1. *Problem Selection*

For the final project we plan to gather information about video game sales, using multiple datasets found from scraping websites/downloading pre-existing datasets and merging them together. Using exploratory data analysis, the information that we will obtain from the data will help us predict global video game sales, so that game development businesses can decide which factors (publisher, ratings, season, genre, etc.) are the ones that affect sales the most.

For this proposal, we are predicting the video games overall global sales based on which factors will have the largest impact on the sales being high. For this project, we will focus on the use of regression analysis to determine the predictions of sales based on each individual sales around the world such as the US, Europe, Japan, etc., and we will also use clustering to determine which console is the most popular and what genre is the most played. Lastly, we will perform a cluster analysis to determine which yearly season games garner the most sales from. The regression analysis could help us determine which variables factor more into the global sales. For clustering, we want to find a pattern in the data to see how similar or how different the distances between them are. We want to explore the distances between points in the consoles, genres, seasons, etc.

2. *Data Collection*

There's two datasets we're using, *vgsales.csv* and *vgratings.csv*. The former dataset was obtained from the following source: https://www.kaggle.com/gregorut/videogamesales. The dataset has **11 variables** and **16598 entries**. The types of data include **int64**, **object**, and **float64**.

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

The latter dataset was obtained from scraping (see *metacritic_scraper.py* for the code we used to perform this task) the following website: https://www.metacritic.com/browse/games/score/metascore/all/all/filtered?page=0. We figured the former dataset wasn't suitable for this assignment alone, so we wanted to scrape more data (ratings, specific release date, etc.). The dataset has **6 variables** and **18009 entries**. The types of data include **float64**, **object**, and **int64**.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18009 entries, 0 to 18008
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Number        18009 non-null  float64
 1   Name          18009 non-null  object
 2   Platform      18009 non-null  object
 3   Release_Date  18009 non-null  object
 4   Metascore     18009 non-null  int64
 5   Userscore     18009 non-null  object
dtypes: float64(1), int64(1), object(4)
memory usage: 844.3+ KB
```

Looking ahead, the merged dataset has **14 variables** and **5733 entries**. The types of data include **float64**, **int64** and **object**.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5733 entries, 0 to 5732
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Name           5733 non-null   object
 1   Platform       5733 non-null   object
 2   Genre          5733 non-null   object
 3   Publisher      5733 non-null   object
 4   NA_Sales       5733 non-null   float64
 5   EU_Sales       5733 non-null   float64
 6   JP_Sales       5733 non-null   float64
 7   Other_Sales    5733 non-null   float64
 8   Global_Sales   5733 non-null   float64
 9   Release_Date   5733 non-null   object
 10  Metascore      5733 non-null   int64
 11  Userscore      5733 non-null   float64
 12  Season         5733 non-null   object
 13  Season_Number  5733 non-null   int64
dtypes: float64(6), int64(2), object(6)
memory usage: 671.8+ KB
```

3. *Data Preparation*

Our goal in this step was to prepare the datasets for merging. From the general explorations we did on our data, we found that the best **foreign keys** to merge the two datasets on were **'Name'**, **'Platform'** and **'Year'**. As such, we had to address any inconsistencies in the foreign keys as well as deal with any irrelevant or redundant variables and deal with any missing values.

For the **sales table** (from *vgsales.csv*), we **first** had to match the 'Platform' abbreviated data with the 'Platform' unabbreviated data from the ratings table as they were inconsistent.

```
Distinct platforms in sales_data:  ['Wii' 'NES' 'GB' 'DS' 'X360' 'PS3' 'PS2' 'SNES' 'GBA' '3DS' 'PS4' 'N64'
 'PS' 'XB' 'PC' '2600' 'PSP' 'XOne' 'GC' 'WiiU' 'GEN' 'DC' 'PSV' 'SAT'
 'SCD' 'WS' 'NG' 'TG16' '3DO' 'GG' 'PCFX']
Distinct platforms in ratings_data:  ['Nintendo 64' 'PlayStation' 'PlayStation 3' 'Dreamcast' 'Xbox 360' 'Wii'
 'Xbox One' 'Switch' 'PlayStation 2' 'PlayStation 4' 'GameCube' 'Xbox'
 'PC' 'Wii U' 'Game Boy Advance' '3DS' 'DS' 'PlayStation Vita'
 'PlayStation 5' 'PSP' 'Xbox Series X' 'Stadia']
```

