## ABSTRACT

This report is done to show the analysis dataset as shown in summary in the introduction section. This purpose of this analysis is to predict the global sales of the video games using regression analysis. Various regressors were used, as well as classification and clustering methods outputs. Indicative metrics showed that all the regressors were effective predicting sales efficiency of all the regressors. Using numerical data from various independent numerical variables (not including year). However, Random Forest proved to be the most effective in terms faster deployment.

## INTRODUCTION

The dataset consist of various data of hundreds of different video games for each year, spanning from 1980s to 2016 The following are the features of the video games as stated on Kaggle (Video Game Sales Dataset Updated -Extra Feat, Accessed: 25th May 2023).

Table 1 (Sourced from Kaggle.com)

| Column Name | Description |
|---|---|
| **Name** | The name of the video game. |
| **Platform** | The platform on which the game was released, such as PlayStation, Xbox, Nintendo, etc. |
| **Year of Release** | The year in which the game was released. |
| **Genre** | The genre of the video game, such as action, adventure, sports, etc. |
| **Publisher** | The company responsible for publishing the game. |
| **NA Sales*** | The sales of the game in North America. |
| **EU Sales*** | The sales of the game in Europe. |
| **JP Sales*** | The sales of the game in Japan. |
| **Other Sales*** | The sales of the game in other regions. |
| **Global Sales*** | The total sales of the game across the world. |
| **Critic Score** | The average score given to the game by professional critics. |
| **Critic Count** | The number of critics who reviewed the game. |
| **User Score** | The average score given to the game by users. |
| **User Count** | The number of users who reviewed the game. |
| **Developer** | The company responsible for developing the game. |
| **Rating** | The rating assigned to the game by organizations such as the ESRB or PEGI. |

*The Sales is assumed to be in millions of dollars as it is not stated in the source

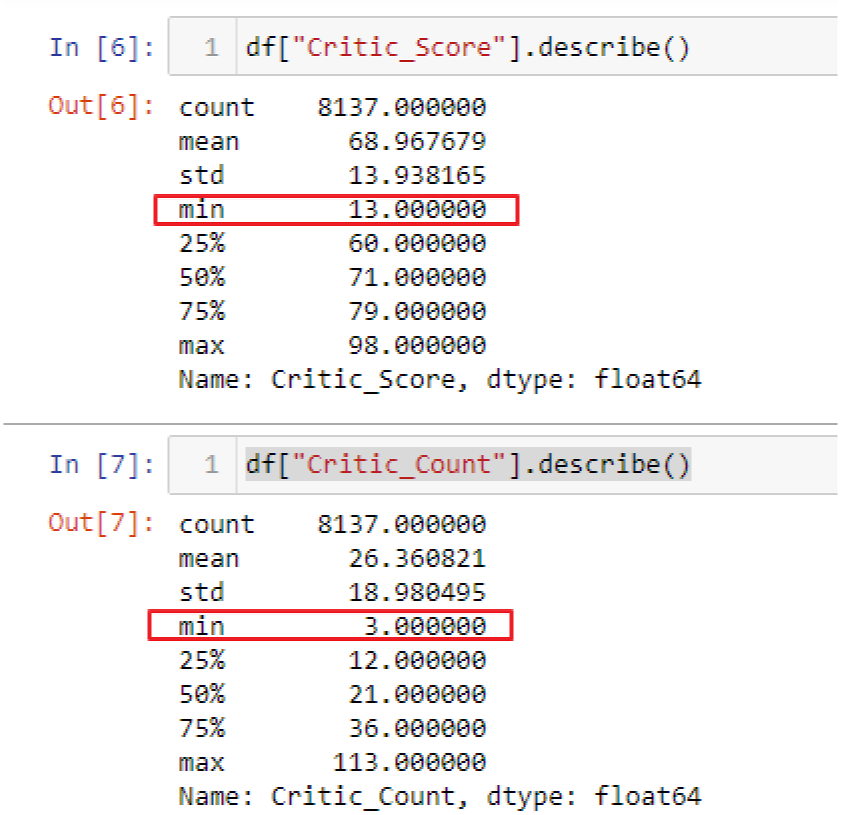
With this background information, analyzing the data and running knowledge, a better understanding was gained in order to analyze the data accordingly and then make an informed decision as to what model can be used for the different target data.

## 1.0. METHODOLOGY.

1.1. **Data Preprocessing**: This was done using Pandas and Numpy, both were employed in order to achieve the aim of cleaning the data.

A. **Data Cleaning**: Null values and data types that are not fitting were found. For example, numerical data such as Critic Score, Critic Count, User Score and User Count have null values that account for around half of the dataset, also object data types were found instead of floats. These were dealt with as highlighted below:

i. **Null values were replaced**. Median value of gotten from available values within the ‘User Score’ column were used to replace the null values. Within the ‘Critic Score’ and ‘Critic Count’ the null values were replaced with zero based on the fact that their min score is greater than zero as shown in the figure below



```
In [6]:   df["Critic_Score"].describe()

Out[6]:  count    8137.000000
         mean       68.967679
         std        13.938165
         min        13.000000
         25%        60.000000
         50%        71.000000
         75%        79.000000
         max        98.000000
         Name: Critic_Score, dtype: float64
```

```
In [7]:   df["Critic_Count"].describe()

Out[7]:  count    8137.000000
         mean       26.360821
         std        18.980495
         min         3.000000
         25%        12.000000
         50%        21.000000
         75%        36.000000
         max       113.000000
         Name: Critic_Count, dtype: float64
```

Fig 1

ii. **Dropping the null values**: Null values of numerical data were dropped in ‘User Score’, ‘User count’ and this is justified based on two facts observed: first, there are zero values in both ‘User counts’ and ‘User Score’ as opposed to Critic Count and Critic Score.

iii. -Year of Release that were null were also dropped as the number of null values were just around 200 in number.

-Other data feature having few null values such as Name (2 null values), Publisher (54 null values) were dropped.

-Categorical data in the variable having null values as seen in ‘Developer’ and ‘Rating’ were replaced with unknown in order not affect the numerical aspect of the data.

### 1.2. Exploratory Analysis and Visualization

Analysis were done to identify trends and features of the variable, also relationships or correlations where also studied to help us understand the data before building a model that can work well for it.

i. **Regrouping ‘Platform’ Variable:** This was done by identifying platforms that are equal or more than 700 in number and then categorizing those less into ‘others’. The condition was created and then a new column ‘Platform_regrouped’ was formed. This is important as it helps in focusing on a few big platforms that will have significant effect on the target data. Also, it helps in making it easy for us to do One-Hot Encoding.

