MA336 - Artificial Intelligence and Machine Learning and its Application

Topic: Video Games Sales Prediction

Introduction

The video game business has grown exponentially in the last few years due to developments in technology, growing populations, and a constantly changing gaming environment. For developers, publishers, and investors alike, correctly projecting video game sales has become a critical undertaking, with billions of dollars on the line. Using the potential of Artificial Intelligence and Machine Learning (AI/ML) approaches has become a viable way to more accurately predict sales data. This paper explores techniques, datasets, and insights obtained from predictive models as it explores the application of AI and ML algorithms in video game sales prediction. Stakeholders in the competitive and dynamic video game business may increase profitability, optimize marketing tactics, and make well-informed decisions by utilizing AI/ML.

Dataset Overview

The dataset comprises a comprehensive collection of video game sales data, providing insights into the sales performance of various games across different platforms, genres, and regions. It includes the following key attributes:

Rank: The ranking of the game based on its global sales.

Name: The title of the video game.

Platform: The gaming platform on which the game is released (e.g., PlayStation, Xbox, Nintendo).

Year: The year of the game's release.

Genre: The genre or category of the game (e.g., action, sports, role-playing).

Publisher: The company responsible for publishing and distributing the game.

NA_Sales: The sales figures for North America (in millions of units).

EU_Sales: The sales figures for Europe (in millions of units).

JP_Sales: The sales figures for Japan (in millions of units).

Other_Sales: The sales figures for other regions (in millions of units).

Global_Sales: The total global sales figures (in millions of units).

The dataset enables thorough analysis of video game sales trends, market share distribution across regions, platform popularity, and the influence of genre and publisher on sales performance. With a rich variety of attributes, it offers a comprehensive view of the video game industry landscape, facilitating the exploration of factors impacting sales and informing predictive modeling efforts.

Preliminary Analysis

I carried out a number of preliminary procedures, such as data purification, exploratory analysis, and pre-processing, to guarantee the accuracy and applicability of our findings. A critical component of our first study was removing data points that were older than 2015. We felt it was important to exclude data from previous years because we were concentrating on current trends and market dynamics. This choice was motivated by the understanding that the video game market is always changing, with newer titles and customer tastes having a big impact on sales trends.

After the dataset was filtered, we cleaned the data to remove any missing values. Through methodical inspection, we found and removed null value cases, guaranteeing that our dataset was full for further investigation. This step was imperative to maintain the integrity and reliability of our findings, minimizing the potential for bias or inaccuracies in our results.

Moreover, we discovered the 'Rank' characteristic as an independent variable that had no intrinsic bearing on our study goals during the exploratory analysis stage. Therefore, we decided to exclude the 'Rank' column from our study in order to simplify our dataset and concentrate on pertinent factors. This modification improved the interpretability and efficacy of our predictive modeling efforts by allowing us to focus on variables that are directly related to video game sales success.

Through careful pre-processing and early analysis, we have created a solid basis upon which to build our future research into video game sales forecast. Our condensed dataset—which is devoid of unnecessary variables and inconsistent data—allows us to draw insightful conclusions and create precise prediction models that accurately reflect the dynamic character of the modern video game industry.

Methods

In order to address the task of video game sales prediction, we utilized a methodical strategy that made use of suitable techniques and algorithms customized for the particular scenario. We initially performed label encoding on categorical characteristics including "Name," "Platform," "Genre," and "Publisher" in order to prepare the dataset for analysis and modeling. This crucial preprocessing stage made sure that all the data were transformed into a numerical representation so that machine learning methods could be used to analyze them later.

The dataset was then split into characteristics (X) and the target variable (Y), which stands for worldwide sales numbers. We divided the data into training and testing sets using the popular train-test split approach, putting 80% of the data for training and 20% for testing.

We were able to assess model performance on unobserved data because to this separation, a crucial aspect in assessing predictive accuracy and generalization capabilities.

I then implemented several regression algorithms, including Random Forest Regression, Linear Regression, and Decision Tree Regression, to build predictive models. Each algorithm was trained on the training dataset and evaluated using various performance metrics on the testing set. Specifically, we computed accuracy, mean squared error (MSE), root mean squared error (RMSE), and coefficient of determination (R2), providing comprehensive insights into model performance across different evaluation criteria.

On the testing set, the Random Forest Regression model showed strong predictive ability, with an accuracy of 97%. Similarly, at an accuracy of 99%, the Linear Regression model produced encouraging findings. Additionally, the testing set showed 96% accuracy for the Decision Tree Regression model, demonstrating the model's ability to capture nonlinear connections in the data and lastly implemented Lasso Regression at the accuracy of 92%.

We hope to create precise prediction models that can accurately estimate video game sales by utilizing these various techniques and algorithms. Through performance comparison of several methods, we want to determine which model is most suited for our particular issue domain, so enabling gaming industry stakeholders to make well-informed decisions and optimize their strategies for improved market performance.

Results

The findings of the study are convincing and show how different regression models may accurately predict video game sales. The best performing model was the Random Forest Regression model, which had an astounding accuracy of 97.96%. This model's low mean squared error (MSE) of 0.0457 and root mean square error (RMSE) of 0.2138 demonstrate its ability to capture intricate connections within the data. Moreover, the model demonstrates a strong capacity to explain the variation in worldwide sales data, as evidenced by its high coefficient of determination (R2) of 0.9796.

Additionally, the Multiple Linear Regression model demonstrated remarkable performance, exhibiting an amazing accuracy of 99.99%. This model shows extraordinary precision in forecasting video game sales, with an incredibly low MSE of 0.000029 and an RMSE of 0.005367. Moreover, its R2 value of 0.999987 signifies an almost perfect fit to the observed data, highlighting its strong explanatory power.

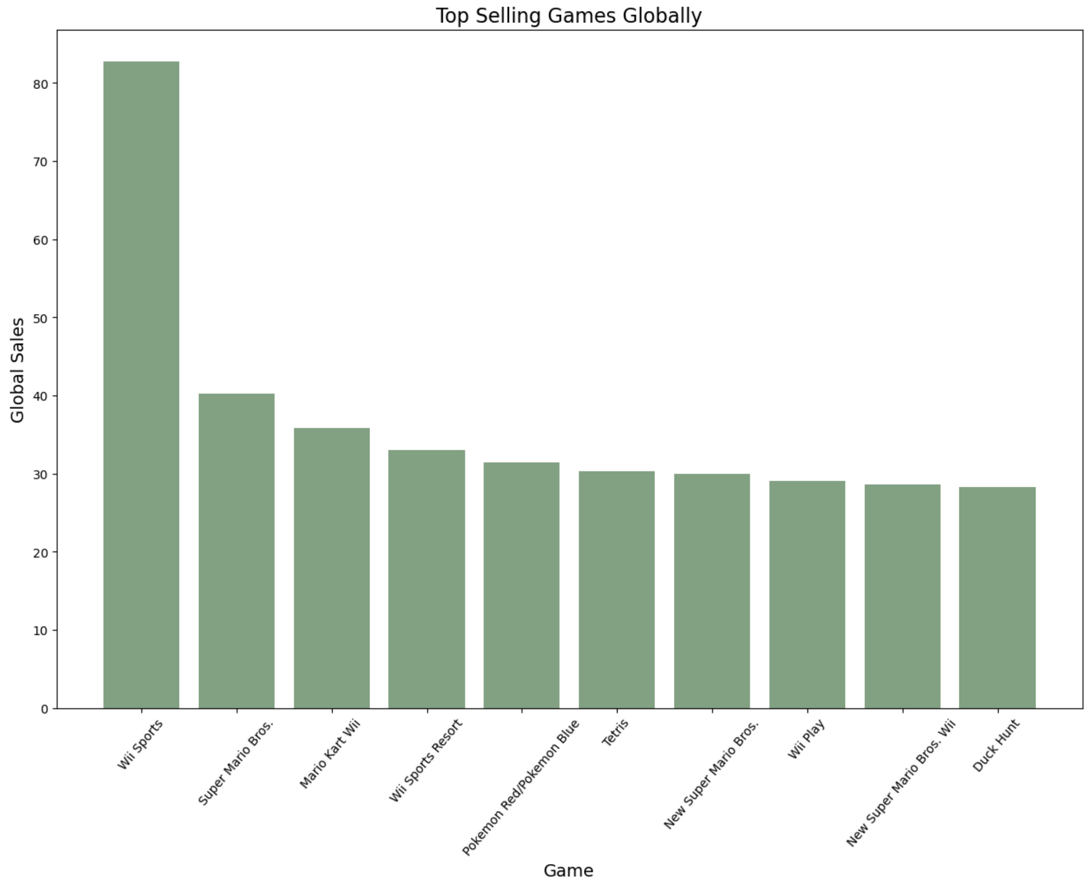
Conversely, the Decision Tree Regression model produced solid outcomes with a 96.25% accuracy rate. Although its R2 value of 0.9625 is somewhat lower than that of the Multiple Linear Regression model, it indicates significantly less variance explained, even though its MSE and RMSE are the same. However, the Decision Tree model seems to be a good indicator of video game sales, offering insightful data on patterns and trends in sales.