Next, we had to address the irrelevant variable 'Rank' by dropping it. The variable 'Rank' is irrelevant because we're probably not going to end up using it, and if we were, we could just re-calculate it, as it's based on the 'Global_Sales' variable. **Then**, we had to address the missing values in the 'Publisher' variable. Since there were **only 58 missing values** (out of a dataset of size 16598), we decided to replace the missing values with "Unknown". **On the other hand**, for the 'Year' variable, we opted to outright remove the observations with missing values. Like the previous variable, there were only a few missing values (**271**), however we had no way of being able to 'predict' these missing values, so we just went with the next best thing. **Lastly**, we had to resolve one minor inconsistency in that we had to convert the 'Year' variable from type float64 to type int64, as there's no reason in having it be type float64 in the first place.

For the **ratings table** (from *vgratings.csv*), we **first** decided to drop the 'Number' column as it's irrelevant. We originally scraped this info for easing the scraping process, and as such, we no longer needed it, and if we did, we could just re-calculate it using the 'Metascore' variable, as it's based on it. **Next**, we noticed that although there weren't any missing values, the variable 'Userscore' had "tbd" for many entries.

	Number	Name	Platform	Release_Date	Metascore	Userscore
count	18009.000000	18009	18009	18009	18009.000000	18009
unique	NaN	11820	22	4366	NaN	95
top	NaN	Cars	PC	November 14, 2006	NaN	tbd
freq	NaN	9	4605	48	NaN	1277
mean	9005.000000	NaN	NaN	NaN	70.405408	NaN
std	5198.894834	NaN	NaN	NaN	12.396993	NaN
min	1.000000	NaN	NaN	NaN	11.000000	NaN
25%	4503.000000	NaN	NaN	NaN	63.000000	NaN
50%	9005.000000	NaN	NaN	NaN	72.000000	NaN
75%	13507.000000	NaN	NaN	NaN	79.000000	NaN
max	18009.000000	NaN	NaN	NaN	99.000000	NaN

We decided to consider these entries as having missing values for 'Userscore'. As such, we converted all "tbd" entries to NaN. While we were at it, we also converted the 'Userscore' variable from type object to type float64, as it was only originally type object because "tbd" was a string. **Next**, we created a new variable 'Year' for the purpose of being able to merge this dataset with the sales_data. We did this by extracting the year from the 'Release_Date' variable, which is in the format MM DD, YYYY. **Lastly**, we made the most important decision of removing the NaN values that were converted from the "tbd" entries in the 'Userscore' variable.

	Name	Platform	Release_Date	Metascore	Userscore
497	Madden NFL 2005	GameCube	August 9, 2004	90	NaN
924	Tiger Woods PGA Tour 2005	GameCube	September 20, 2004	88	NaN
1220	NASCAR 2005: Chase for the Cup	Xbox	August 31, 2004	86	NaN
1410	Moto Racer Advance	Game Boy Advance	December 5, 2002	86	NaN
2109	Pinball FX 2: Marvel Pinball - Vengeance and V...	Xbox 360	December 13, 2011	84	NaN
...	...	...	...	...	...
17817	Jackass the Game	DS	January 8, 2008	35	NaN
17840	King of Clubs	Wii	August 4, 2008	35	NaN
17900	Jenga World Tour	DS	November 13, 2007	32	NaN
17915	Dream Chronicles	PlayStation 3	November 23, 2010	31	NaN
17917	Smash 'N' Survive	PlayStation 3	February 22, 2012	31	NaN

1277 rows × 5 columns

This meant removing **1277 entries**. Typically, this isn't recommended practice, but our reason for doing this instead of dropping the variable outright is because the variable will prove to be very valuable for when we construct our linear regression model.

At this point, it's okay for us to merge both datasets into one. We opted to lowercase all game names to take into account any inconsistencies in the naming conventions that both datasets have. We then performed an inner join on the variables **'Name'**, **'Platform'** and **'Year'**.