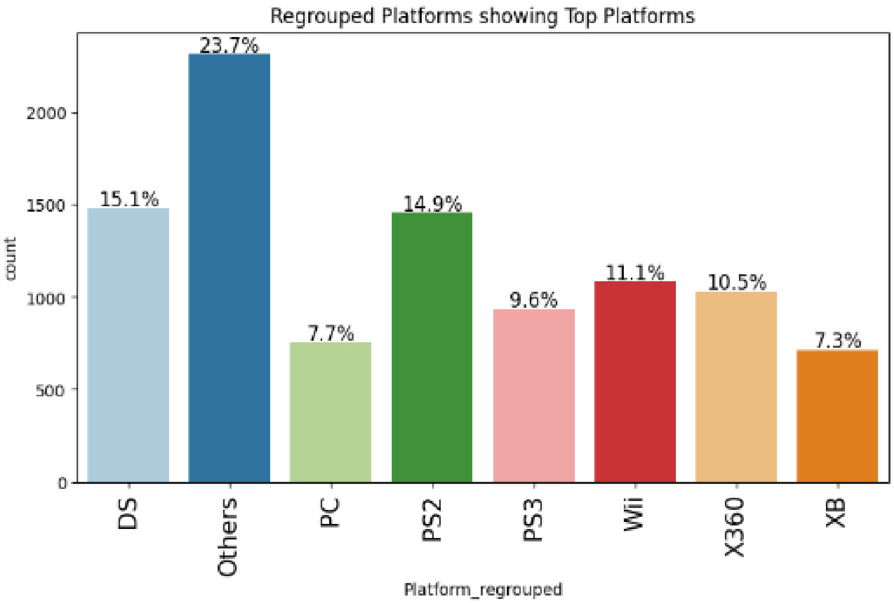
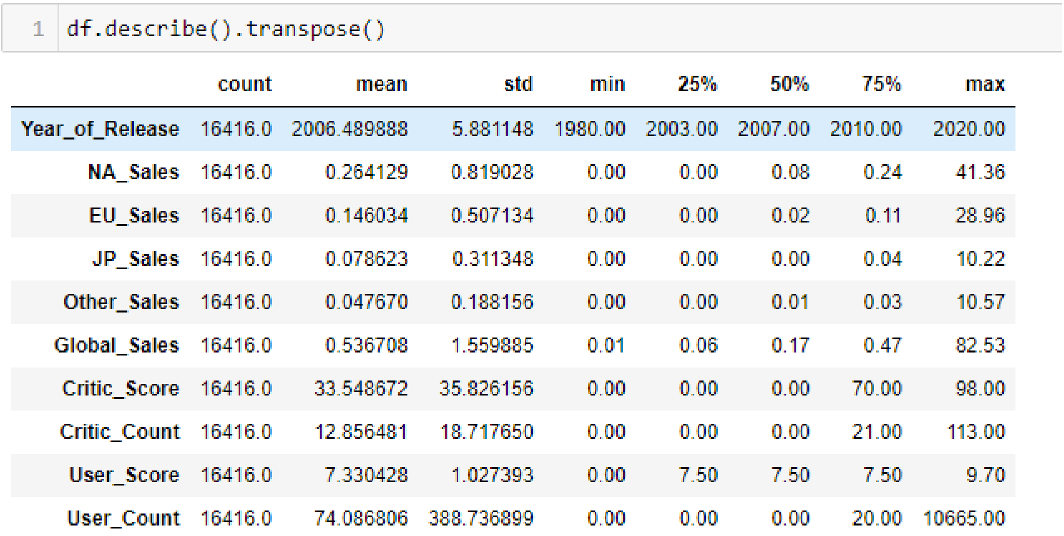


