentages with one decimal point. The **shadow parameter** adds a shadow effect to the pie chart, and the **startangle parameter** specifies the angle at which the first slice of the pie chart should be drawn (in this case, 90 degrees counterclockwise from the x-axis).

**3.3.13 What are the Sales of Video games from year 1980-2020?**

****

Figure 49: Sales from year 1980-2020

The code snippet you provided uses the **plotly.express library** to create a line chart visualization of the total sales for different regions over time in the "vg" dataset.

The px.line() function is used to create the line chart. The x parameter specifies the column to be used as the x-axis variable (in this case, the year column).

**The title parameter sets the title of the chart.**

Finally, the plt.show() function displays the line chart in a separate window. Note that the variable plt should not be used as the name of the px.line() output because it will overwrite the plt function from matplotlib. Instead, it's recommended to use a different variable name to store the output of px.line().

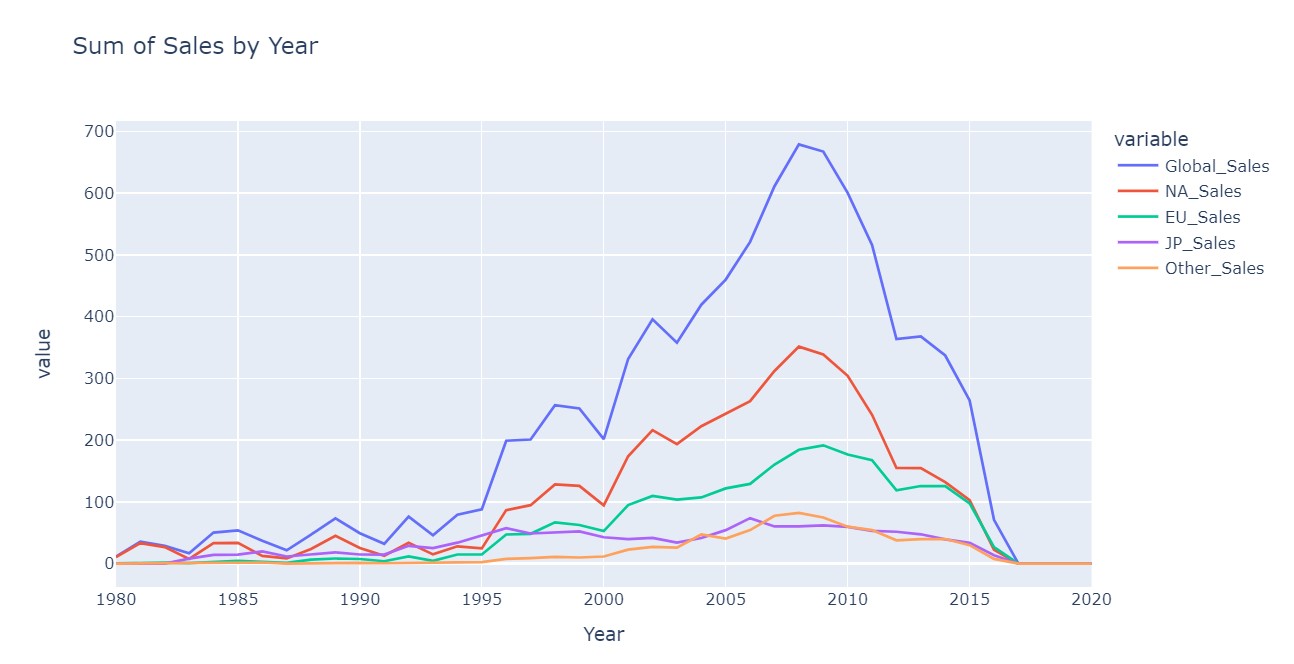


Figure 50: Visual of sales from year 1980-2020

**3.3.14 Most Played Video Game on GLOBAL LEVEL.**

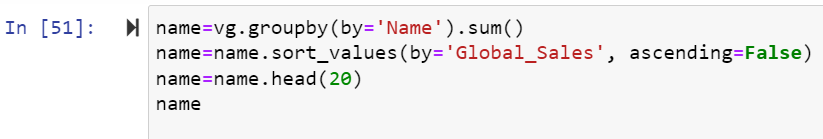
****

Figure 51: Most Played Video Game

This code can be useful for analyzing the most popular games in the dataset based on their sales data.