Since we no longer needed to merge anymore, the 'Year' variable became redundant (as we have the year encapsulated in the 'Release_Date' variable), and as such, we safely dropped it from the merged dataset.

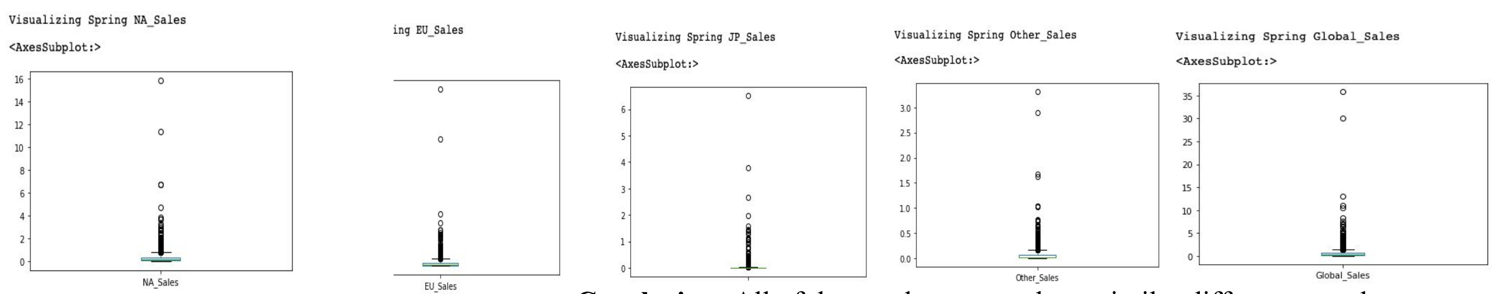
4. *Data Exploration*

To explore the data we computed the mean of all of the sales, metascores, and user scores individually for each of the seasons.Then we performed an hypothesis test to compare all of the sales in each of the seasons, using the f_oneway ANOVA hypothesis test. This test is important to explore the dataset, because it allows us to perform a statistical hypothesis test on two or more variables, and will return the test statistic and the p-value. Additionally we plotted all of the sales, metascores, and user scores separately for each of the seasons using box plots and displayed descriptive statistics.

The spring mean for the NA_Sales variable found that the mean was **0.348 (in millions)**, for the variable EU_Sales the mean was **0.214 (in millions)**, JP_Sales mean was **0.050 (in millions)**, Other_Sales mean was **0.071 (in millions)**, and Global_Sales mean was **0.683 (in millions)**, therefore, the highest mean for the spring season (not including the global sales) was the NA_Sales variable. The hypothesis test for the sales in the spring found that the t-test statistic was **96.749**, and the p-value was **0.00**, this means that there are significant differences between the variables. The summer mean for the NA_Sales variable found that the mean was **0.379 (in millions)**, for the variable EU_Sales the mean was **0.173 (in millions)**, JP_Sales mean was **0.039 (in millions)**, Other_Sales mean was **0.061 (in millions)**, and Global_Sales mean was **0.65 (in millions)**, therefore, the highest mean for the summer season (not including the global sales) was NA_Sales. The hypothesis test for the sales in the summer found that the t-test statistic was **112.122**, and the p-value was **0.00**, this means that there are significant differences between the variables. The autumn mean for the NA_Sales variable found that the mean was **0.51 (in millions)**, for the variable EU_Sales the mean was **0.31 (in millions)**, JP_Sales mean was **0.057 (in millions)**, Other_Sales mean was **0.109 (in millions)**, and Global_Sales mean was **1.00 (in millions)**, therefore, the highest mean for the autumn season (not including the global sales) was NA_Sales. The hypothesis test for the sales in the autumn found that the t-test statistic was **241**, and the p-value was **0.00**, this means that there are significant differences between the variables. The winter mean for the NA_Sales variable found that the mean was **0.30 (in millions)**, for the variable EU_Sales the mean was **0.17 (in millions)**, JP_Sales mean was **0.049 (in millions)**, Other_Sales mean was **0.05 (in millions)**, and Global_Sales mean was **0.59 (in millions)**, therefore, the highest mean for the autumn season (not including global sales) was NA_Sales. The hypothesis test for the sales in the winter found that the t-test statistic was **116**, and the p-value was **0.00**, this means that there are significant differences between the variables.