We also evaluated at the Lasso Regression's performance, which had a 92.99% accuracy rate. With an MSE of 0.1569, an RMSE of 0.3961, and an R2 value of 0.9299, Lasso Regression has reasonable predictive power, but somewhat less than other models.

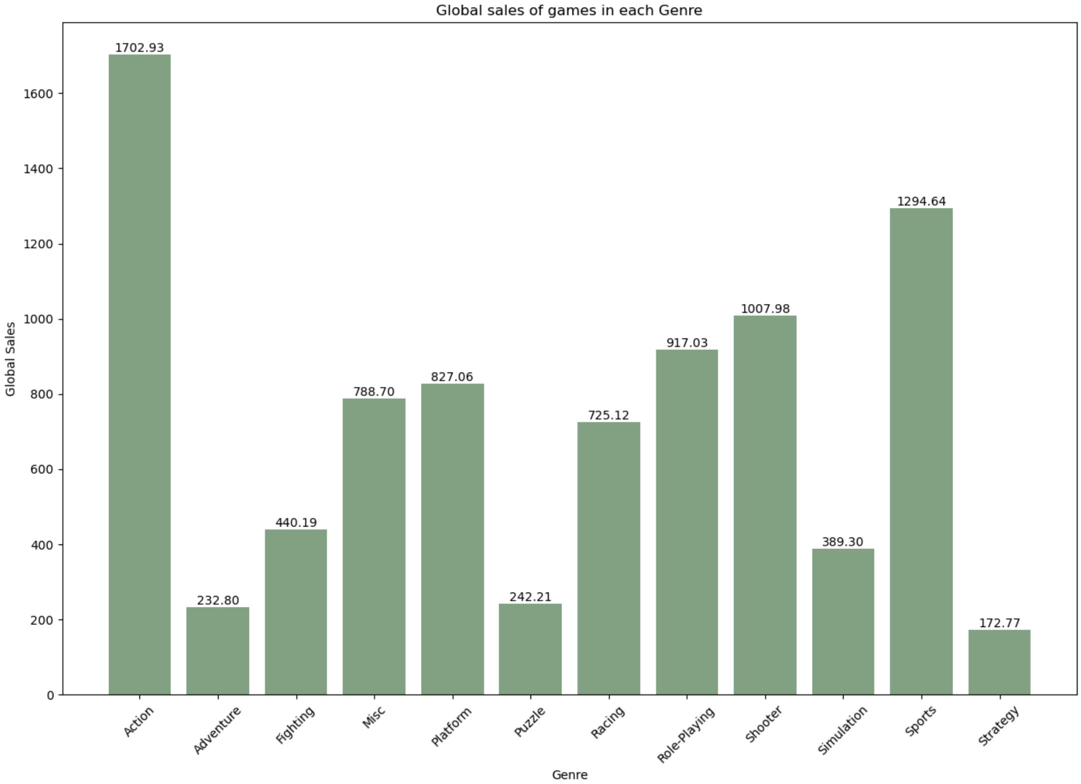
Ideally the accuracy should not be too close to or equal to 0. Thats why Lasso Regression is a best fit model for our dataset.

Figures (Data Visualisation)

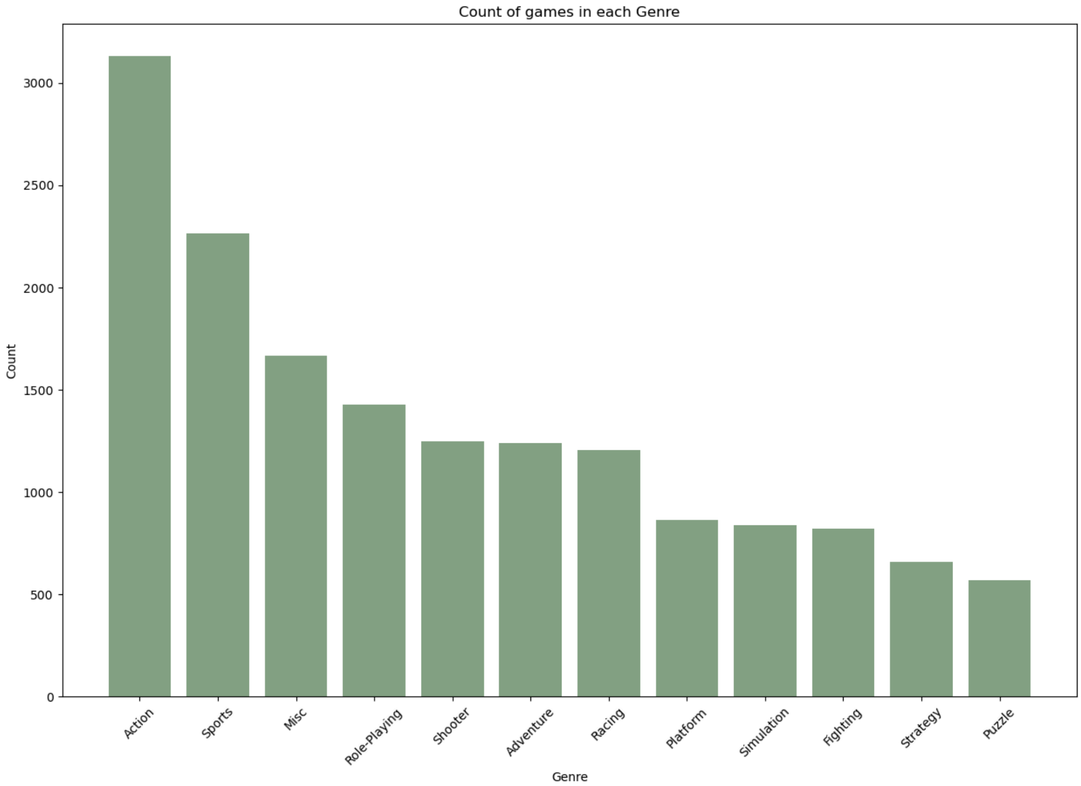
1. This graph represents Top Selling Games Globally



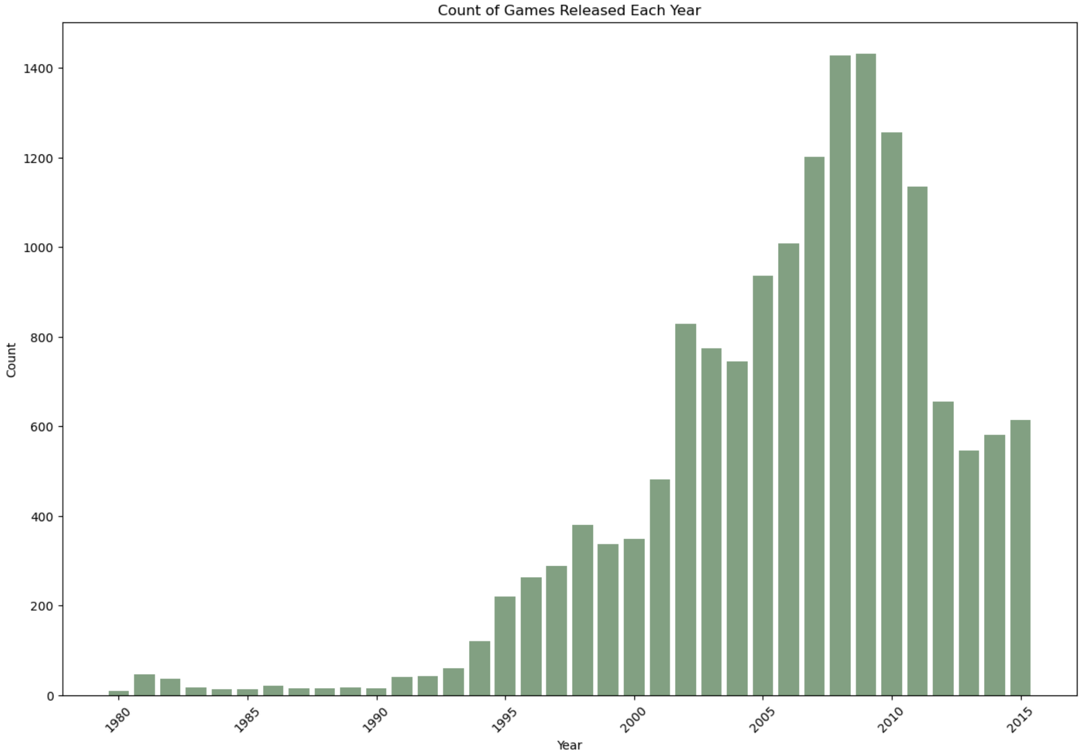
1. Global sales of games accross each Genre