Fig 2

### ii. Statistical Analysis:



```
df.describe().transpose()
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Year_of_Release** | 16416.0 | 2006.489888 | 5.881148 | 1980.00 | 2003.00 | 2007.00 | 2010.00 | 2020.00 |
| **NA_Sales** | 16416.0 | 0.264129 | 0.819028 | 0.00 | 0.00 | 0.08 | 0.24 | 41.36 |
| **EU_Sales** | 16416.0 | 0.146034 | 0.507134 | 0.00 | 0.00 | 0.02 | 0.11 | 28.96 |
| **JP_Sales** | 16416.0 | 0.078623 | 0.311348 | 0.00 | 0.00 | 0.00 | 0.04 | 10.22 |
| **Other_Sales** | 16416.0 | 0.047670 | 0.188156 | 0.00 | 0.00 | 0.01 | 0.03 | 10.57 |
| **Global_Sales** | 16416.0 | 0.536708 | 1.559885 | 0.01 | 0.06 | 0.17 | 0.47 | 82.53 |
| **Critic_Score** | 16416.0 | 33.548672 | 35.826156 | 0.00 | 0.00 | 0.00 | 70.00 | 98.00 |
| **Critic_Count** | 16416.0 | 12.856481 | 18.717650 | 0.00 | 0.00 | 0.00 | 21.00 | 113.00 |
| **User_Score** | 16416.0 | 7.330428 | 1.027393 | 0.00 | 7.50 | 7.50 | 7.50 | 9.70 |
| **User_Count** | 16416.0 | 74.086806 | 388.736899 | 0.00 | 0.00 | 0.00 | 20.00 | 10665.00 |

Fig 3

The numerical features of the data were analysed using the '***.describe ()***' method on python, and as shown the figure above. As regards the sales, it is assumed based on domain knowledge of Video game sales, the sales figures are in millions of dollars. With that in mind the following were observed:

1. Global sales had the highest average sales among all with $536,708
2. With regards to regions or country, North America possess the highest sales with $264,129
3. Critic Scores were given on a scale of 0 to 100 and the average score given by Critics was 33.
4. User Scores were given on a scale of 0 to 10 and the average score given by User was 7.
5. We can see a huge variation between the mean value and semi-interquartile value, which indicates the presence of outliers, which will be dealt with later in this report.

iii. **Trends of relationship between the variable and target (Global Sales):** Using Scatter Matrix, Correlation Matrix, insight was gained on how the variables relatesd with Global Sales.

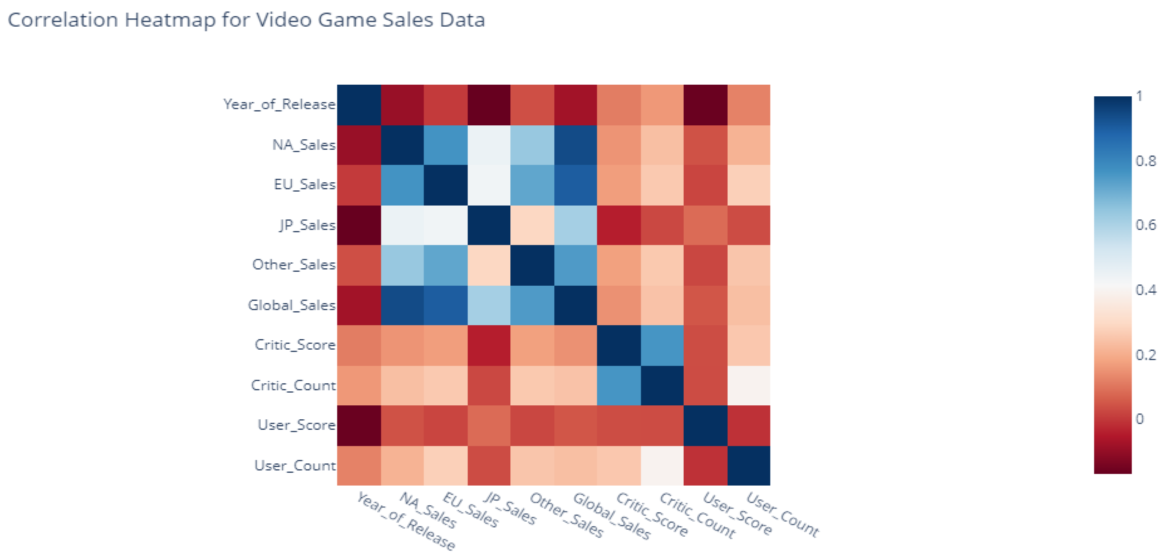


Fig 4

From the Correlation matrix above, we high correlation (signified by dark blue) between Global Sales and the Sales Features: ‘Na_Sales’ having highest correlation, followed by EU_Sales, Other_Sales and JP Sales in descending order.

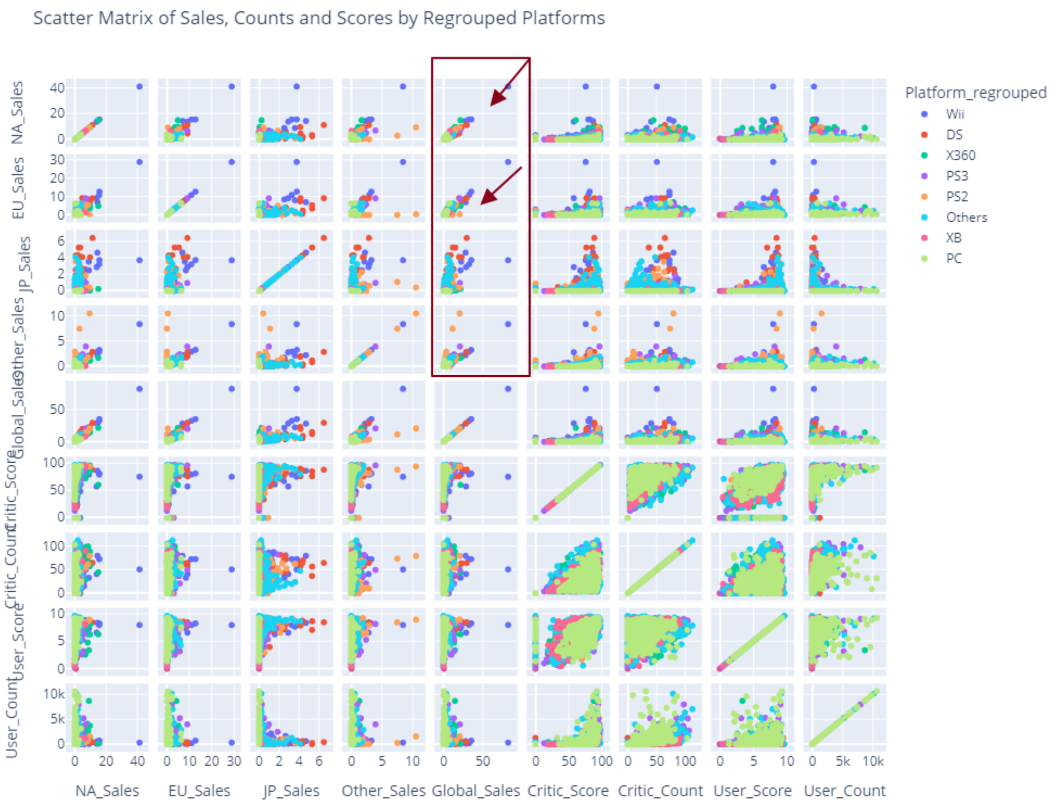


Fig 5: Scatterplot showing relationship between the features of the video game dataset

The scatter plot matrix also showed clearly at a glance how all of the numerical feature relate with one another. It is clear as seen highlighted with arrow that NA Sales, EU_Sales, Other_Sales and JP_Sales correlates with Global sales, unlike others such as Critic_Score, Critic_Count, User_Score and User_Count.

**1.3. Inferences from EDA:** From the analysis, variables we can employ for machine learning can be seen, however the presence of outliers could hinder training the model well, hence treating outliers have to be deployed.

**1.4. Removing Outliers:** This was done by creating a function that clips outlier values to the upper and lower whisker, using the principle from the formula below:

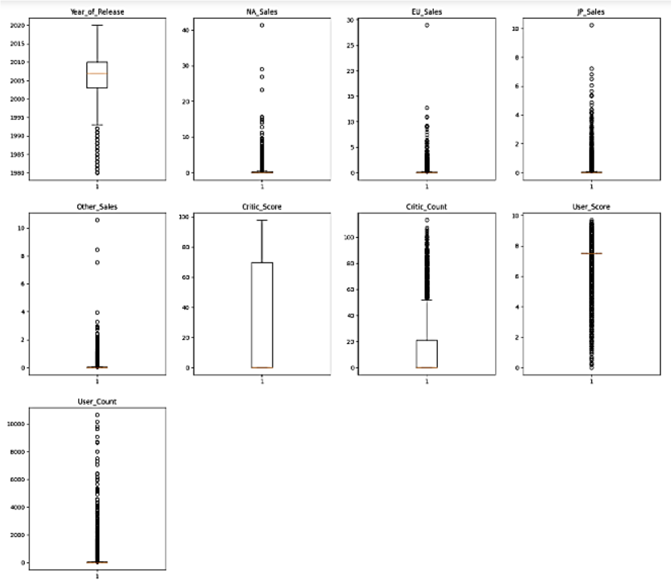


*Fig 6: Boxplot Showing presence of outliers*

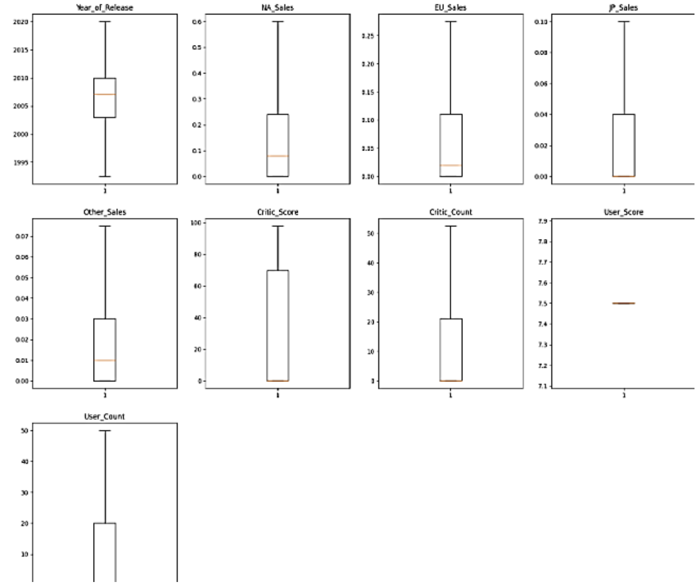


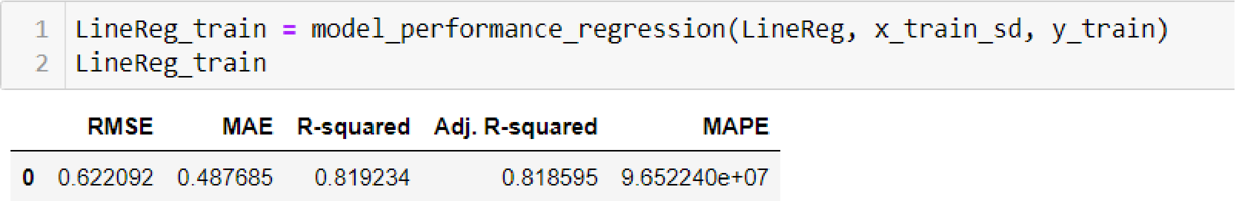
*Fig 7: Boxplot of numerical data after removal of Outliers*

## 2.0. Deployment of Models and Selection of Best Performing Model

With these preliminary analysis done, models were then deployed accordingly and the following questions were answered to form a basis for discussion for this report

**QUESTION A: Which of the variables in the video game dataset or a combination of them best predicts "global sales" of video games and why? Provide quantitative justifications for your answers.**

**ANSWER:** Multiple linear regression model was deployed to determine the extent to which the variables predict the global sales. The following were the quantitative results showing the performance of the model



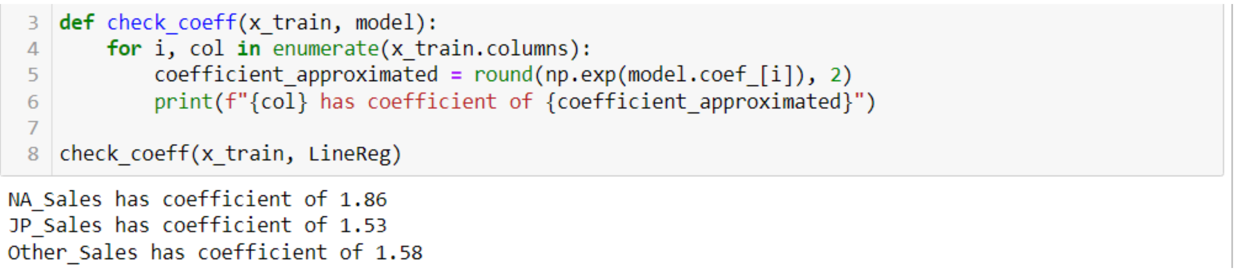
```
LineReg_train = model_performance_regression(LineReg, x_train_sd, y_train)
LineReg_train
```

|   | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| **0** | 0.622092 | 0.487685 | 0.819234 | 0.818595 | 9.652240e+07 |

*Fig 8: Model Performance metrics for Linear Regression Model*

From the above figure, the metrics seems to perform to show that it predicted well judging from the RMSE and MAE values.

The coefficient of the variable where obtained using a function as shown in Fig(), the variable alongside their coefficient were printed to identify good predictors, the higher and positive the coefficient, the more the variable have a good predictive influence on the sales