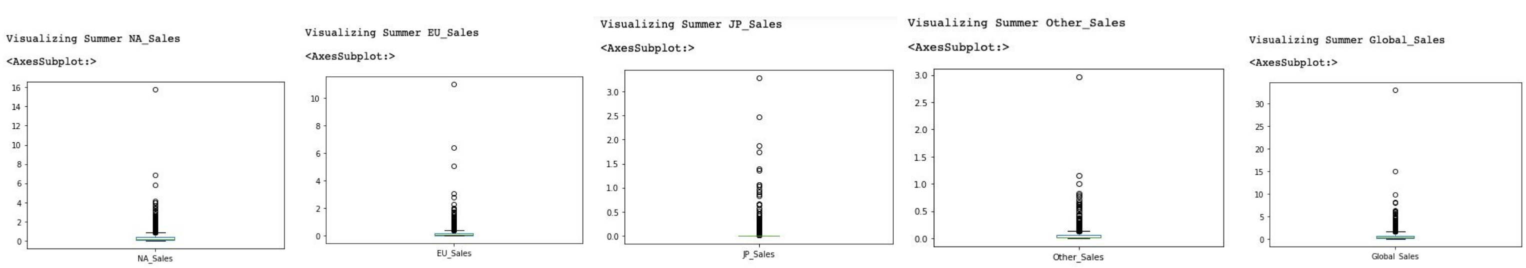
Next we computed the mean and hypothesis test on metascores and user scores separately for each of the seasons. The spring mean for metascores was **70.6 (out of 100)**, the summer mean was **70.2 (out of 100)**, the autumn mean was **71.9 (out of 100)**, and the winter mean for metascores was **70.3 (out of 100)**, so the highest mean was the autumn metascores. The hypothesis test for all of the seasons metascores found that the t-test statistic was **4.490**, and the p-value was **.0019**, this shows that the variables continue to be significantly different. The spring mean for user scores was **7.15 (out of 10)**, the summer mean was **7.11 (out of 10)**, the autumn mean was **7.2 (out of 10)**, and the winter mean for user scores was **7.2 (out of 10)**, the highest mean appears in the autumn and winter seasons. The hypothesis test for all of the seasons user scores found that the t-test statistic was **1.804**, and the p-value was **0.144**, this shows that the variables have similar distributions. The most important variables for data analysis that we found are NA_Sales, EU_Sales, JP_Sales, Other_Sales, and Global_Sales.

Spring Sales Comparison

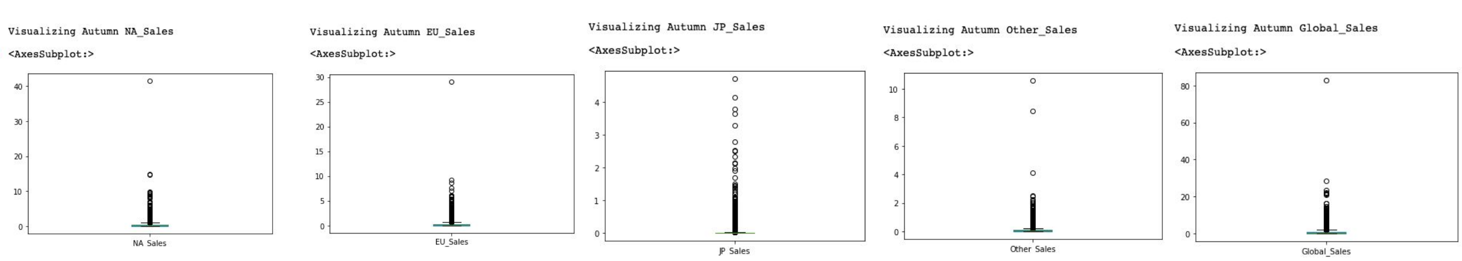


Conclusion: All of the graphs seem to have similar differences and they are noticeably, therefore, sales would be helpful predictors.