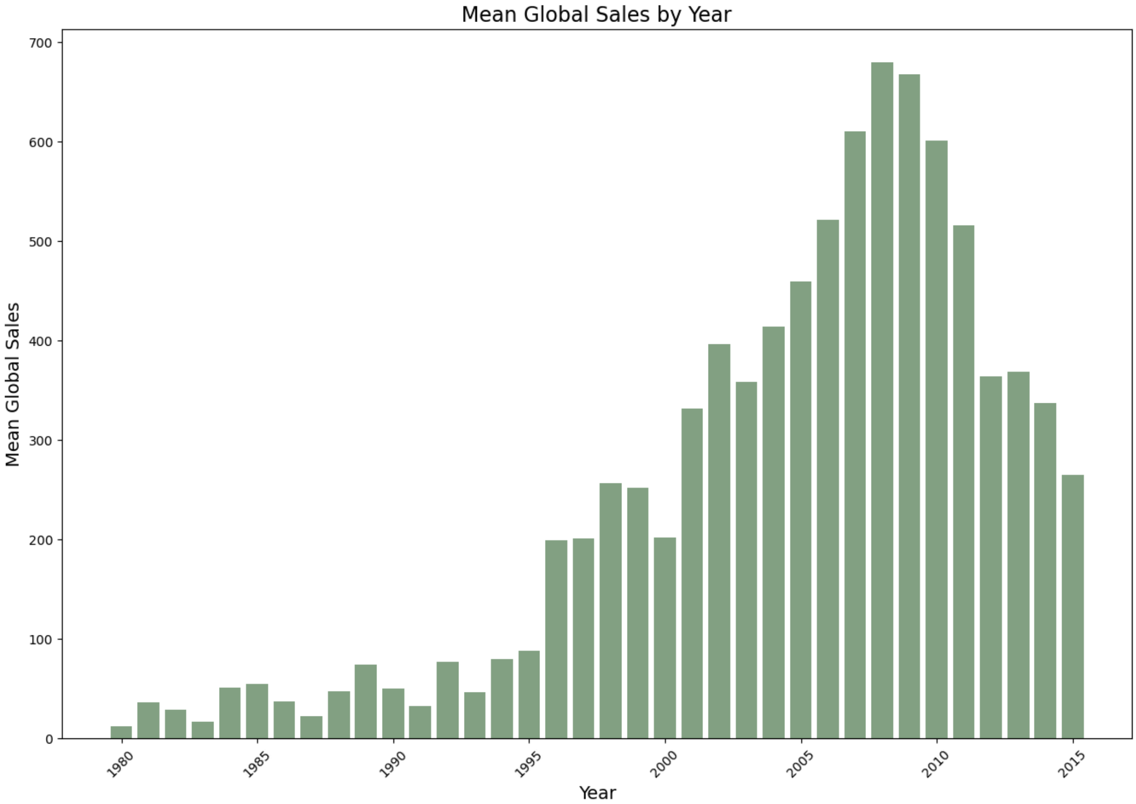


1. This graph shows total number of games in each Genre

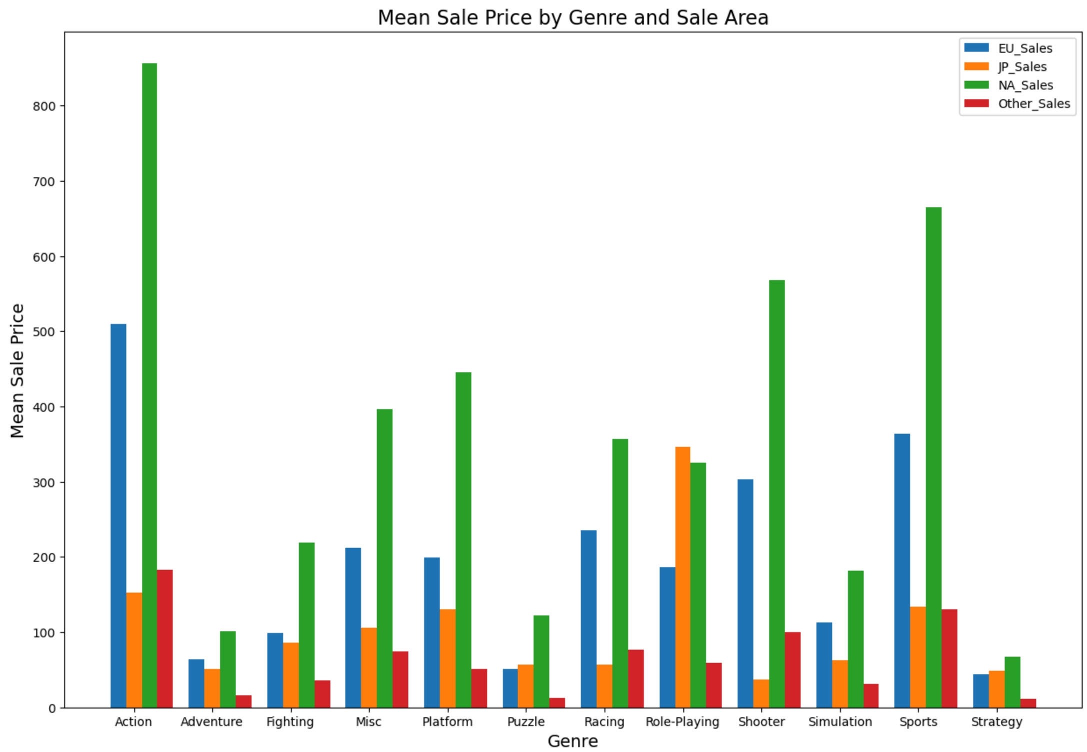


1. The below graph shows Count of games released each year

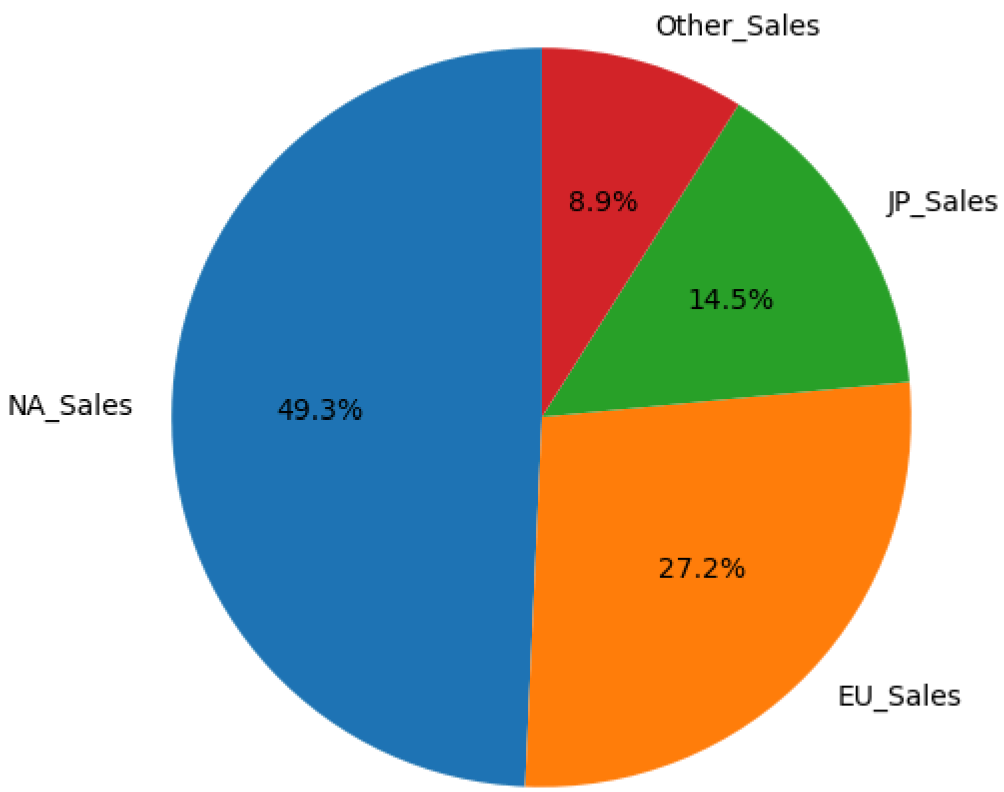




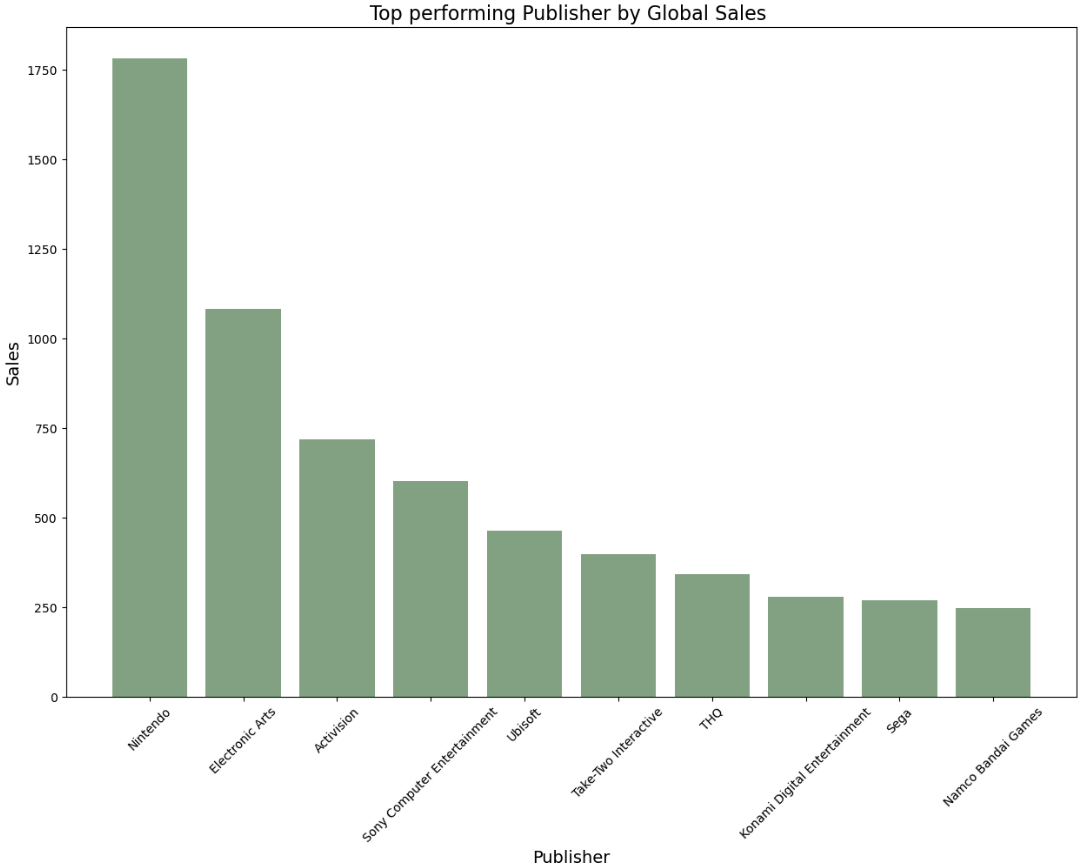
1. This bar plot show mean sale price by Genre and Sale Area



1. The below pie chart shows how much percentage each Sales contribute to Global Sales



1. This graph Top performing Publisher in GLobal Sales



Conclusion

To sum up, our experiment has shown how effective machine learning and artificial intelligence methods are in forecasting video game sales. By means of thorough data pretreatment, exploratory analysis, and model validation, we have acquired significant understanding of the intricate dynamics inside the gaming sector.

This study highlight how crucial it is to use sophisticated regression methods, such Decision Tree, Linear, and Random Forest regression, in order to predict video game sales with accuracy. The Random Forest model outperforms other models, demonstrating how well ensemble learning captures complex patterns and nonlinear interactions in the data. Furthermore, the competitive performance of Decision Tree Regression and Linear Regression models demonstrates the adaptability and usefulness of conventional regression techniques in predictive modeling.

This research also highlights the role that feature engineering and hyperparameter tuning play in improving the accuracy and generalization capabilities of models. We are able to create more stable and dependable prediction models for predicting market swings and sales patterns by choosing pertinent predictor variables and optimizing algorithmic parameters.

Importing required libraries and CSV file

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVR

color = (0.2, # redness