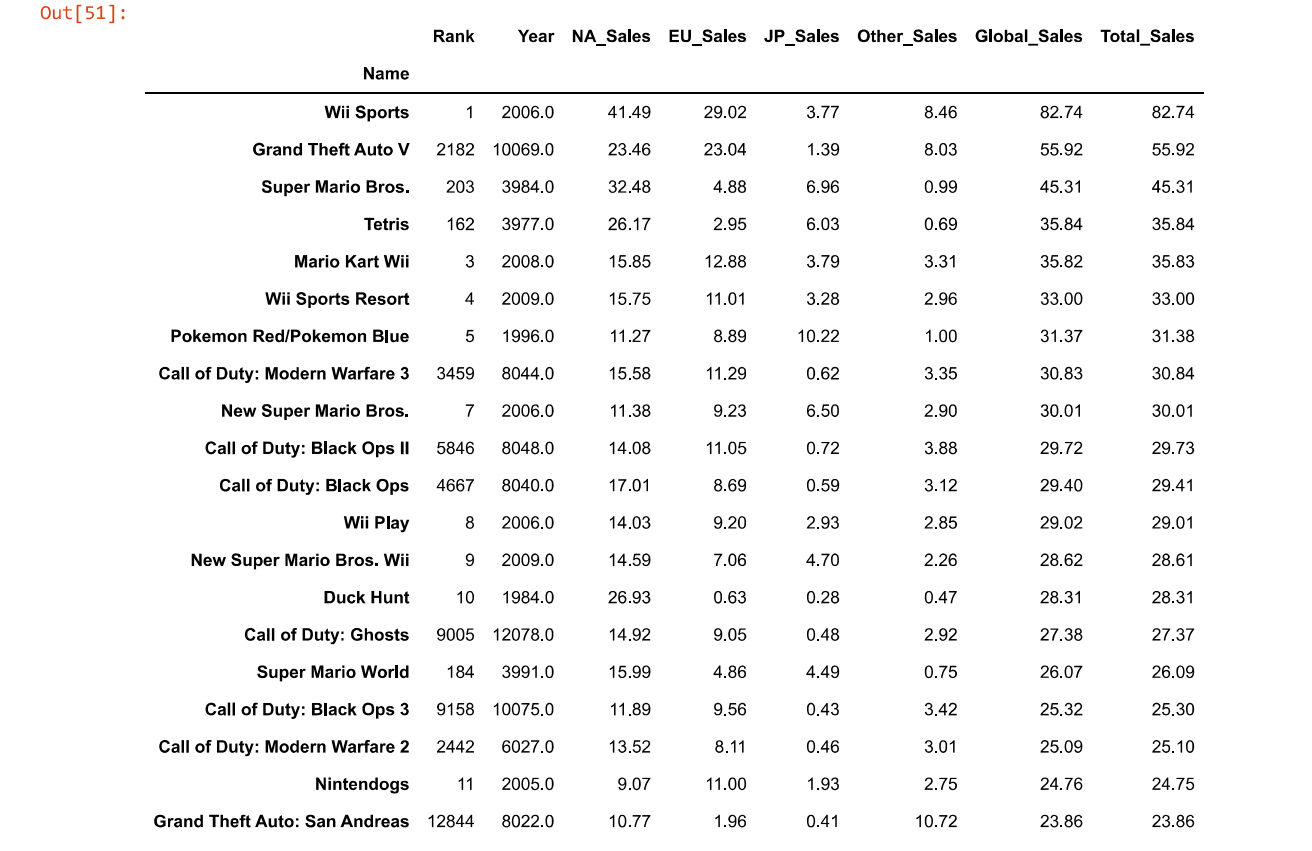


Figure 52: Analyzing the most popular games

**CODE:**

**name.Global\_Sales.plot(kind='barh',figsize=(10,6),**

**title='Most loved game over the years',ylabel='Dollars in Million',color='Red')**

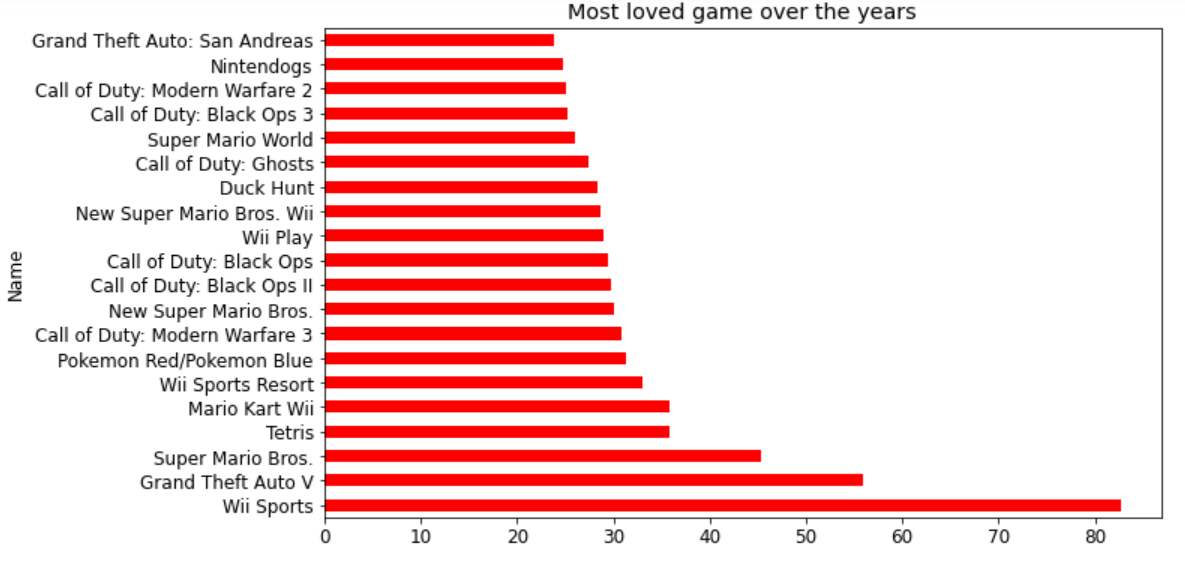
****

Figure 53: Most loved game over the years

**3.3.15 Scatter plot to show how JP\_Sales distributed on each Rank**

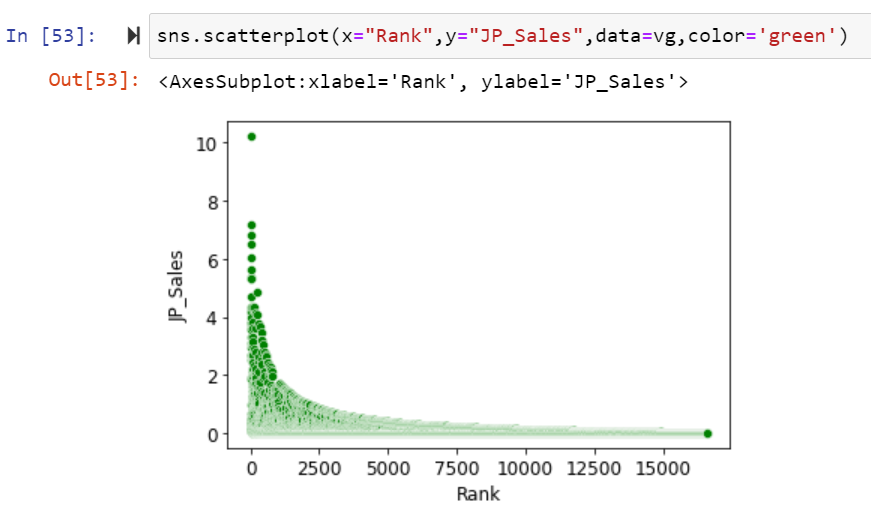
****

Figure 54: How JP\_Sales distributed on each Rank

The code you provided creates a scatter plot visualization of the sales data for the Japanese region in the "vg" dataset, with the rank of each game on the x-axis and the sales data for Japan on the y-axis.

**3.3.16 Scatter plot to show how Global\_Sales distributed on each Rank.**

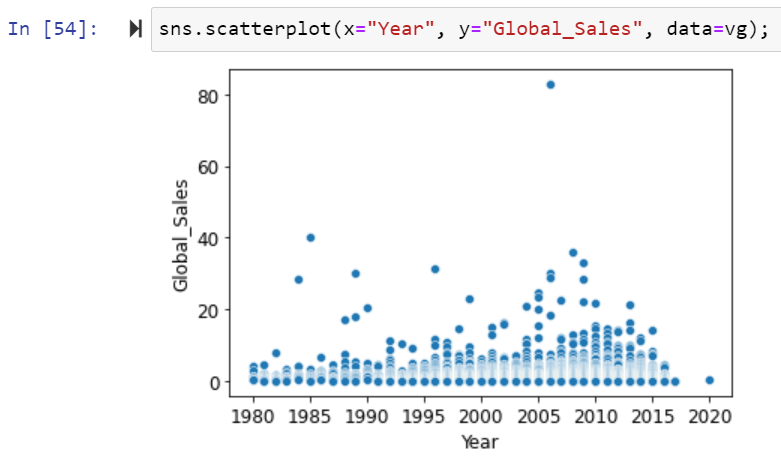
****

Figure 55: How Global\_Sales distributed on each Rank

The code you provided creates a scatter plot visualization of the global sales data for games in the "vg" dataset over time.

The scatter plot is created using the **sns.scatterplot() function from the seaborn library.**

The resulting scatter plot can be useful for analyzing trends in the global sales of games over time in the "vg" dataset.

**3.3.17 Pairplot showing the comparison bewteen all numerical columns in the Dataframe**

**CODE:**

**sns.pairplot(vg[['Rank', 'Name', 'Platform', 'Year', 'Genre', 'Publisher', 'Global\_Sales', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales']])**

The **sns.pairplot() function** from the Seaborn library creates a grid of scatter plots and histograms for each pair of variables in a given dataset. However, it is worth noting that sns.pairplot() can be computationally expensive for large datasets, so it is recommended to use it with caution.