```python
def check_coeff(x_train, model):
    for i, col in enumerate(x_train.columns):
        coefficient_approximated = round(np.exp(model.coef_[i]), 2)
        print(f"{col} has coefficient of {coefficient_approximated}")

check_coeff(x_train, LineReg)
```

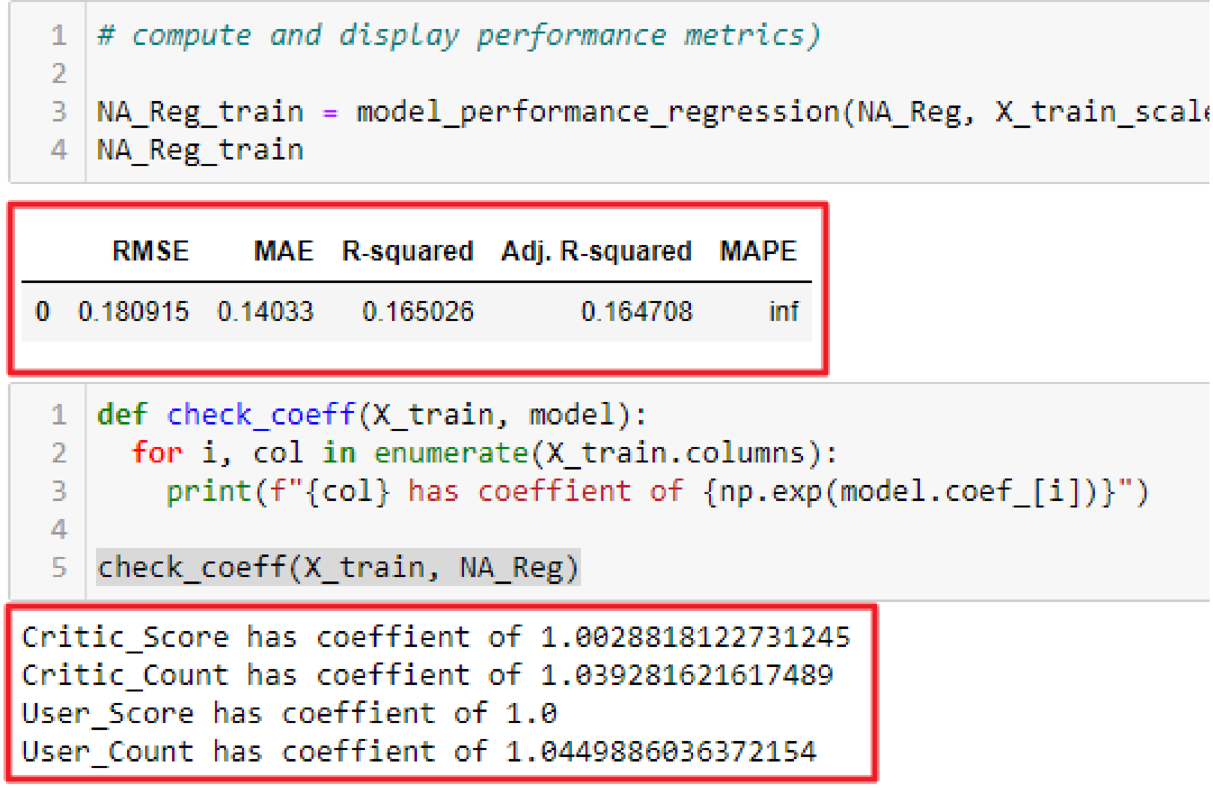
```
NA_Sales has coefficient of 1.86
JP_Sales has coefficient of 1.53
Other_Sales has coefficient of 1.58
```

*Fig 9: Coefficients of NA, JP and Others*

From the above figure, NA_Sales, JP_Sale and Other_Sales best predicts the Global Sales

**QUESTION B: What effect will the number of critics and users as well as their review scores have on the sales of Video games in North America, EU and Japan?**

For North America Sales, Critic and User Score/Count was seen to have a weak impact on the NA_Sales, the co-efficient was for the 4 variables were approximately 1 as shown in the figure below:



```python
# compute and display performance metrics)

NA_Reg_train = model_performance_regression(NA_Reg, X_train_scale
NA_Reg_train
```

|   | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| 0 | 0.180915 | 0.14033 | 0.165026 | 0.164708 | inf |

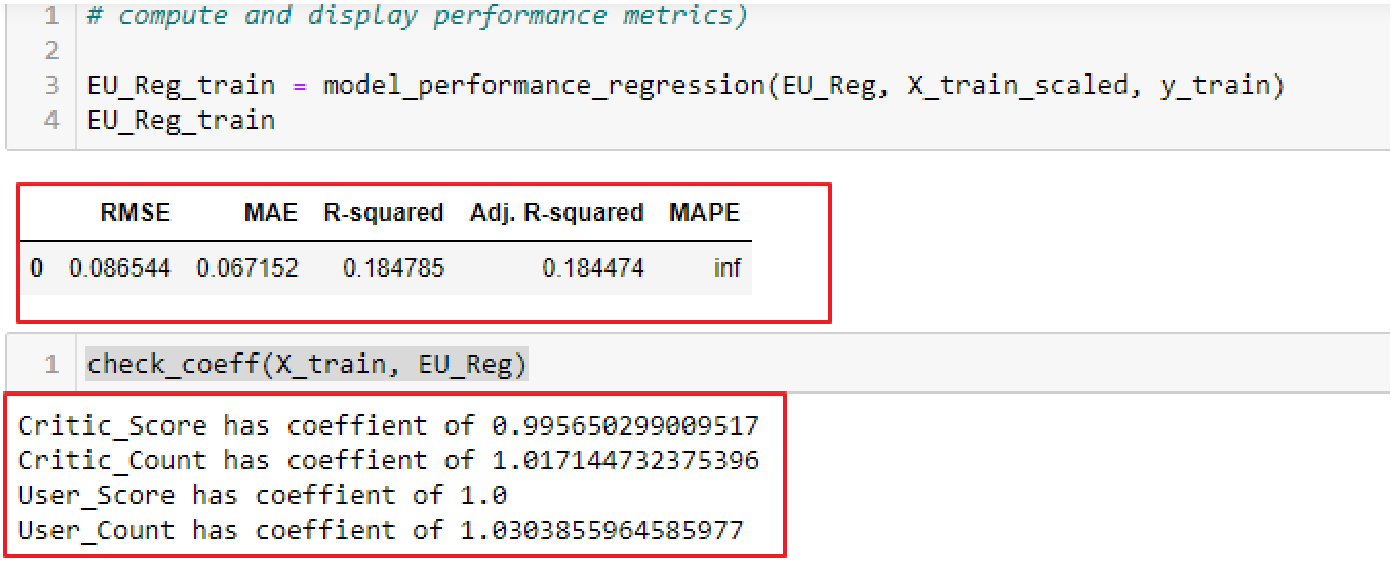
```python
def check_coeff(X_train, model):
  for i, col in enumerate(X_train.columns):
    print(f"{col} has coeffient of {np.exp(model.coef_[i])}")