         0.4, # greenness
         0.2, # blueness
         0.6 # transparency
         )
```

In [3]:
```
dataset = pd.read_csv('vgsales.csv')
dataset.head()
```

Out[3]:

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

In [4]:
```
dataset.shape
```

Out[4]:
```
(16598, 11)
```

In [5]:
```
# The data above year 2015 is not enough to consider in the analysis so we are remo
drop_row_index = dataset[dataset['Year'] > 2015].index
dataset = dataset.drop(drop_row_index)
```

In [6]:
```
dataset.shape
```

Out[6]:
```
(16250, 11)
```

In [7]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16250 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Rank          16250 non-null  int64  
 1   Name          16250 non-null  object 
 2   Platform      16250 non-null  object 
 3   Year          15979 non-null  float64
 4   Genre         16250 non-null  object 
 5   Publisher     16194 non-null  object 
 6   NA_Sales      16250 non-null  float64
 7   EU_Sales      16250 non-null  float64
 8   JP_Sales      16250 non-null  float64
 9   Other_Sales   16250 non-null  float64
 10  Global_Sales  16250 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.5+ MB
```

Data Exploration, Cleaning and Visualization

In [8]:
```
dataset.describe()
```

Out[8]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Globa
count	16250.000000	15979.000000	16250.000000	16250.000000	16250.000000	16250.000000	16250
mean	8233.153785	2006.197071	0.268924	0.148146	0.078601	0.048614	0
std	4775.382512	5.714810	0.824467	0.509035	0.312196	0.190271	1
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0
25%	4095.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0
50%	8213.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0
75%	12340.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0
max	16600.000000	2015.000000	41.490000	29.020000	10.220000	10.570000	82