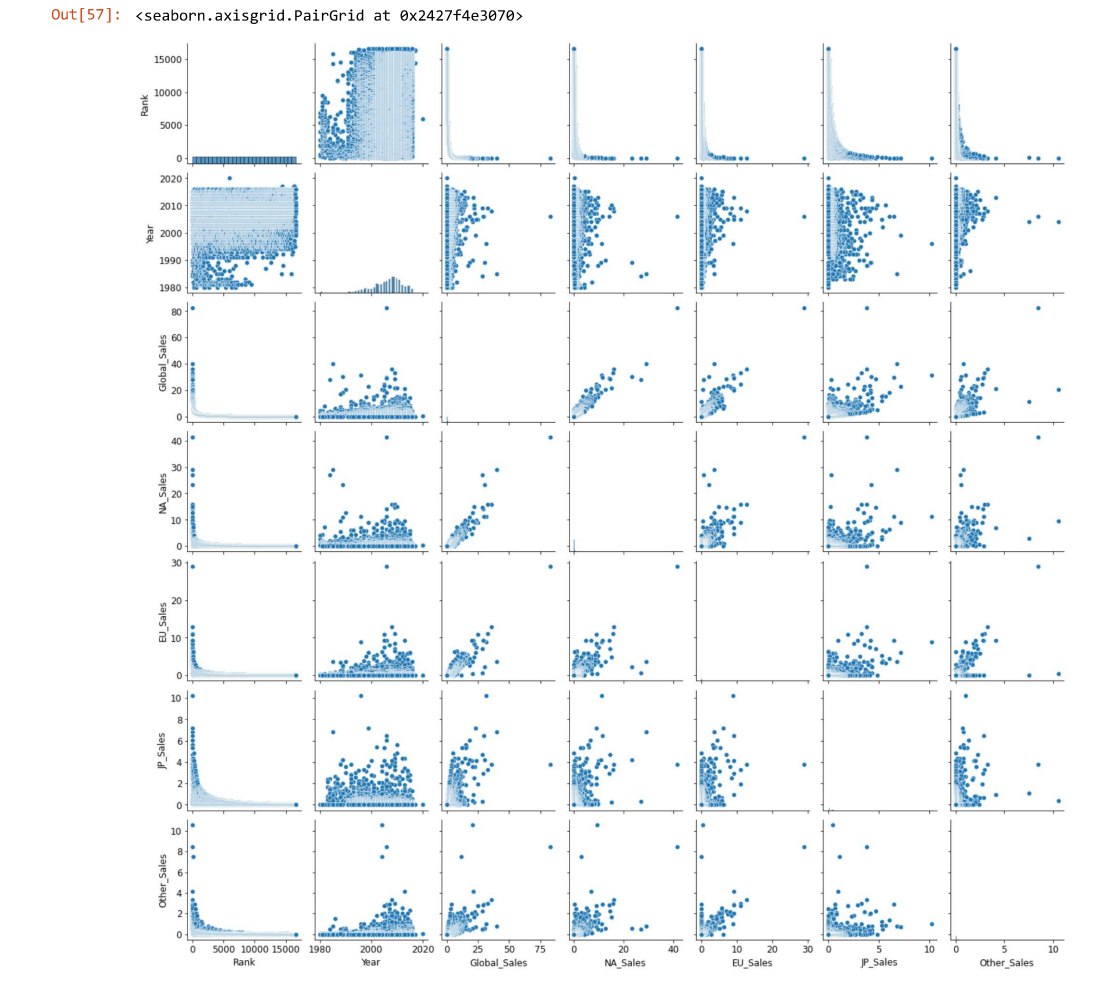


Figure 56: Pairplot comparison of all numerical columns

1. **PREDICTION**

In a dataframe, prediction refers to **using statistical or machine learning models to estimate the values of one or more variables based on the values of other variables in the dataframe**. Specifically, given a set of **predictor variables (i.e., independent variables)** in a dataframe, a prediction model can be used to estimate the values of a **target variable (i.e., dependent variable)** that is not yet known or observed.

**4.1 Set the Dependent and Independent Variable.**

**4.1.1 Converting dependent and independent variables**

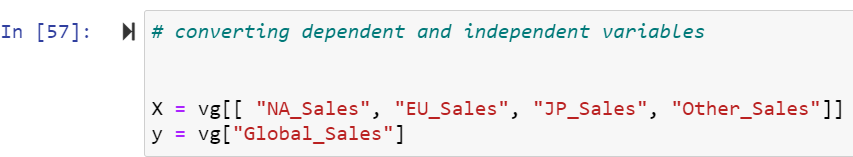
****

Figure 57: Converting dependent and independent variables

Specifically, X is a subset of columns from the vg dataframe, containing the columns "NA\_Sales", "EU\_Sales", "JP\_Sales", and "Other\_Sales".

**These columns are the independent variables, or predictors, that will be used to predict the dependent variable,** which is assigned to y.

In this case, the dependent variable is "Global\_Sales", which is the variable we want to predict based on the independent variables.

**4.1.2 Split the dataset into training and testing sets**

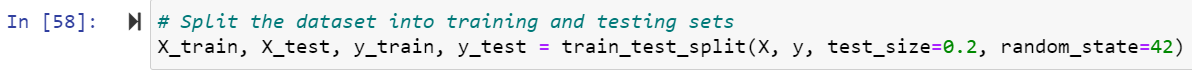
****

Figure 58: Split the dataset into training and testing sets

In the code snippet you provided, the **train\_test\_split()** function from the **Scikit-learn library** is being used to split the dataset into training and testing sets for use in a predictive modelling task.