check_coeff(X_train, NA_Reg)
```

```
Critic_Score has coeffient of 1.0028818122731245
Critic_Count has coeffient of 1.039281621617489
User_Score has coeffient of 1.0
User_Count has coeffient of 1.0449886036372154
```

*Fig 10: Model Performance and Coefficients of NA Sales using Critics and User Score and counts*

For EU Sales, Critic and User Score/Count was seen to have a weak impact on the EU_Sales, the co-efficient was for the 4 variables were approximately 1 as shown in the figure below



```python
# compute and display performance metrics)

EU_Reg_train = model_performance_regression(EU_Reg, X_train_scaled, y_train)
EU_Reg_train
```

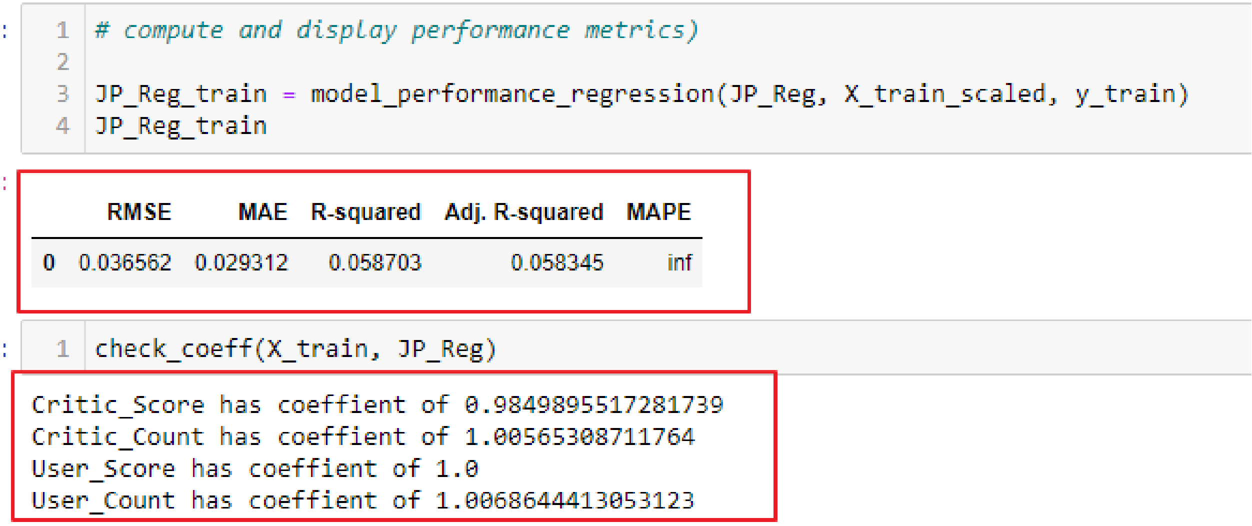
|   | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| 0 | 0.086544 | 0.067152 | 0.184785 | 0.184474 | inf |

```python
check_coeff(X_train, EU_Reg)
```

```
Critic_Score has coeffient of 0.995650299009517
Critic_Count has coeffient of 1.017144732375396
User_Score has coeffient of 1.0
User_Count has coeffient of 1.0303855964585977
```

*Fig 11: Model Performance and Cofficients of EU Sales using Critics and User Score and counts*

For JP Sales, Critic and User Score/Count was seen to have a weak impact on the JP_Sales, the co-efficient was for the 4 variables were approximately 1 as shown in the figure below



```python
# compute and display performance metrics)

JP_Reg_train = model_performance_regression(JP_Reg, X_train_scaled, y_train)
JP_Reg_train
```

|   | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| 0 | 0.036562 | 0.029312 | 0.058703 | 0.058345 | inf |

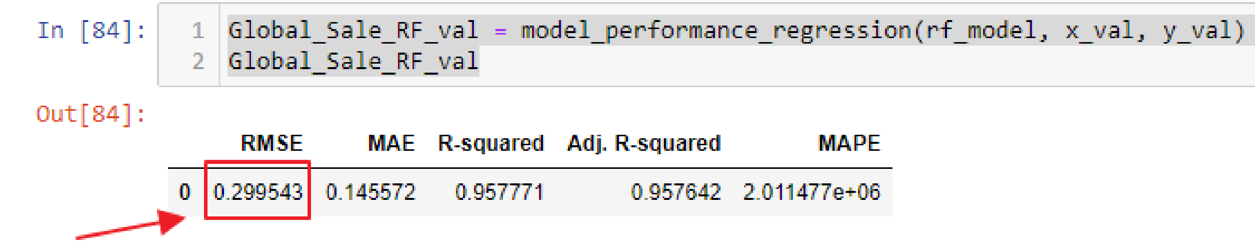
```python
check_coeff(X_train, JP_Reg)
```

```
Critic_Score has coeffient of 0.9849895517281739
Critic_Count has coeffient of 1.00565308711764
User_Score has coeffient of 1.0
User_Count has coeffient of 1.0068644413053123
```

*Fig 12: Model Performance and Cofficients of JP Sales using Critics and User Score and counts*

**QUESTION C: What propelled the choice of your regressor for this task? Aptly discuss with quantitative reasons!**

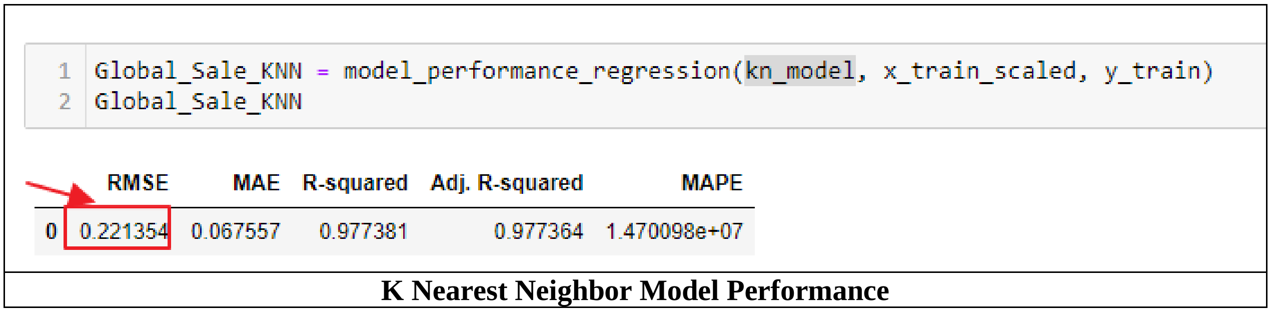
Tree based regressor Random Forest Regression, and other regressors which included: Gradient Boosting Regression, Lasso Regression, Support Vector and K Neighbours were deployed. Their performance metrics showed on the following table