In [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
Rank              0
Name              0
Platform          0
Year            271
Genre             0
Publisher        56
NA_Sales          0
EU_Sales          0
JP_Sales          0
Other_Sales       0
Global_Sales      0
dtype: int64
```

In [10]:
```
dataset.dropna(inplace = True)
```

In [11]:
```
# Rank is a independent varial having no impact
dataset.drop('Rank' , axis = 1 , inplace = True)
```

In [12]:
```
dataset.isnull().sum()
```

Out[12]:
```
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
```

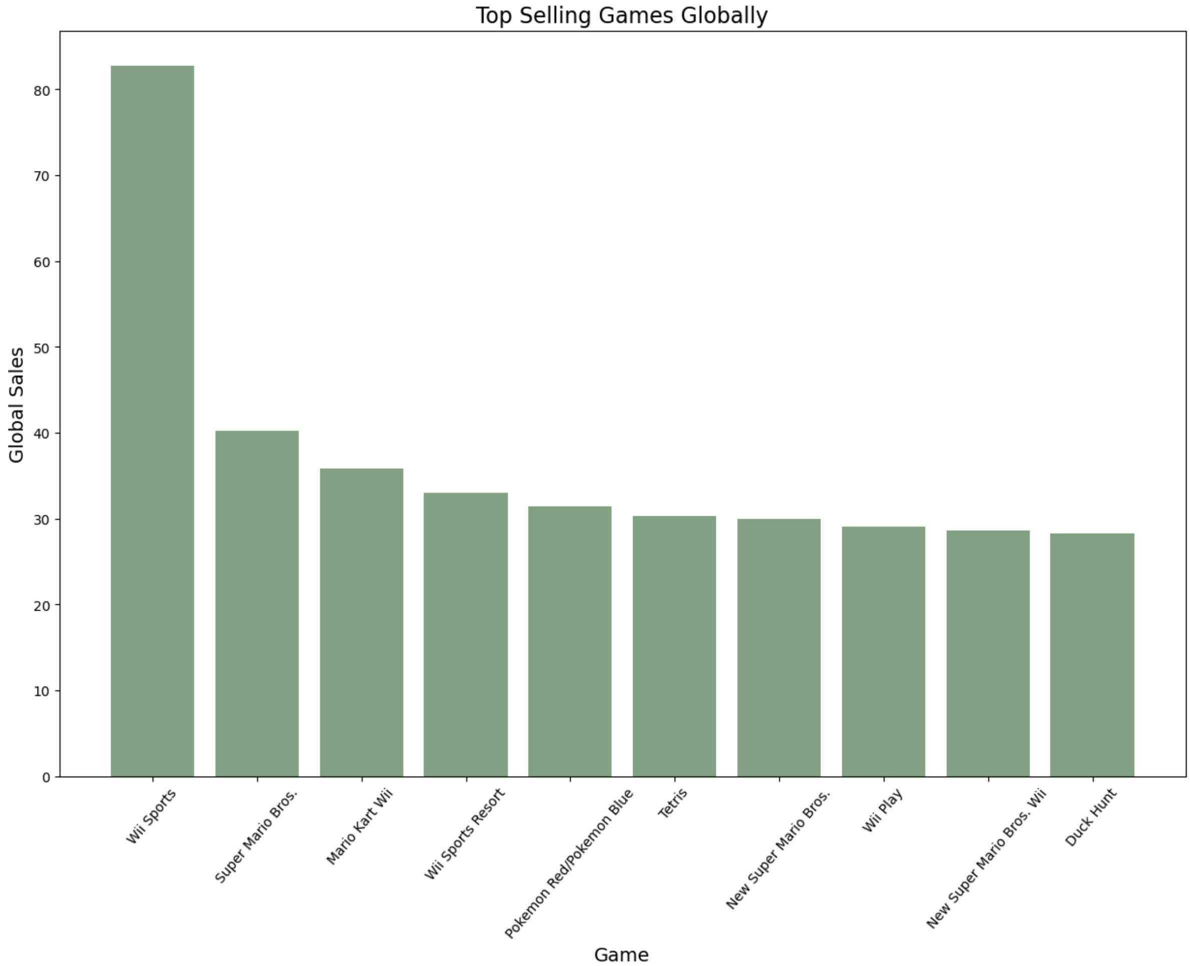
In [13]:
```
dataset.head(10)
```

Out[13]:

	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00
5	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4.22	0.58
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.38	9.23	6.50	2.90
7	Wii Play	Wii	2006.0	Misc	Nintendo	14.03	9.20	2.93	2.85
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.59	7.06	4.70	2.26
9	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0.28	0.47

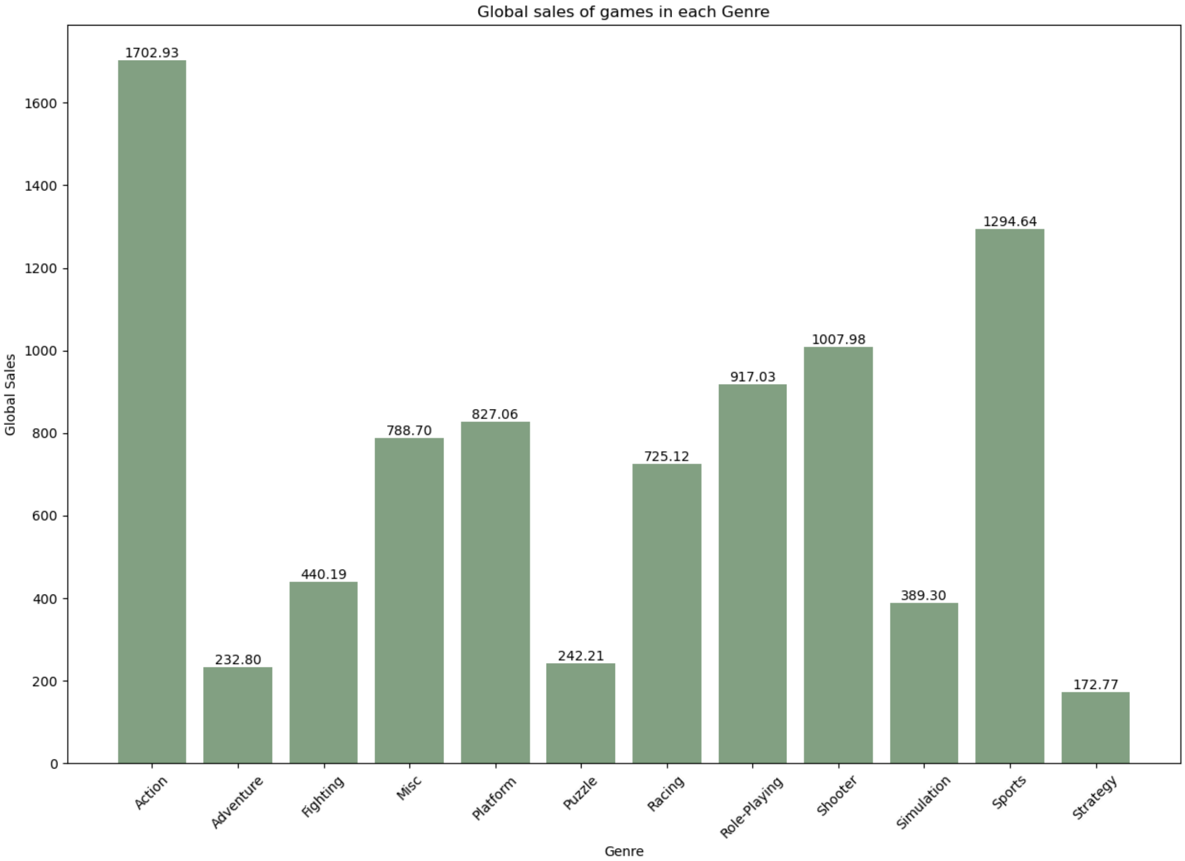
Data Visualisation

In [14]:
```
# Top selling games by global sells
top_game = dataset.sort_values('Global_Sales' , ascending = False)
top_selling_games = dataset.head(10)
# print(top_selling_games)

plt.figure(figsize=(15, 10))
plt.bar(top_selling_games['Name'] , top_selling_games['Global_Sales'] , color = col
plt.xlabel('Game', fontsize=14)
plt.ylabel('Global Sales', fontsize=14)
plt.title('Top Selling Games Globally', fontsize=16)
plt.xticks(rotation=50)
plt.show()
```



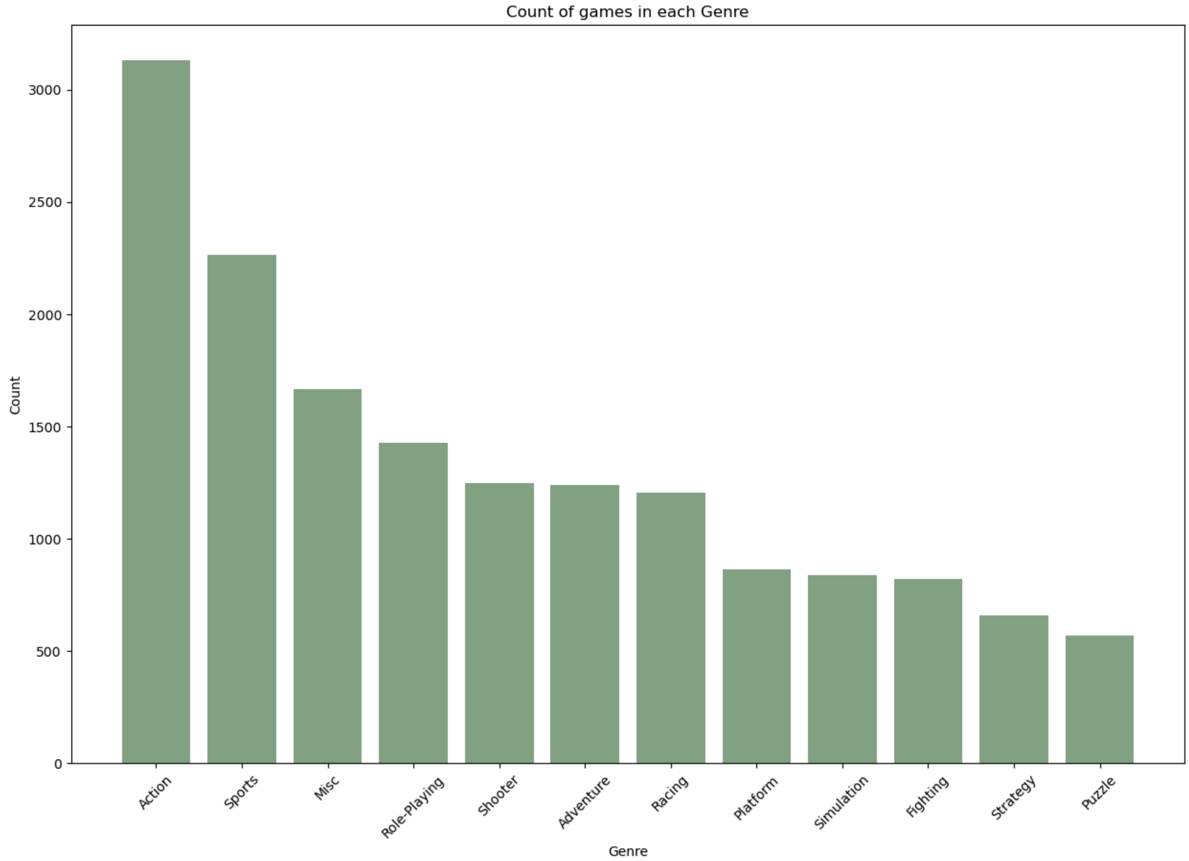
In [15]:

```
# Get the sales of games in each genre
genre_by_sales = dataset.groupby('Genre')['Global_Sales'].sum().reset_index()
genre_by_sales
#print(dataset['Genre'])
#print(genre_by_sales)