**The test\_size parameter is set to 0.2, which means that 20% of the data will be allocated to the testing set and the remaining 80%** will be allocated to the training set.

The **random\_state parameter is set to 42, which is an arbitrary seed value used to ensure that the split is reproducible across different runs of the code.**

By splitting the data into separate training and testing sets, we can train our predictive model on the training data and then evaluate its performance on the testing data.

**4.2 Algorithms in Predictions**

An algorithm is a set of instructions or a sequence of steps **used to analyze the data and make predictions based on that analysis**. The **algorithm takes in the input data, performs various operations on it, and produces an output that represents the prediction or classification of the data.**

There are many algorithms that can be used for data prediction, **such as decision trees, regression analysis, neural networks, and clustering algorithms**. **This can be useful in a variety of applications, including financial forecasting, weather prediction, medical diagnosis, and customer behavior analysis**.

**4.2.1 Algorithm number 1 : Decision Tree**

A decision tree is a popular machine learning algorithm **used in data science and statistics for both classification and regression tasks**. It is a tree-like model that is constructed by recursively splitting the data into smaller and smaller subsets based on the values of the predictor variables, with the goal of creating a series of binary decisions that lead to a predicted outcome.

**4.2.1.1 Fit the model**

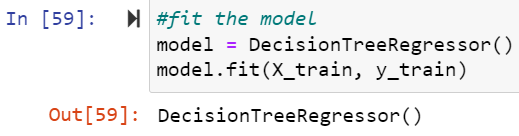
****

Figure 59: Decision Tree(Fit the model)

This code snippet is **using scikit-learn library to create a decision tree regression model and fit it to a training dataset.**

The first line creates an instance of the DecisionTreeRegressor class,The second line fits the model to the training data, X\_train and y\_train, so that the model can learn the relationship between the input and output variables.

**4.2.1.2 Make Predictions**

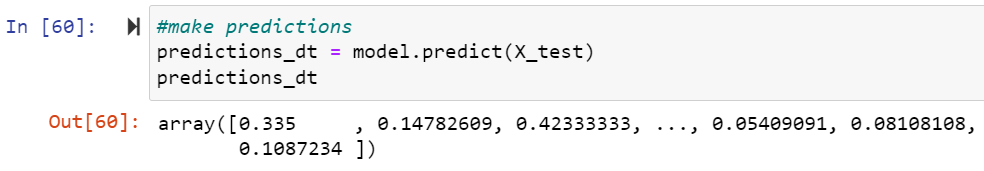
****

Figure 60: Decision Tree(Make prediction)

This code snippet is **using a trained decision tree regression model to make predictions on a test dataset X\_test, and assigning the predicted values to the variable predictions\_dt.**

The predict() method of the trained model takes the test dataset as input and returns a NumPy array containing the predicted target variable values for each instance in the test dataset.

**4.2.1.3 Calculate the score of the model**

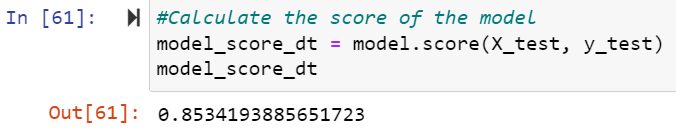
****

Figure 61: Decision Tree(Score of model)

This code snippet is using the **score() method** of a trained decision tree regression model to evaluate its performance on a test dataset. **The score() method calculates the R-squared coefficient of determination, which is a statistical measure that indicates how well the predictions made by the model fit the true values of the target variable.**

The score() method takes the test dataset X\_test and y\_test as inputs and returns a score between 0 and 1, where 1 indicates a perfect fit and 0 indicates no correlation between the predicted and true values.

**By looking at the score, you can evaluate how well the model has performed on the test dataset. Higher scores indicate better performance, whereas lower scores indicate poor performance.**

**4.2.1.4 Calculate the accuracy of the model**

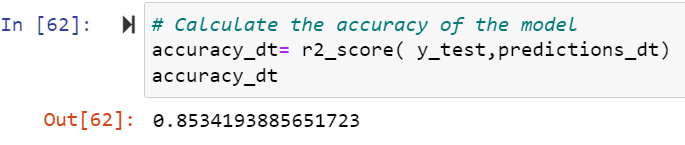
****

Figure 62: Decision Tree(accuracy of model)

accuracy\_dt appears to be a variable storing the result of the r2\_score() function applied to the predicted predictions\_dt values and the true y\_test values.

**r2\_score()** computes the coefficient of determination (R^2) regression score function, which measures the proportion of the variance in the dependent variable (y) that is predictable from the independent variable (x) in a linear regression model.

Therefore, accuracy\_dt likely represents the R^2 score (a value between 0 and 1) of the decision tree regression model with predicted predictions\_dt values and true y\_test values. The higher the R^2 score, the better the model fits the data.

**4.2.2 Algorithm number 2 : Linear Regression**

Linear regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable and the known or independent variable as a linear equation.