```python
In [84]: Global_Sale_RF_val = model_performance_regression(rf_model, x_val, y_val)
         Global_Sale_RF_val
Out[84]:
```

|   | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| 0 | 0.299543 | 0.145572 | 0.957771 | 0.957642 | 2.011477e+06 |

**Random Forest Model Performance**



```python
Global_Sale_KNN = model_performance_regression(kn_model, x_train_scaled, y_train)
Global_Sale_KNN
```

|  | RMSE | MAE | R-squared | Adj. R-squared | MAPE |
|---|---|---|---|---|---|
| 0 | 0.221354 | 0.067557 | 0.977381 | 0.977364 | 1.470098e+07 |

**K Nearest Neighbor Model Performance**

*Fig 12: Model performance comparison between Random Forest and K-Nearest Neighbor*

From the table above, it can be seen that KNN Model is the choicest model as it has the lowest error 0.22 and the highest r2 score of 0.98. This performance is followed closely by that of Random Forest having error of 0.29 and r2 score of 0.96.

## QUESTION D: USE ALL THE RELEVANT CATEGORICAL VARIABLES IN THE VIDEO GAME DATASET AS THE TARGET VARIABLE AT EACH INSTANCE AND DETERMINE WHICH OF THE VARIABLES PERFORMED BEST IN CLASSIFYING THE DATASET. EXPLAIN YOUR FINDINGS.

**ANSWER:** Random Forest Classifier was used to test the impact of the following categorical variable:

1. Rating,
2. Genre and
3. Platform Regrouped

From the EDA and visualization done earlier, the above 3 categorical variables were selected as they have not too large number of unique elements. The 'Platform_Regrouped' variable (as shown in Fig) was created by selecting elements that do not have their number of appearance more than 700.

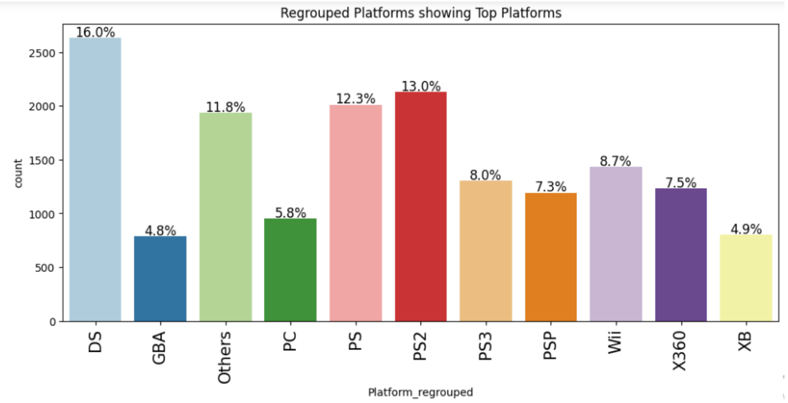


*Fig 13: Barplot showing 'Platform' elements regrouped to show those more than 700*

Elements in the variable ‘Rating’ were selected based on their high occurrence as shown in the barplot below, those in the category of Unknown were not used.

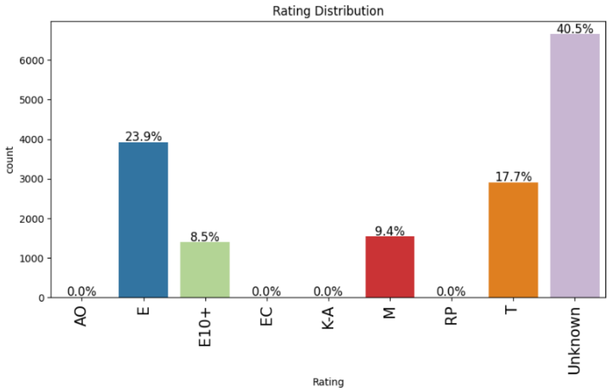


*Fig 13: Barplot showing ‘Rating’ elements grouped according to percentage of their number.*

Genre has 12 elements as seen in the barplot below, hence all was used in the training the categorical model.

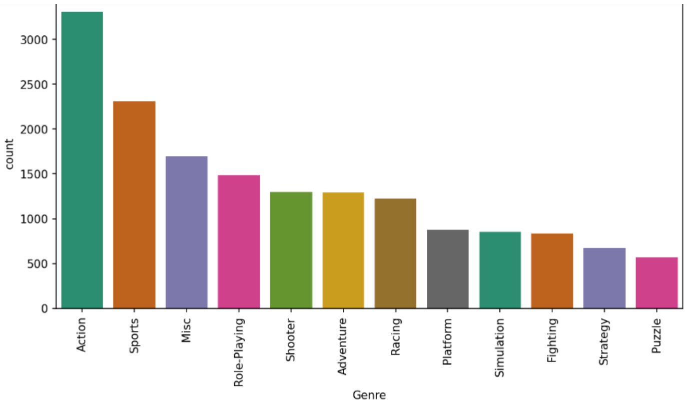
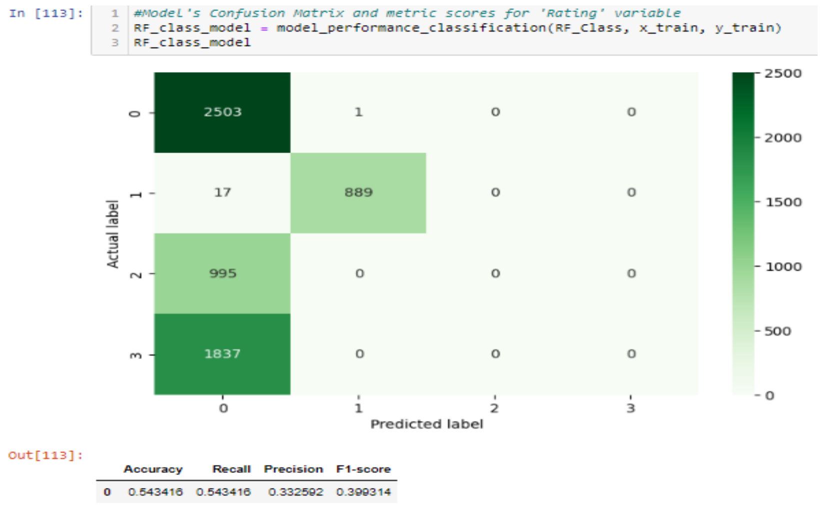


*Fig 13: Barplot showing ‘Rating’ elements grouped according to the number of their occurrence.*

The model used was Random Forest Classifier and it gave the following output which was visualized using confusion matrix and metrics using F1 Score as show below respectively for Rating, Platform and Genre

## RATING