# Genre VS Count of games in each genre
plt.figure(figsize=(15, 10))
bar_plot = plt.bar(genre_by_sales['Genre'], genre_by_sales['Global_Sales'], color=c
plt.xlabel('Genre')
plt.ylabel('Global Sales')
plt.title('Global sales of games in each Genre')
plt.xticks(rotation=45)
plt.bar_label(bar_plot, fmt='%.2f', label_type='edge')
plt.show()
```

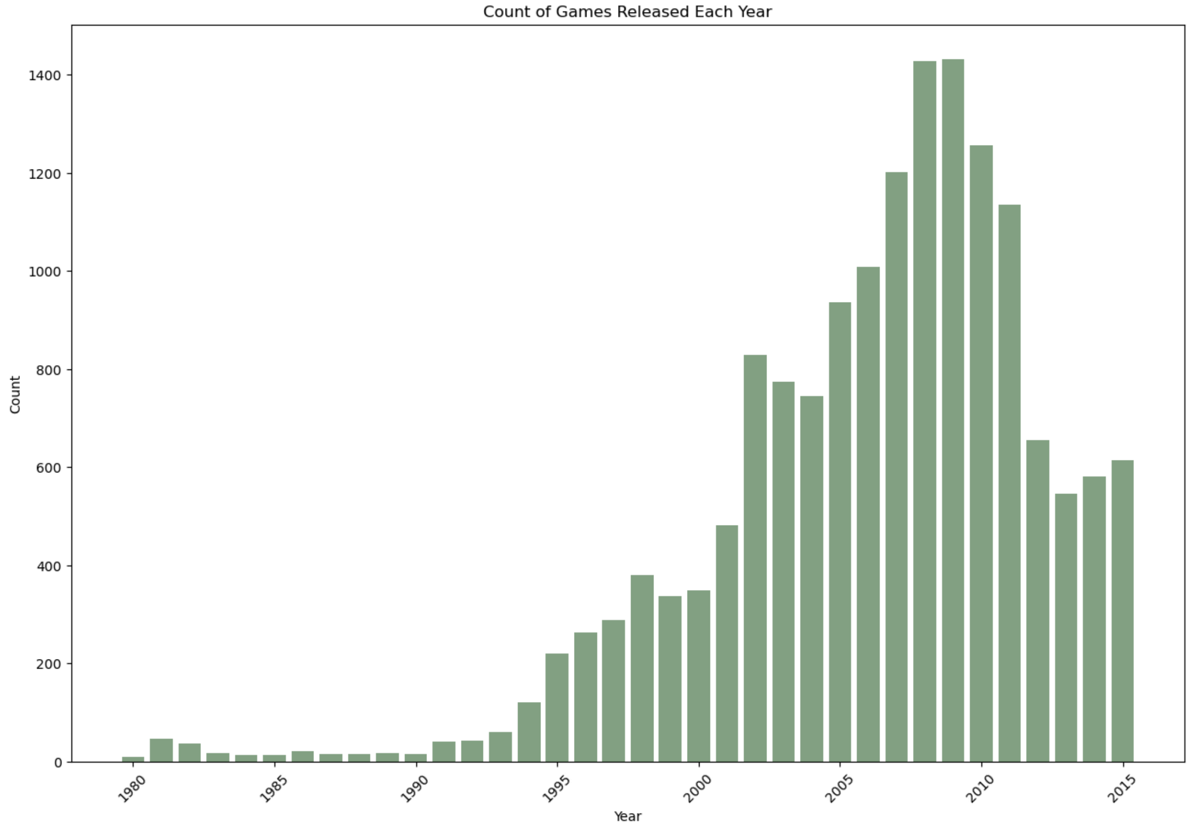


In [16]:
```python
# Get the counts games of each genre
genre_counts = dataset['Genre'].value_counts()
print(genre_counts)
# print(dataset['Genre'])

# Genre VS Count of games in each genre
plt.figure(figsize=(15, 10))
plt.bar(genre_counts.index, genre_counts.values, color = color)
plt.xlabel('Genre')
plt.ylabel('Count')
plt.title('Count of games in each Genre')
plt.xticks(rotation=45)
plt.show()
```

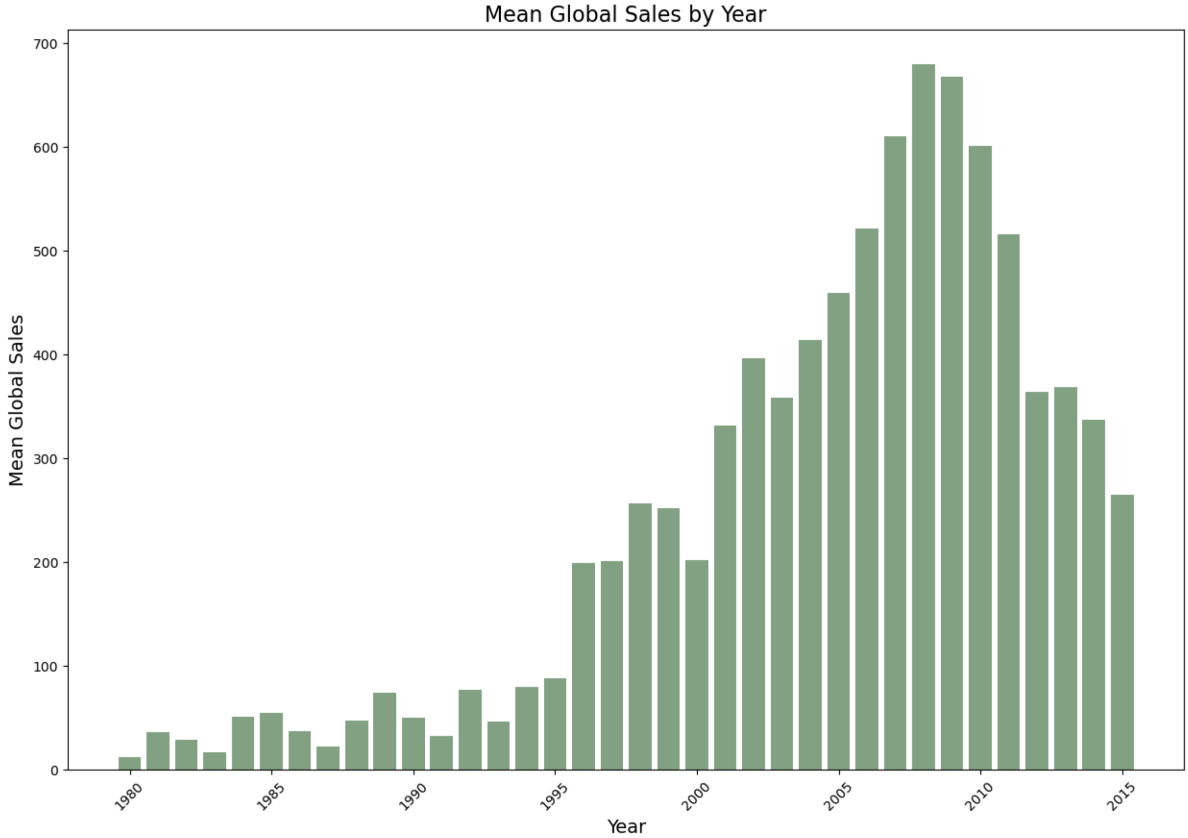
```
Genre
Action          3132
Sports          2266
Misc            1668
Role-Playing    1428
Shooter         1250
Adventure       1241
Racing          1205
Platform         865
Simulation       838
Fighting         822
Strategy         660
Puzzle           570
Name: count, dtype: int64
```



In [17]:
```
# Count of game released in each year
year_counts = dataset.groupby('Year')['Name'].count()
# print(year_counts)

# Sort the grouped data in descending order of counts
sorted_year_counts = year_counts.sort_values(ascending=False)

# Plot
plt.figure(figsize=(15, 10))
plt.bar(sorted_year_counts.index, sorted_year_counts.values, color = color)
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('Count of Games Released Each Year')
plt.xticks(rotation=45)
plt.show()
```



In [18]:
```
data_year = dataset.groupby(by=['Year'])['Global_Sales'].sum()
data_year = data_year.reset_index()

# Plotting
plt.figure(figsize=(15, 10))
plt.bar(data_year['Year'], data_year['Global_Sales'], color = color)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Mean Global Sales', fontsize=14)
plt.title('Mean Global Sales by Year', fontsize=16)
plt.xticks(rotation=45)
plt.show()
```



In [19]:
```
comp_genre = dataset[['Genre', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']]