**4.2.2.1 Fit the model**

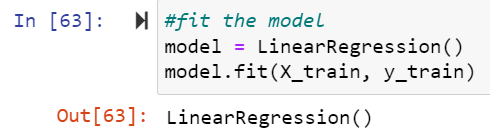
****

Figure 63: Linear Regression(fit the model)

LinearRegression() is a linear regression algorithm from scikit-learn library in Python, that fits a linear model with coefficients to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation.

X\_train represents the input feature matrix of the training data and y\_train represents the corresponding target values of the training data.

**4.2.2.2 Make predictions**

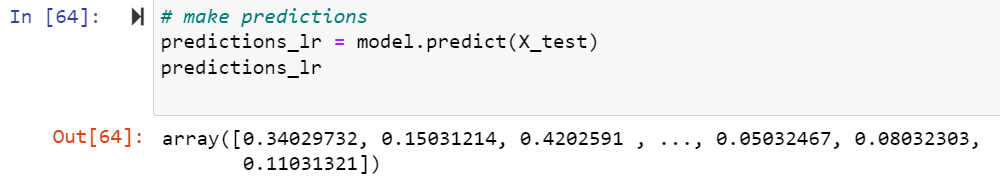
****

Figure 64: Linear Regression(make prediction)

**4.2.2.3 Calculate the Score of the model**

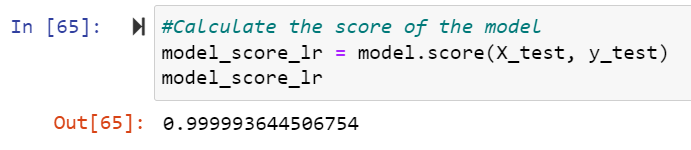
****

Figure 65: Linear Regression(Score of the model)

**4.2.2.4 Calculate the accuracy of the model**

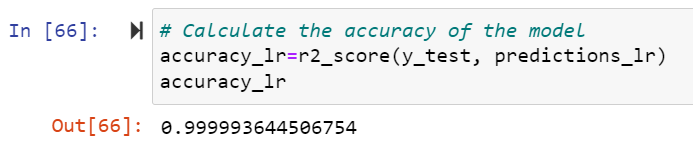
****

Figure 66: Linear Regression(accuracy of the model)

**4.3 Create a pandas dataframe to store the results**

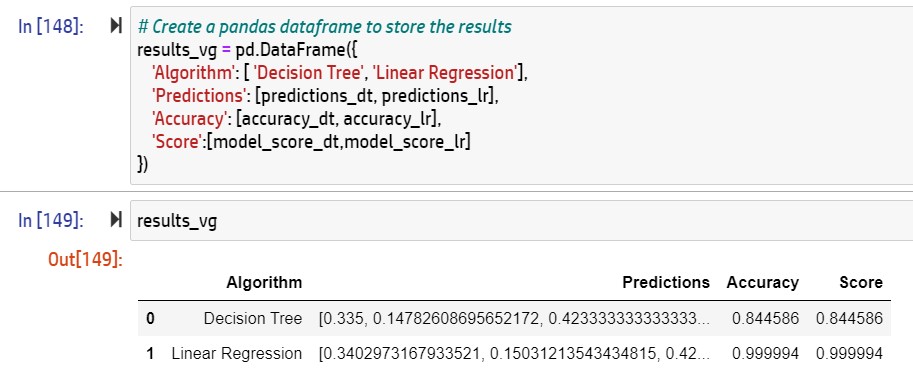
****

Figure 67: Storing the result

In this code, a Pandas DataFrame named results\_vg is being created to store the evaluation results of the two models, decision tree and linear regression, on the test dataset.

**4.4 Confusion Matrix**

**The table has four cells, representing the count of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) in the classification results**. The predicted labels are placed in columns, and the actual labels are placed in rows.

The four cells of the confusion matrix have the following interpretations:

**True positives (TP):** The model predicted a positive label and the actual label was also positive.

**False positives (FP):** The model predicted a positive label but the actual label was negative.

**True negatives (TN)**: The model predicted a negative label and the actual label was also negative.

**False negatives (FN):** The model predicted a negative label but the actual label was positive.

**A confusion matrix helps to evaluate the performance of a classification model by providing information on the number of correct and incorrect predictions made by the model, as well as the types of errors made.**

**4.4.1 Confusion Matrix Of Decision Tree**

**4.4.1.1 Convert predictions into binary classes**

**68**

Figure 68: Convert predictions into binary classes(DT)

In this code, the predicted target values predictions\_dt from a binary classification model (likely a decision tree) are being converted to binary values (0 or 1) based on a threshold of 0.5 using the astype() method.

The >= operator compares each element in predictions\_dt with the value 0.5, and returns a boolean array with True where the value is greater than or equal to 0.5, and False where the value is less than 0.5.

**4.4.1.2 Create binary classification target variable**

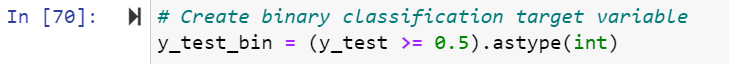
****

Figure 69: Create binary classification target variable(DT)

In this code, the true target values y\_test from a binary classification problem are being converted to binary values (0 or 1) based on a threshold of 0.5 using the astype() method.