```
In [113]:
#Model's Confusion Matrix and metric scores for 'Rating' variable
RF_class_model = model_performance_classification(RF_Class, x_train, y_train)
RF_class_model
```

| Actual label \ Predicted label | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 2503 | 1 | 0 | 0 |
| 1 | 17 | 889 | 0 | 0 |
| 2 | 995 | 0 | 0 | 0 |
| 3 | 1837 | 0 | 0 | 0 |

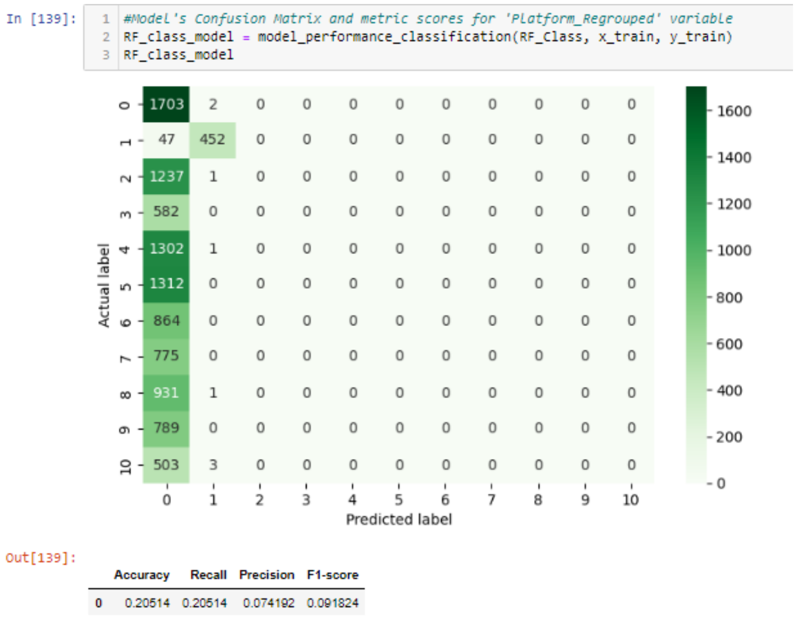
Out[113]:

|   | Accuracy | Recall | Precision | F1-score |
|---|---|---|---|---|
| 0 | 0.543416 | 0.543416 | 0.332592 | 0.399314 |

*Fig 14: Confusion Matrix for ‘Rating’*

From the confusion matrix above and F1 score, it can be seen that the model could not categorize the data correctly, as only rating '0'→E and '1'→ T rating were properly grouped, this obviously is because Rating 'E' and 'T' dominates by number. The fact that the F1 score is 0.39 shows that the model did not perform well.

### PLATFORM (Regrouped)



In [139]:
```
#Model's Confusion Matrix and metric scores for 'Platform_Regrouped' variable
RF_class_model = model_performance_classification(RF_Class, x_train, y_train)
RF_class_model
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1703 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 47 | 452 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1237 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 582 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1302 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 1312 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 864 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 775 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 931 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 789 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 503 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

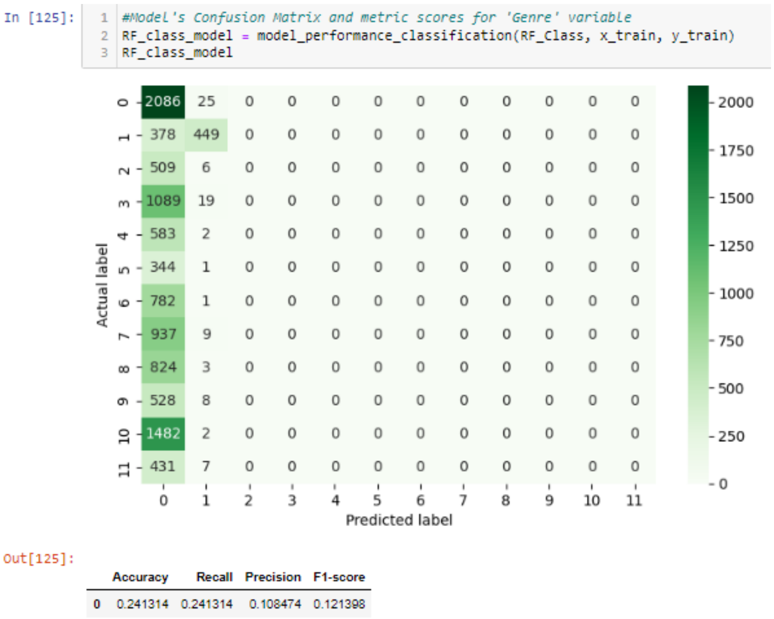
Out[139]:

| | Accuracy | Recall | Precision | F1-score |
|---|---|---|---|---|
| 0 | 0.20514 | 0.20514 | 0.074192 | 0.091824 |

*Fig 15: Confusion Matrix for 'Platform'*

The confusion matrix only could classify two of the elements, the model F1 score was 0.09, showing that the data could not be classified based on the parameters used

### GENRE



In [125]:
```
#Model's Confusion Matrix and metric scores for 'Genre' variable
RF_class_model = model_performance_classification(RF_Class, x_train, y_train)
RF_class_model
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2086 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 378 | 449 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 509 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1089 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 583 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 344 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 782 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 937 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 824 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 528 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 1482 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 431 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Out[125]:

| | Accuracy | Recall | Precision | F1-score |
|---|---|---|---|---|
| 0 | 0.241314 | 0.241314 | 0.108474 | 0.121398 |

*Fig 16: Confusion Matrix for 'Genre'*

The confusion matrix only could classify two of the elements, the model F1 score was 0.12, showing that the data could not be classified based on the parameters used.

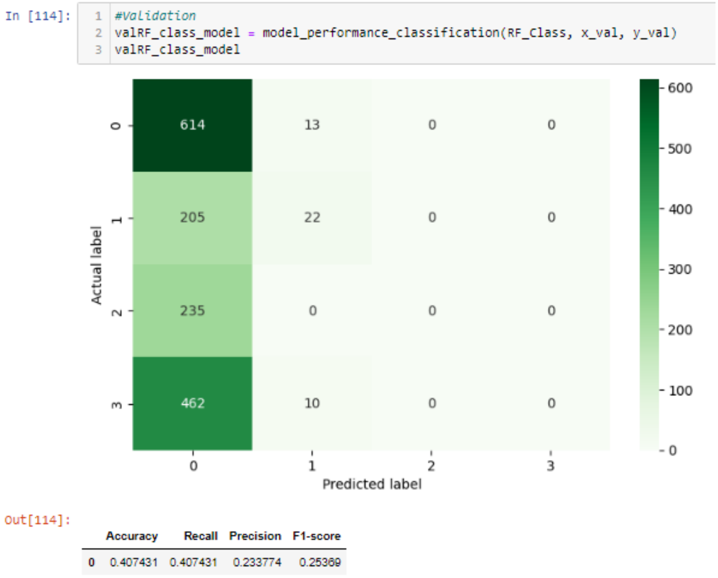
## QUESTION E: HOW DID YOU CHECK WHETHER YOUR MODELS DID NOT OVERFIT?

Using cross validation, some set of the data was used to check that the model did not overfit by comparing the metric scores and confusion matrix with the test data performance (the accuracy, precision and F1 scores), if the their performance are close to each other, then the model generalized well, however if way too below the model's metrics then it is overfitting and vice versa. Also, with regards with Random Forest Classification, the hyperparameters were left in default mode to avoid overfitting.

The following cross validation metrics interpreted below show if the model overfitted, generalized well or underfitted:

### RATING

The cross validation for 'Platform' set metrics as shown below ***showed that the model was underfitting*** as the metrics were way lower than the test set.