Summer Sales Comparison

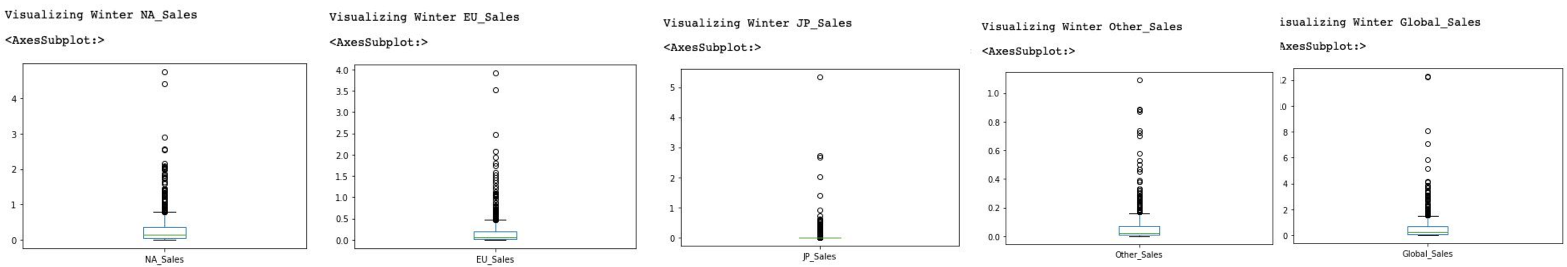


Autumn Sales Comparison



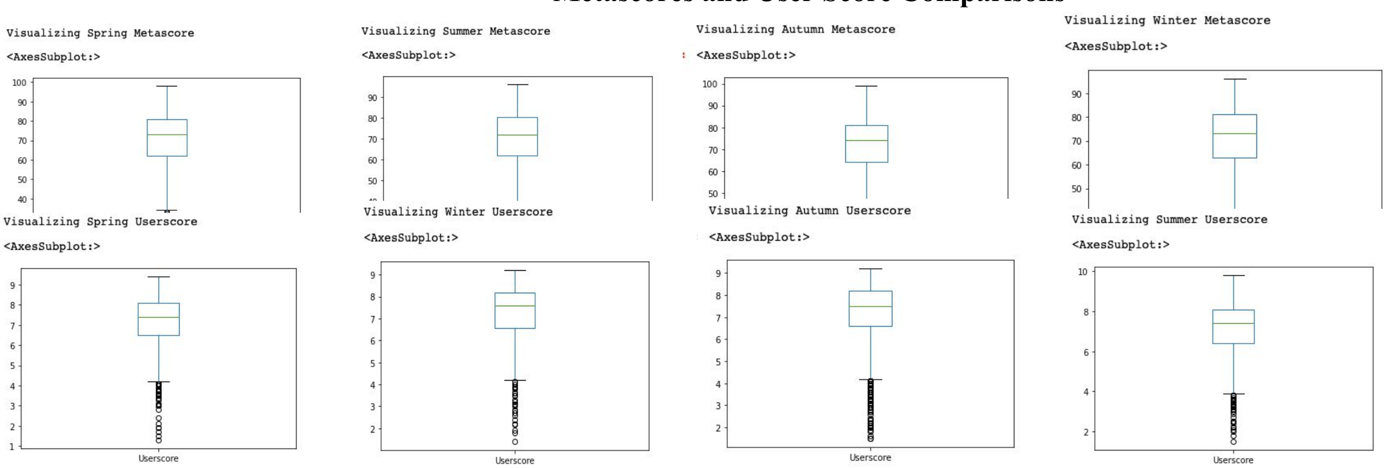
Conclusion: The graphs have clear noticeable differences, therefore, the sales variables continue to be good predictors.

Winter Sales Comparison



Conclusion: The variables continue to have noticeable differences, therefore, we conclude that the sales variables are good predictors.

Metascores and User Score Comparisons



Conclusion: The metascore and user score outlier differences are barely noticeable, therefore, metascore and user score would not be good predictors.