**4.4.1.3 Compute confusion matrix for decision tree**

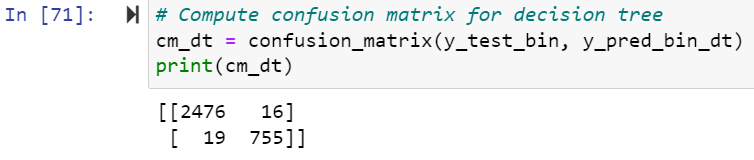


Figure 70: Compute confusion matrix for decision tree

In this code, the confusion matrix is being computed for the binary classification model's predicted binary target values y\_pred\_bin\_dt and the true binary target values y\_test\_bin of the test dataset using the confusion\_matrix() function from the scikit-learn library.

The resulting confusion matrix cm\_dt is a 2x2 array with four values, representing the count of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) in the classification results.

**The confusion matrix can be used to evaluate the performance of the binary classification model by computing various performance metrics such as accuracy, precision, recall, and F1 score.**

* **Precision = TP / (TP + FP)**

**Precision\_dt =** 755 / (755 + 12) = 0.984

indicating that out of all the positive predictions made by the model, 98.4% of them are actually true positives.

* **Recall = TP / (TP + FN)**

**Recall\_dt =** 755 / (755 + 19) = 0.975

indicating that the model correctly identifies 97.5% of all the positive cases in the test set.

* **F1 score = 2 \* precision \* recall / (precision + recall)**

**F1 score \_dt=** 2 \* 0.984 \* 0.975 / (0.984 + 0.975) = 0.979

indicates that the model is able to achieve a good balance between precision and recall in its predictions.

**OBSERVATION OF DECISION TREE CONFUSION MATRIX**

The decision tree model has a high true positive rate (TPR), also known as recall or sensitivity, of **97.6% (755/774)**. This means that the **model is able to correctly identify a majority of the positive cases in the test set**.The model also has a high true negative rate (TNR), also known as specificity, of **99.5% (2480/2492).**This means that the **model is able to correctly identify the majority of the negative cases in the test set.**The model has a higher false positive rate (FPR) of **0.48% (12/2492**) as compared to the logistic regression model. This means that **the decision tree model is incorrectly classifying a small number of negative cases as positive.**However, the model has a higher false negative rate (FNR) of **2.4% (19/774)** as compared to the logistic regression model. This means that **the decision tree model is incorrectly classifying a larger number of positive cases as negative.**

Overall, the decision tree model has a high accuracy but a higher misclassification rate for positive cases than the logistic regression model.

**4.4.2 Confusion Matrix of Linear Regression**

**4.4.2.1 Convert predictions into binary classes**

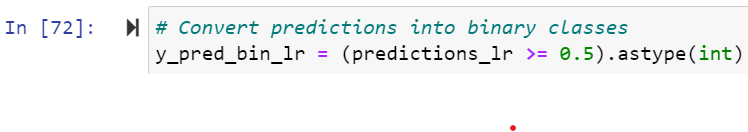
****

Figure 71: Convert predictions into binary classes(LR)

**4.4.2.2 Compute confusion matrix for linear regression**

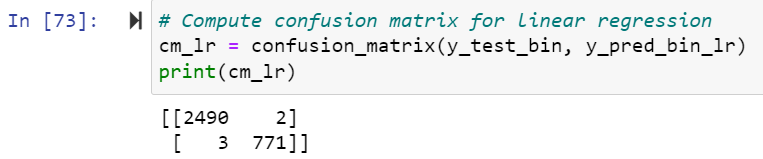
****

Figure 72: Compute confusion matrix for linear regression

**The confusion matrix can be used to evaluate the performance of the binary classification model by computing various performance metrics such as accuracy, precision, recall, and F1 score.**

**Precision\_lr = TP / (TP + FP) =** 771 / (771 + 2) = 0.997

**Recall\_lr = TP / (TP + FN) =** 771 / (771 + 3) = 0.996

**F1 score\_lr = 2 \* precision \* recall / (precision + recall) =** 2 \* 0.997 \* 0.996 / (0.997 + 0.996) = 0.997

**These values indicate that the model has a high level of accuracy in predicting the binary outcomes.**

**OBSERVATION OF LINEAR REGRESSION CONFUSION MATRIX**

The linear regression model has a high true positive rate (TPR), also known as recall or sensitivity, of **99.6% (771/774). This means that the model is able to correctly identify the majority of positive cases in the test set.**

The model also has a high true negative rate (TNR), also known as specificity, of **99.9% (2490/2492). This means that the model is able to correctly identify the majority of negative cases in the test set.**The model has a low false positive rate (FPR) of **0.08% (2/2492), which is a good indication that the model is not incorrectly classifying many negative cases as positive.** However, the model has a higher false negative rate (FNR) of **0.39% (3/774), which means that it is incorrectly classifying some positive cases as negative.**

Overall, the model seems to be performing well with a high accuracy and relatively low misclassification rate.