```
In [114]:
#Validation
valRF_class_model = model_performance_classification(RF_Class, x_val, y_val)
valRF_class_model
```

| Actual label \ Predicted label | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 614 | 13 | 0 | 0 |
| 1 | 205 | 22 | 0 | 0 |
| 2 | 235 | 0 | 0 | 0 |
| 3 | 462 | 10 | 0 | 0 |

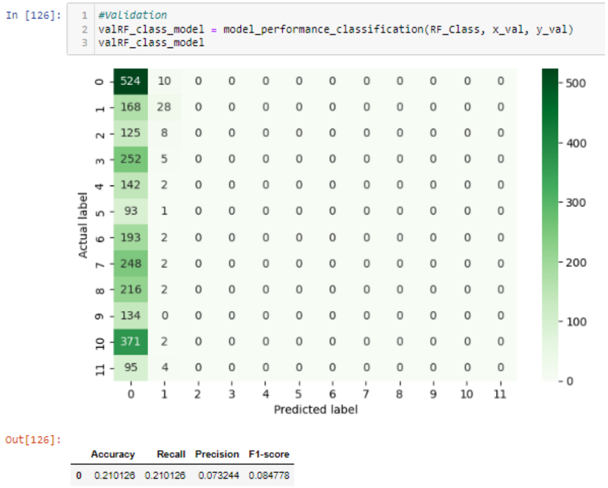
Out[114]:

| | Accuracy | Recall | Precision | F1-score |
|---|---|---|---|---|
| 0 | 0.407431 | 0.407431 | 0.233774 | 0.25369 |

*Fig 17: Confusion Matrix for 'Rating'*

### GENRE

The validation set metrics as shown below ***showed that the model was underfitting*** as the metrics were way too lower than the test set.



```
In [126]:
#Validation
valRF_class_model = model_performance_classification(RF_Class, x_val, y_val)
valRF_class_model
```

| Actual label \ Predicted label | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 524 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 168 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 125 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 252 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 142 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 93 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 193 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 248 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 216 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 134 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 371 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 95 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

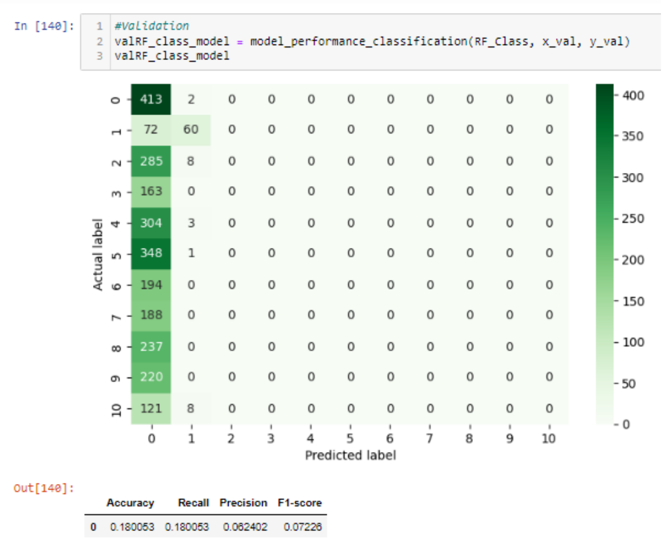
Out[126]:

| | Accuracy | Recall | Precision | F1-score |
|---|---|---|---|---|
| 0 | 0.210126 | 0.210126 | 0.073244 | 0.084778 |

*Fig 18: Confusion Matrix for 'Genre' for the validation data*

**PLATFORM**

The validation set metrics as shown below ***showed that the model was not overfitting*** as the metrics were close to the test set.



```
In [140]:
#Validation
valRF_class_model = model_performance_classification(RF_Class, x_val, y_val)
valRF_class_model
```

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 413 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 72 | 60 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 285 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 163 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 304 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 348 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 194 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 188 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 237 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 220 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 121 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
Out[140]:
   Accuracy    Recall  Precision  F1-score
0  0.180053  0.180053   0.062402   0.07226
```

*Fig 18: Confusion Matrix for 'Platform' for the validation data*

**QUESTION F: Can your classification models be deployed in practice based on their performances? Explain.**

The result so far is poor, the model was not able to predict the right classification, evidently from the F1 scores, most of them were less than 1%!, The model is struggling under the relationship of the video games to the features, hence we can deploy the classification model.

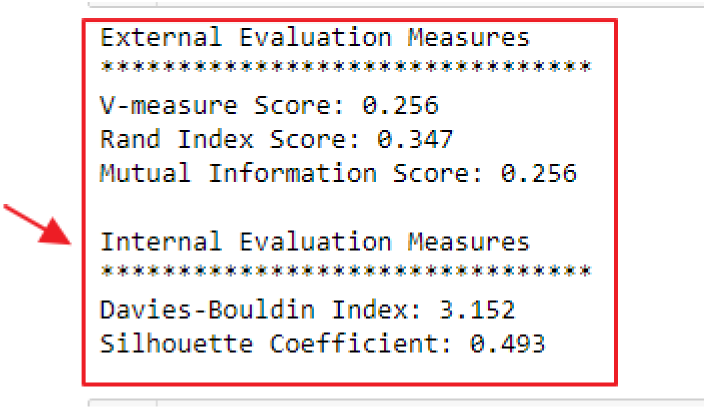
**QUESTION G: IN THE VIDEO GAME DATASET, USE A RELEVANT CATEGORICAL VARIABLE AND OTHER RELEVANT NON-CATEGORICAL VARIABLES TO FORM GROUPS AT EACH INSTANCE. BY EMPLOYING INTERNAL AND EXTERNAL EVALUATION METRICS, DETERMINE WHICH CATEGORICAL VARIABLE BEST DESCRIBES THE GROUPS FORMED**

***Table 2: Comparison between the Internal and External Measures of KNN and DBScan of the Categorical variables: Rating, Genre, Platform and Developer***

|  | **Variables** |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| **External Evaluation Measures** | Rating |  | Genre |  | Platform (Regrouped) |  | Developer |  |
|  | KMeans | DBSCAN | KMeans | DBSCAN | KMeans | DBSCAN | KMeans | DBSCAN |
| V-measure Score | 0.256 | 0.212 | 0.023 | 0.062 | 0.048 | 0.135 | 0.195 | 0.241 |
| Rand Index Score | 0.347 | -0.003 | 0.008 | 0.024 | 0.031 | 0.008 | 0.353 | -0.146 |
| Mutual Information Score | 0.256 | 0.206 | 0.023 | 0.048 | 0.047 | 0.122 | 0.160 | 0.091 |
| **Internal Evaluation Measures** |  |  |  |  |  |  |  |  |
| Davies-Bouldin Index | 3.152 | 3.131 | 3.152 | 3.131 | 3.152 | 3.131 | 3.152 | 3.131 |
| Silhouette Coefficient | 0.493 | -0.412 | 0.493 | -0.412 | 0.493 | -0.412 | 0.493 | -0.412 |

From the table above, it can be seen that mostly KMeans has higher scores than DBScan, however it can also be seen, generally, that the scores are very low, meaning that the variables seems not be a good determinant for the target variable.

Nevertheless, the variable that seem to be the best is *Rating* as all scores for internal measures are higher as highlighted below



```
External Evaluation Measures
********************************
V-measure Score: 0.256
Rand Index Score: 0.347
Mutual Information Score: 0.256

Internal Evaluation Measures
********************************
Davies-Bouldin Index: 3.152
Silhouette Coefficient: 0.493
```

## CONCLUSION

From the various regression and classification model, we can see comparison between using different variable to see which performed well in determining the target. The comparison showed that Other Sales and NA_sales determined the Global Sales more than the rest. For categorical variables, the ‘Rating’ variable did well in determining the target variable.

Out of all the models, Radom Forest Regressor and KNN classifier did well predicting the target and classifying well.