5. *Data Modeling*

To start off, we created dummy variables for some of our categorical data ('Platform', 'Genre' and 'Season'). We then partitioned the dataset using the holdout method. We also made sure to standardize each partition.

From the Lasso linear regression model we built, every feature except 'NA_Sales' and 'EU_Sales' results in a coefficient of 0. As such, these features resulting in a coefficient of 0 should be dropped. Regardless, about 98% of our variance is being explained by our model.

```
[ 0.72576583  0.55357641  0.          0.          0.          0.
 -0.          0.         -0.         -0.         -0.          0.
 -0.         -0.          0.          0.          0.          0.
 -0.          0.          0.         -0.          0.          0.
  0.         -0.         -0.          0.          0.         -0.
 -0.         -0.          0.         -0.          0.         -0.
  0.         -0.         -0.         -0.        ]
```

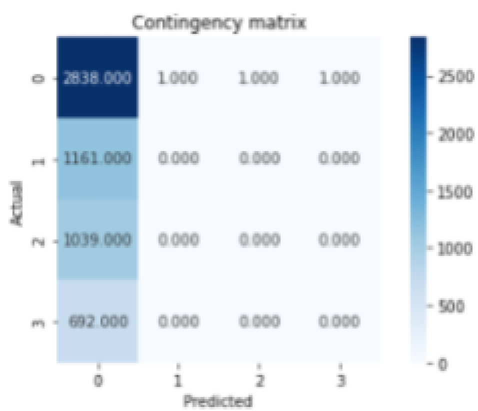
From the multi-linear regression model we built, given the R_squared of the model resulted in about 0.99, we'd like to say the model accurately fits the data. We built this model using the features 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales', 'Platform_PS2', 'Genre_Action' and 'Genre_Simulation'. These features are solid predictors for 'Global_Sales'. About 99% of our variance is being explained by our model.

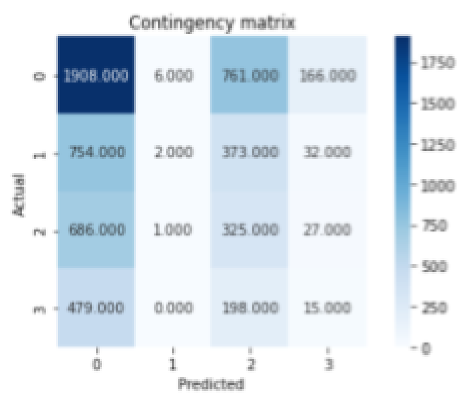
Now for clustering, we first factorize the variables from each category. Then, we standardize the dataset. All the clusters we are observing in this case use single linkage only. We first factorized each column that we chose to look at so that the text can turn into index numbers, so that we can cluster them together. When creating the clustering method we used the X_scaled that was standardized and linked them together with the single linkage. Then the clusters get set with how many clusters we need to categorize each case when we find or are given except for DBSCAN where it does not matter since it does it for you.

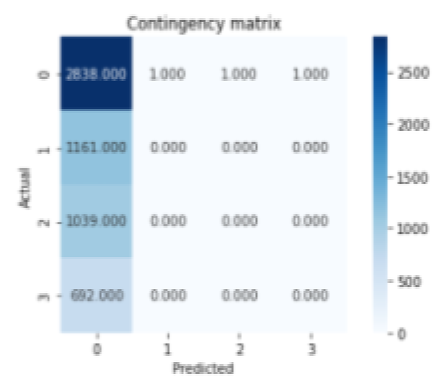
For the platform clustering, we found that there were 18 clusters that needed to be looked at due to the printing out the unique size of each column. The cluster method that had the highest precision and results is the DBSCAN method. The DBSCAN method for platform results is that the adjusted Rand index is -3.246, which is kind of low compared to Hierarchical Clustering who has the worst out of all clustering methods being -6.123. Thus, DBSCAN should be used to cluster platforms.