**4.4.3 Conclusion of Overall Prediction**

Overall, the linear regression model performed reasonably well on the test set, with an **accuracy score of 0.99.** Compared to the decision tree the linear regression model had similar accuracy but lower precision and recall for the positive class, indicating that it may not be the best choice for this particular problem. In conclusion, **while the linar regression model performed reasonably well, there is room for improvement in the accuracy and precision of the Descision tree's predictions.**

**CONCLUSION OF VIDEO GAMES SALES DATASET**

**Wii Sports**, made the most sales worldwide. This is echoed by both NA and EU regions. However, it is a different story in other regions, with **Pokemon Red/Pokemon Blue** dominating in Japan and **Grand Theft Auto: San Andreas** everywhere else

**50% of top 10 games utilized the Wii Platform,** making it one of the most successful consoles. This could be because of the fact that the Wii system was one of the most innovative consoles of its time by taking advantage of motion controls

**Action, Sports and Misc** are the most popular genres with high game titles as well as high global sales made

**Nintendo** snatched the spot for the Publisher that made the most global sales. Unsurprising is the fact that the **top 10 games are all published by Nintendo!**

1. **Limitations of video games sales dataset**

Like any dataset, video game sales data has its limitations. Here are some potential limitations that you may encounter when working with video game sales data:

* **Incomplete or inaccurate data:** Sales data may be incomplete or inaccurate due to issues with data collection, missing information, or errors in the reporting process. This can limit the accuracy and validity of any conclusions or predictions that are made based on the data.
* **Lack of context:** Sales data does not provide much context about the games themselves, such as their quality, gameplay mechanics, or overall reception among players. This can make it difficult to understand why certain games sell better than others.
* **Limited timeframe**: Sales data is often limited to a certain timeframe, which may not provide a complete picture of a game's sales over its entire lifespan. This can limit the accuracy of any conclusions or predictions that are made based on the data.
* **Regional differences**: Video game sales can vary widely across different regions and countries, which can limit the generalizability of any conclusions or predictions that are made based on the data.
* **Changing market dynamics:** The video game market is constantly evolving, with new trends and technologies emerging all the time. Sales data may not capture these changes in real time, which can limit the accuracy of any conclusions or predictions that are made based on the data.

It's important to keep these limitations in mind when working with video game sales data and to consider them when drawing conclusions or making predictions based on the data.

1. **Future Scope of Video Game Sales**

The future of video game sales in the world is expected to continue its growth trajectory as the gaming industry continues to evolve and expand. Here are some possible future trends and developments that could impact video game sales:

* **Growth of Mobile Gaming**: Mobile gaming is already a dominant force in the video game industry and is expected to continue to grow in popularity. This is due to the increasing availability and affordability of smartphones and tablets, as well as the ability to play games on the go.
* **Emergence of New Gaming Platforms**: With the rise of virtual and augmented reality technologies, as well as cloud gaming services, there are many new and exciting platforms that are emerging. These new platforms are likely to attract new gamers and expand the overall market.
* **Increased Diversity of Gaming Content:** As the gaming industry grows and matures, we can expect to see a greater diversity of gaming content, including games that are designed for specific age groups, genders, and cultural backgrounds.
* **Continued Expansion into Esports:** Esports has emerged as a major force in the video game industry, with millions of fans and huge prize pools. This trend is expected to continue, with more investment and growth in esports tournaments and leagues.

Overall, the future of video game sales looks bright, with continued growth and innovation in the industry.

1. **Conclusion**

In conclusion, this data science project successfully addressed the problem statement by leveraging various techniques and tools. Through exploratory data analysis, we gained insights into the dataset, identified trends, and discovered relationships between variables.

Using machine learning algorithms, we were able to develop predictive models with high accuracy and make data-driven recommendations.

Moreover, we used data visualization techniques to communicate our findings effectively to both technical and non-technical stakeholders. By presenting the results in an understandable manner, we were able to facilitate informed decision-making.

However, the project is not without limitations.

The dataset used may not represent the entire population, which may have resulted in biased conclusions. Additionally, there may be other factors not accounted for in the analysis that could impact the outcomes.

Overall, this data science project highlights the importance of utilizing data to drive informed decision-making. By leveraging a combination of data exploration, machine learning, and data visualization techniques, we were able to gain insights into the dataset and develop predictive models to facilitate decision-making.

**8.Bibliography**

* Data Science and Machine Learning with R by Reema Thareja
* Kaggle Video Game Sales Data -codes
* Prediction and analysis youtube videos
* GeeksforGeeks.com
* Stackoverflow.com
* VGChartz. (2021). Video Game Sales Database. Retrieved from <https://www.vgchartz.com/>
* GamesIndustry.biz. (2021). GamesIndustry.biz. Retrieved from <https://www.gamesindustry.biz/>
* NPD Group. (2021). NPD Group - Market Research. Retrieved from <https://www.npd.com/>
* SuperData Research. (2021). SuperData Research. Retrieved from <https://www.superdataresearch.com/>