# comp_genre
comp_map = comp_genre.groupby(by=['Genre']).sum()

comp_table = comp_map.reset_index()
comp_table = pd.melt(comp_table, id_vars=['Genre'], value_vars=['NA_Sales', 'EU_Sal
comp_table.head(10)
```

Out[19]:

	Genre	Sale_Area	Sale_Price
0	Action	NA_Sales	855.90
1	Adventure	NA_Sales	101.59
2	Fighting	NA_Sales	219.14
3	Misc	NA_Sales	396.70
4	Platform	NA_Sales	445.20
5	Puzzle	NA_Sales	122.01
6	Racing	NA_Sales	356.60
7	Role-Playing	NA_Sales	325.11
8	Shooter	NA_Sales	567.72
9	Simulation	NA_Sales	181.51

In [20]:
```
sales_by_region = dataset[['NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']].sum(
sales_by_region = sales_by_region.reset_index()
sales_by_region.columns = ['Region','Total_sales'] + list(sales_by_region.columns[2
sales_by_region
```

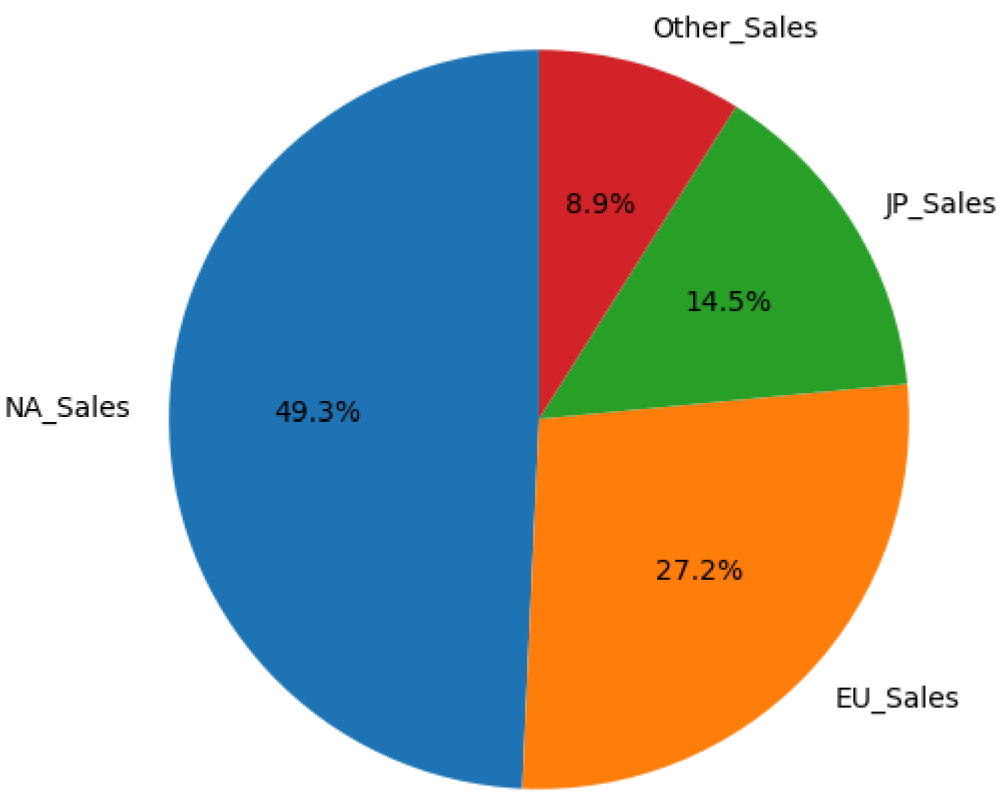
Out[20]:

	Region	Total_sales
0	NA_Sales	4304.72
1	EU_Sales	2379.93
2	JP_Sales	1270.55
3	Other_Sales	781.14

In [21]:
```
labels = sales_by_region['Region']
sizes = sales_by_region['Total_sales']
```

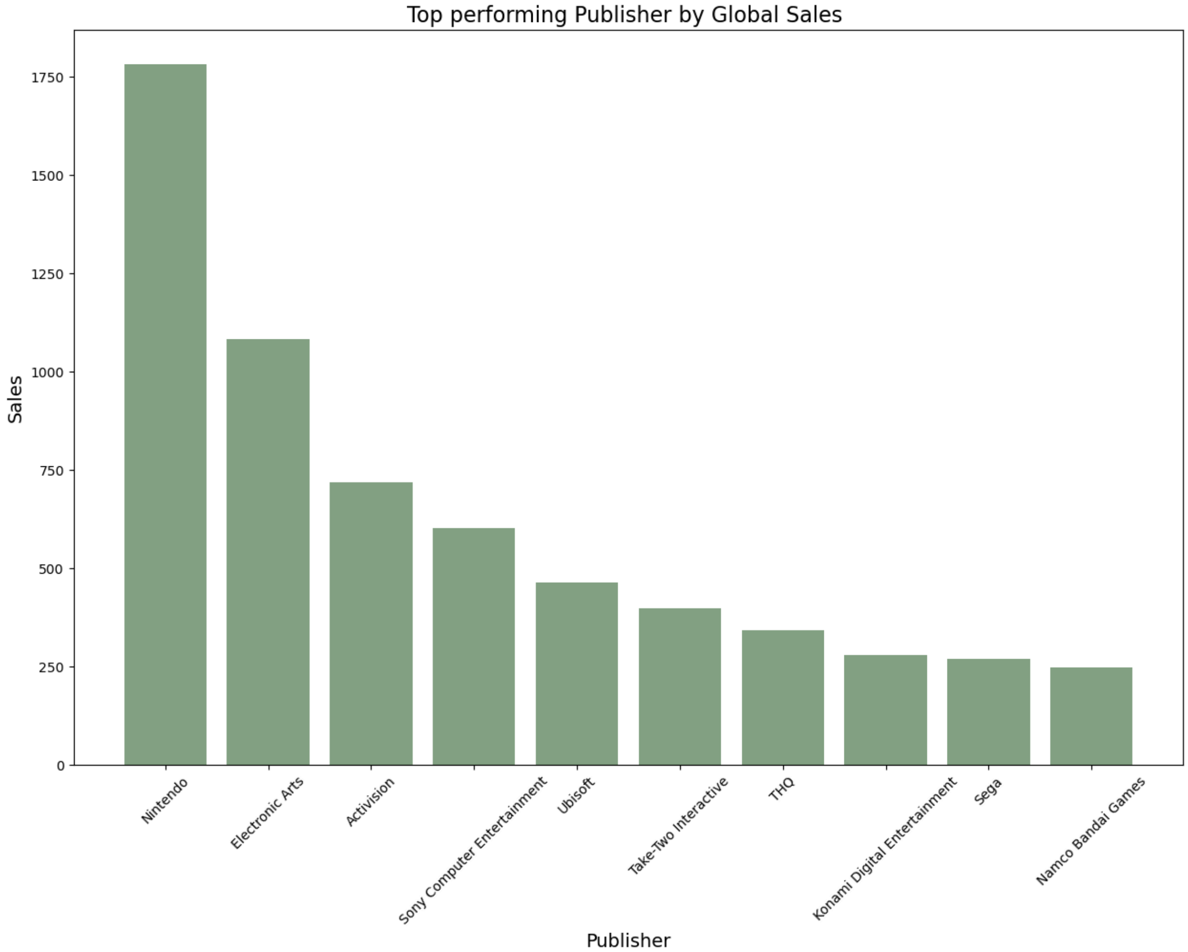
In [22]:
```
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
```

Out[22]:
```
([<matplotlib.patches.Wedge at 0x17be38435d0>,
  <matplotlib.patches.Wedge at 0x17be38ac890>,
  <matplotlib.patches.Wedge at 0x17be38ae590>,
  <matplotlib.patches.Wedge at 0x17be38bc2d0>],
 [Text(-1.0997136849504432, 0.02509603818768038, 'NA_Sales'),
  Text(0.7968607384711724, -0.7582960922246519, 'EU_Sales'),
  Text(0.9365621291923075, 0.5769327327884697, 'JP_Sales'),
  Text(0.30494053449515507, 1.05688753915533, 'Other_Sales')],
 [Text(-0.5998438281547872, 0.013688748102371114, '49.3%'),
  Text(0.43465131189336675, -0.4136160503043555, '27.2%'),
  Text(0.5108520704685313, 0.3146905815209834, '14.5%'),
  Text(0.16633120063372095, 0.5764841122665436, '8.9%')])
```



In [23]:
```
# Top sales by publisher
publisher_sales = dataset.groupby('Publisher')['Global_Sales'].sum()
sort_publisher = publisher_sales.sort_values(ascending = False)
top_publisher =  sort_publisher.head(10).reset_index()
top_publisher
```

```
plt.figure(figsize=(15, 10))
plt.bar(top_publisher['Publisher'],top_publisher['Global_Sales'] , color = color )
plt.xlabel('Publisher', fontsize=14)
plt.ylabel('Sales', fontsize=14)
plt.title('Top performing Publisher by Global Sales ', fontsize=16)
plt.xticks(rotation=45)
plt.show()
```



Model Building

In [24]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15945 entries, 0 to 16597
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Name          15945 non-null  object 
 1   Platform      15945 non-null  object 
 2   Year          15945 non-null  float64
 3   Genre         15945 non-null  object 
 4   Publisher     15945 non-null  object 
 5   NA_Sales      15945 non-null  float64
 6   EU_Sales      15945 non-null  float64
 7   JP_Sales      15945 non-null  float64
 8   Other_Sales   15945 non-null  float64
 9   Global_Sales  15945 non-null  float64
dtypes: float64(6), object(4)
memory usage: 1.3+ MB
```

In [25]:
```
dataset.head()
```

Out[25]:

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

In [26]:

```
## Label encoding
## This step is important as when we perform analysis and use models on the data th
## cannot convert string values to number or float for performin analysis all the v

columns_names = ['Name','Platform', 'Genre', 'Publisher']
for col in columns_names:
    dataset[col] = LabelEncoder().fit_transform(dataset[col])
```

In [27]:

```
dataset.head()
```

Out[27]:

	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_S
0	10664	26	2006.0	10	351	41.49	29.02	3.77	8.46	8.
1	9045	11	1985.0	4	351	29.08	3.58	6.81	0.77	4
2	5400	26	2008.0	6	351	15.85	12.88	3.79	3.31	3
3	10666	26	2009.0	10	351	15.75	11.01	3.28	2.96	3
4	7129	5	1996.0	7	351	11.27	8.89	10.22	1.00	3

Modelling

In [28]:

```
X = dataset.drop(['Global_Sales'], axis=1)
Y = dataset['Global_Sales']
```

In [29]:

```
X.head()
```

Out[29]:

	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales
0	10664	26	2006.0	10	351	41.49	29.02	3.77	8.46
1	9045	11	1985.0	4	351	29.08	3.58	6.81	0.77
2	5400	26	2008.0	6	351	15.85	12.88	3.79	3.31
3	10666	26	2009.0	10	351	15.75	11.01	3.28	2.96
4	7129	5	1996.0	7	351	11.27	8.89	10.22	1.00

In [30]:

```
Y.head()
```

Out[30]:
```
0    82.74
1    40.24
2    35.82
3    33.00
4    31.37
Name: Global_Sales, dtype: float64
```

In [31]:
```
X_train , X_test , y_train , y_test = train_test_split(X,Y , test_size = 0.2 , ranc
```

In [32]:
```
print(X_train)
print(X_test)
print(y_train)
print(y_test)
```

```
        Name  Platform    Year  Genre  Publisher  NA_Sales  EU_Sales  \
8262    6320        28  2008.0     10        482      0.15      0.01   
15702   7073        19  2009.0      1        403      0.00      0.00   
14950  10441        19  2005.0     10        436      0.00      0.00   
12341   3627        19  2011.0      7         55      0.00      0.00   
5049    9880        19  2007.0     10        137      0.14      0.15   
...      ...       ...     ...    ...        ...       ...       ...   
920     2820        15  1998.0     10        137      0.22      1.47   
5321    5956        28  2008.0      6         85      0.11      0.20   
12554   9853         4  2008.0      1         39      0.06      0.00   
237     3665        28  2007.0      3         21      3.19      0.92   
13793   9160        17  2007.0     10        512      0.03      0.00   

       JP_Sales  Other_Sales  
8262       0.00         0.01  
15702      0.02         0.00  
14950      0.02         0.00  
12341      0.06         0.00  
5049       0.00         0.09  
...         ...          ...  
920        0.04         0.14  
5321       0.00         0.04  
12554      0.00         0.00  
237        0.01         0.42  
13793      0.00         0.00  

[12756 rows x 9 columns]
       Name  Platform    Year  Genre  Publisher  NA_Sales  EU_Sales  JP_Sales  \
310    9508        13  2011.0      7         66      1.15      2.09      0.00   
5846   5085         6  2005.0      4        126      0.22      0.08      0.00   
16398  6973         6  2007.0      3        512      0.01      0.00      0.00   
4990   1146        26  2008.0     10         23      0.36      0.00      0.00   
2405   3674        28  2010.0      3         21      0.47      0.32      0.00   
...     ...       ...     ...    ...        ...       ...       ...       ...   
2108   4549        28  2013.0      3        512      0.72      0.19      0.00   
6836    739        19  2010.0      4        109      0.10      0.09      0.00   
10510  1910        16  2005.0      0        445      0.05      0.04      0.00   
8662   7353        15  1996.0      2         17      0.01      0.01      0.13   
1478   3052        17  2013.0      7        454      0.43      0.40      0.32   

       Other_Sales  
310           0.64  
5846          0.01  
16398         0.00  
4990          0.03  
2405          0.08  
...            ...  
2108          0.07  
6836          0.05  
10510         0.01  
8662          0.01  
1478          0.18  

[3189 rows x 9 columns]
8262     0.17
15702    0.02
14950    0.02
12341    0.06
5049     0.38
         ... 
920      1.87
5321     0.35
12554    0.06
237      4.53
```

```
13793    0.04
Name: Global_Sales, Length: 12756, dtype: float64
310      3.88
5846     0.30
16398    0.01
4990     0.38
2405     0.87
         ...
2108     0.98
6836     0.24
10510    0.10
8662     0.16
1478     1.33
Name: Global_Sales, Length: 3189, dtype: float64
```

Random Forest

In [33]:
```
from sklearn.ensemble import RandomForestClassifier

ranf_model = RandomForestRegressor()
ranf_model.fit(X_train, y_train)

ranf_pred = ranf_model.predict(X_test)
ranf_pred
```

Out[33]:
```
array([3.9225, 0.3092, 0.0134, ..., 0.1016, 0.1456, 1.2795])
```

In [34]:
```
ranf_model.score(X_test, y_test)
```

Out[34]:
```
0.9783938393144076
```

In [35]:
```
# Compute the accuracy, MSE, and R2 for the testing set

accuracy_ranf = ranf_model.score(X_test, y_test)

mse_ranf = mean_squared_error(y_test, ranf_pred)

rmse_ranf = np.sqrt(mse_ranf)

r2_ranf = r2_score(y_test, ranf_pred)

print(accuracy_ranf*100)
```

```
97.83938393144076
```

Multiple Linear Regression

In [36]:
```
from sklearn.linear_model import LinearRegression

multi_r_model = LinearRegression()
multi_r_model.fit(X_train, y_train)

mlr_pred = multi_r_model.predict(X_test)
mlr_pred
```

Out[36]:
```
array([3.88015393, 0.31046733, 0.01015627, ..., 0.10023745, 0.16047829,
       1.33001911])
```

In [37]:
```
multi_r_model.score(X_test, y_test)
```

Out[37]:
```
0.9999871337999723
```

In [38]:
```
# Compute the accuracy, MSE, and R2 for the testing set

accuracy_mlr = multi_r_model.score(X_test, y_test)

mse_mlr = mean_squared_error(y_test, mlr_pred)

rmse_mlr = np.sqrt(mse_mlr)

r2_mlr = r2_score(y_test, mlr_pred)

print(accuracy_mlr*100)
```

```
99.99871337999723
```

Decision Tree

In [39]:
```
from sklearn.tree import DecisionTreeRegressor
dt_model = DecisionTreeRegressor( random_state = 32)
dt_model.fit(X_train, y_train)

dt_pred = multi_r_model.predict(X_test)
dt_pred
```

Out[39]:
```
array([3.88015393, 0.31046733, 0.01015627, ..., 0.10023745, 0.16047829,
       1.33001911])
```

In [40]:
```
dt_model.score(X_test, y_test)
```

Out[40]:
```
0.9624834920825148
```

In [41]:
```
# Compute the accuracy, MSE, and R2 for the testing set
accuracy_dt = dt_model.score(X_test, y_test)
mse_dt = mean_squared_error(y_test, dt_pred)
rmse_dt = np.sqrt(mse_dt)
r2_dt = r2_score(y_test, dt_pred)

print(accuracy_dt*100)
```

```
96.24834920825148
```

Lasso regreession

In [42]:
```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha = 0.1)

# Train the model
lasso_model.fit(X_train, y_train)

# Predict on the test set
lasso_pred = lasso_model.predict(X_test)
lasso_pred
```

Out[42]:
```
array([3.31474929, 0.41245608, 0.12487648, ..., 0.19850967, 0.09558673,
       0.96878685])
```

In [43]:
```
lasso_model.score(X_test, y_test)
```

Out[43]:
```
0.9298986871032685
```

In [44]:
```
# Compute the accuracy, MSE, and R2 for the testing set
accuracy_lasso = lasso_model.score(X_test, y_test)
```

```
mse_lasso = mean_squared_error(y_test, lasso_pred)
rmse_lasso = np.sqrt(mse_lasso)
r2_lasso = r2_score(y_test, lasso_pred)

print(accuracy_lasso*100)
```

92.98986871032685

Model comparison

In [45]:

```
# Create a dictionary to store the evaluation metrics
comparison_table = {
    'Model': ['Random Forest', 'Multiple Linear Regression', 'Decision Tree', 'Lass
    'Accuracy': [accuracy_ranf, accuracy_mlr, accuracy_dt, accuracy_lasso],
    'Mean Squared Error (MSE)': [mse_ranf, mse_mlr, mse_dt, mse_lasso],
    'Root Mean Squared Error (RMSE)': [rmse_ranf, rmse_mlr, rmse_dt, rmse_lasso],
    'R-squared (R2)': [r2_ranf, r2_mlr, r2_dt, r2_lasso]
}

# Create a DataFrame
table = pd.DataFrame(comparison_table)

table
```

Out[45]:

	Model	Accuracy	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared (R2)
0	Random Forest	0.978394	0.048364	0.219919	0.978394
1	Multiple Linear Regression	0.999987	0.000029	0.005367	0.999987
2	Decision Tree	0.962483	0.000029	0.005367	0.999987
3	Lasso Regression	0.929899	0.156918	0.396129	0.929899