Next, we did clustering for genres. We factorized genres into 12 different variables. The results method clustering for genres is that DBSCAN method because it contains the highest coefficient silhouette and not super low adjusted Rand index. The values for the highest coefficient silhouette is 0.958, and the adjusted Rand index is -0.0001. Thus, the DBSCAN method is the best choice. However, the Hierarchical Clustering adjusted Rand index is positive being at 0.0001, and the coefficient silhouette is around 0.793 and the actual cluster is pretty not similar to the actual cluster. The results here have concluded that this cluster is good but it is not that accurate and precise.

Lastly, we did clustering for seasons. We factorized Seasons into 4 different variables. The cluster method that had the highest coefficient silhouette, and the decent amount to the precision for the adjusted Rand index is -0.0001, and the coefficient silhouette is 0.958. Although the K-mean adjusted Rand index is positive being is around -0.003, and the coefficient silhouette is around 0.374, the coefficient silhouette is much higher for DBSCAN method, which means these clusters are more cohesive and better separated than ones found in Hierarchical clustering and K-means. In all of the Contingency matrix, for each one of them did not predict any 1 with actual 1 at all, this means that when the clustering it was tested it was not that important factor and there was no pattern found.

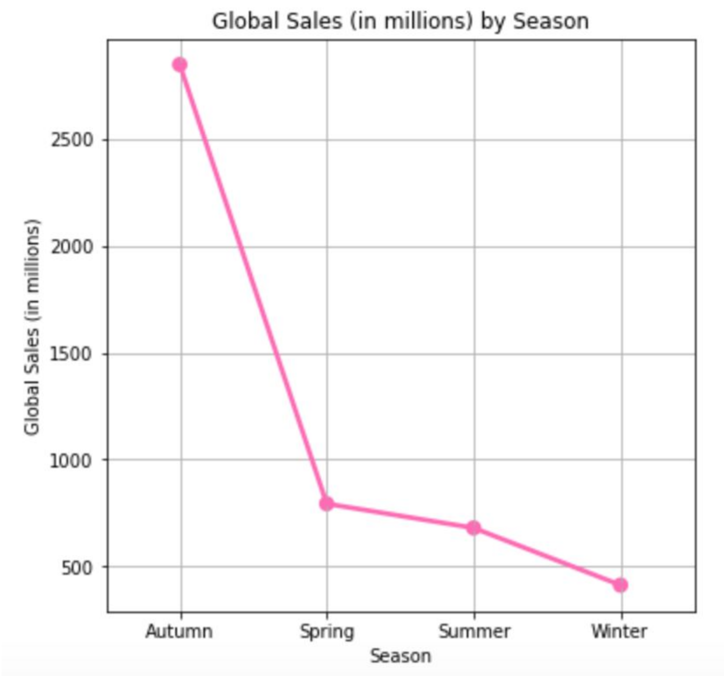






6. *<u>Presentation Results</u>*

From the linear regression model we built, we came to the conclusion that the driving features for global video game sales were ‘NA_Sales’, ‘EU_Sales’, ‘JP_Sales’, ‘Other_Sales’, ‘Platform_PS2’, ‘Genre_Action’, ‘Genre_Simulation’. The individual region sales were obviously going to cooperate toward the global sales (even our visualization results hinted at that), however; we only performed this analysis to see if there was even a small hope that some specific platform or genre would cooperate toward global sales, and to our surprise, we did find out such results. If a game was made for the PS2 console and it was a hybrid Action/Simulation game, our data suggests that it could potentially be successful sales wise. To our dismay, however; we didn't find any evidence that selling a specific game at a specific season would help drive up sales, despite the fact that the chart below shows what could have been a pattern. As you can see, Autumn sales seem to have the most sales out of all other seasons.



From our cluster analysis, we found that grouping by 'Platform' and 'Genre' produces the best results. As such, this indicates that games tend to be similarly successful depending on the Genre or Platform of the game. This makes some sense in that certain groups only buy certain genres of video games. An individual who primarily plays shooters will most likely buy other shooter games as well. As such, the user market is the same. This can be equally said about platforms in that certain users only buy for specific platforms.

Lastly, we want to make a note of what we could have done better. If we were to repeat our process, we would definitely try to find some evidence to suggest that maybe year cooperates toward overall game sales (instead of by season). Year would have been more specific and interesting to pursue.