Video Game Sales – Predictions

Jordan Navarra

Feb 2022

ITCS 5156 Machine Learning

Original paper:

“Video Games Sales Analysis: A Data Science Approach” (TM Geethanjali, 2020)

TM Geethanjali, Ranjan D, Swaraj HY, Thejaskumar MV, Chandana HP

May 5th, 2020

International Journal of Creative Research Thoughts (IJCRT)

Introduction-

The paper I based my work on is titled: "Video Games Sales Analysis: A Data Science Approach." (TM Geethanjali, 2020) By taking a dataset of video game sales ranging from the 1980s to modern day, we attempt to predict future sales. My motivation for choosing this field is that video games are a great passion of mine, as a whole it has been one of the largest growing industries for at least two decades, and continues to grow, and sales data is useful for marketing and predicting trends, whether you're a developer or a content creator. It is certainly relevant form a finance perspective to continue researching the *why* behind video games' successes.

The initial dataset holds 10 features:

	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
...	...	...	...	...	...	...	...	...	...	...
16593	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.00	0.00	0.01
16594	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.00	0.00	0.01
16595	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.00	0.00	0.01
16596	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.00	0.00	0.01
16597	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.00	0.01

16598 rows × 10 columns

The dataset holds some useful features, but some meaningless features and some redundant features as well. I found an altered version of the same dataset that includes more features, and I applied my own preprocessing to them to hopefully strengthen the model. Then I continued the in steps of the paper by using Linear Regression to predict North America Sales. I added extra models as well to see how they would perform.

Background-

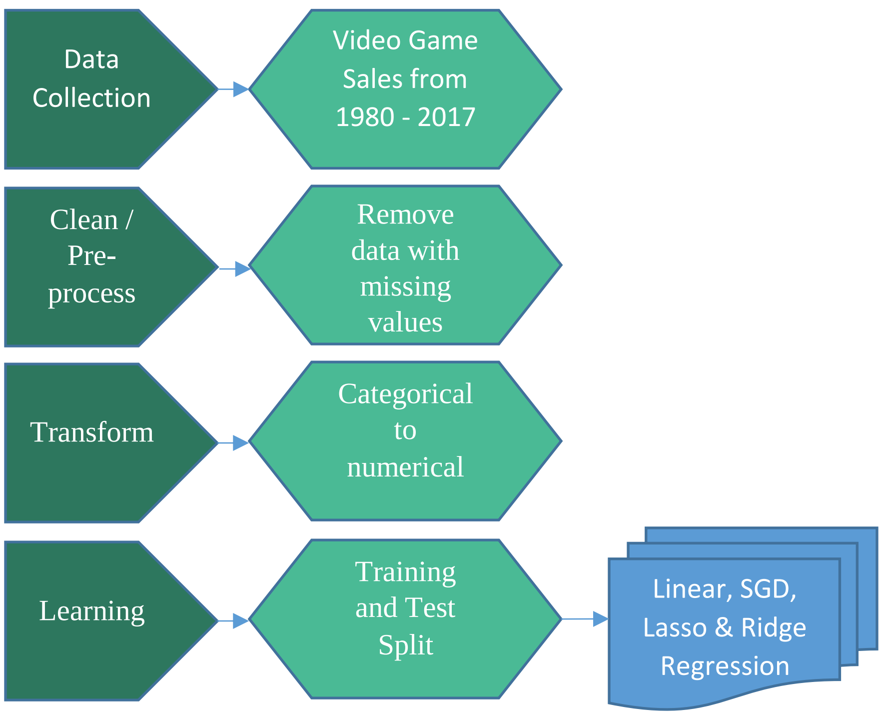
Kaggle had no shortage of Video game sales papers, including ones that use the same or slightly altered datasets. For 'EDA – Video Game Sales" by Murilao (MURILÃO, 2020), it was more of a visual and informational write-up rather than predictions using models. I chose this study because it analyzed the same data I found on Kaggle, and they make a good use of Data

Visualization. This paper is a great informational write-up, as it included histograms, calculated frequency of games, consoles/platforms and years, skews and variance.

In a similar study done by a student at Masaryk University in the Czech Republic he used Support Vector Machine, Regression Trees and Naïve Bayes to predict not only sales, but current player base. (Trněný, 2017) It seems his dataset was more robust, and his models and methods had the proper depth to make great predictions.

Methods and Experiments-

Framework:



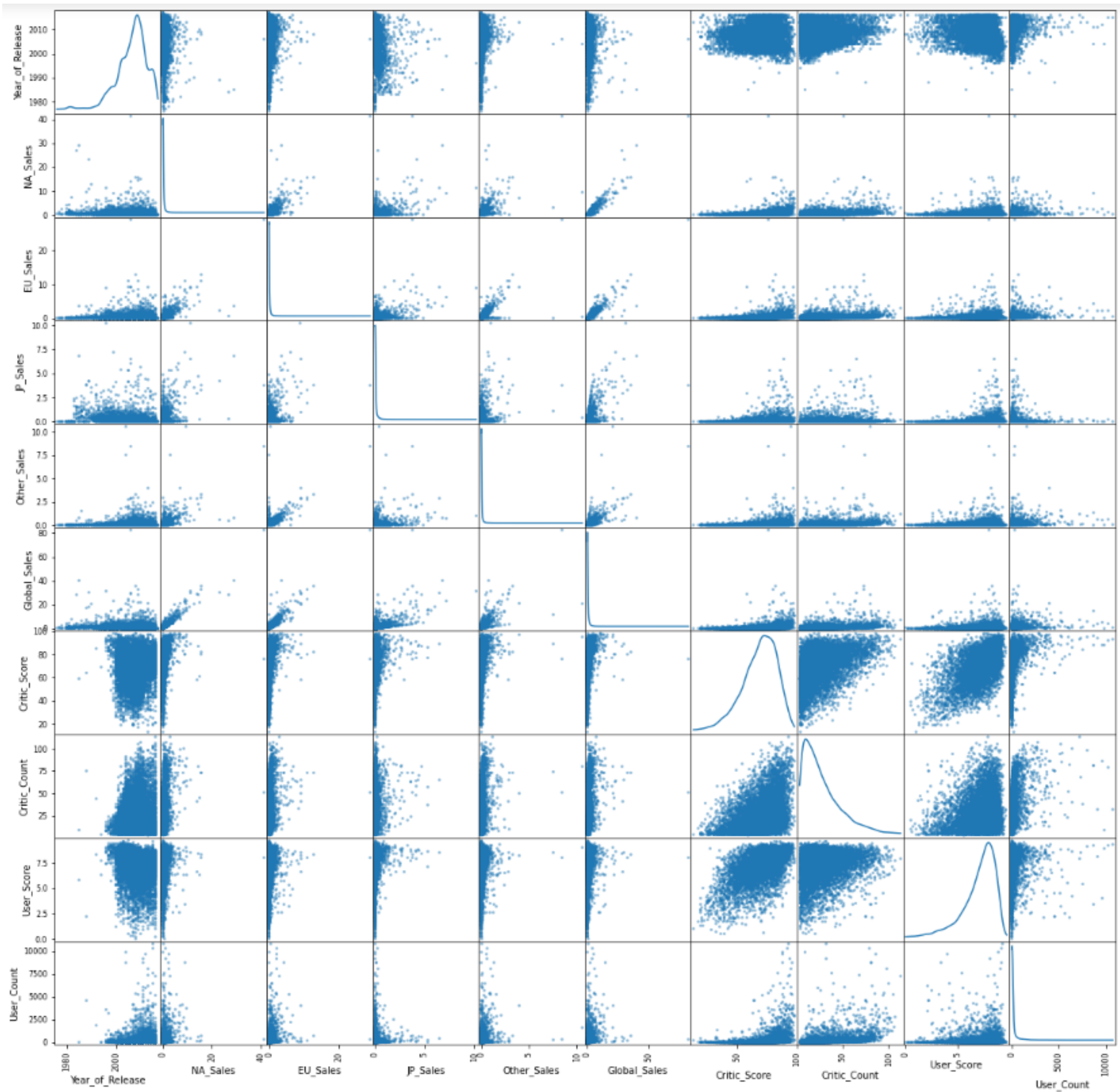
Half of the papers datasets were sales by region, such as sales in Japan and sales in Europe. The original paper used NA Sales as a target feature, so I went with that. However, using sales (Japan sales, EU sales and Global sales) to predict NA sales doesn't seem feasible for a future predictions, if I ever chose to use this model again. With that in mind, I did a correlation matrix to check out the features, and it turns out all of the sales are correlated with one another, therefore they are redundant. I removed the other 'Sales' features.

Since that reduced me down to only a few non Target features, I decided to use an altered version of the dataset. In my code I referred to it as df_vgsales2017. It included other useful features- User Score, User Count, Critic Score, Critic Count and Rating (E for Everyone, T for Teen, etc.).

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Rating
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.54	76.0	51.0	8.0	324.0	E
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	NaN	NaN	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.80	3.79	3.29	35.57	82.0	73.0	8.3	712.0	E
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.95	3.28	2.95	32.78	80.0	73.0	8.0	193.0	E
4	Pokemon Red/Pokemon Blue	G	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17411	Nancy Drew: The Deadly Secret of Olde World Park	DS	2007.0	Adventure	Majesco Entertainment	0.00	0.00	0.00	0.00	0.01	64.0	7.0	NaN	NaN	E
17412	Fashion Designer: Style Icon	DS	2007.0	Simulation	505 Games	0.00	0.00	0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN
17413	Ashita no Joe 2: The Anime Super Remix	PS2	2002.0	Fighting	Capcom	0.00	0.00	0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN
17414	NadePro!! Kisama no Seiyuu Yatte Miro!	PS2	2009.0	Adventure	GungHo	0.00	0.00	0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN
17415	Brian Lara 2007 Pressure Play	PSP	2007.0	Sports	Codemasters	0.00	0.00	0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN

17416 rows × 15 columns

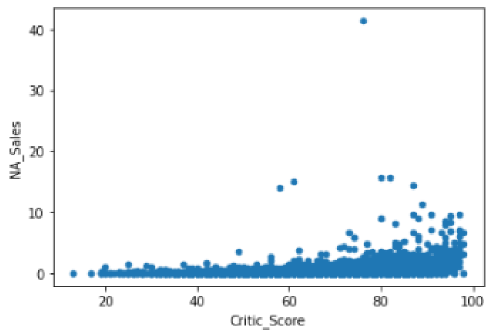
Scatter Matrix of data:



Some useful individual Scatter Plots show the relationship between critic score vs sales, user score vs sales, and platform vs sales:

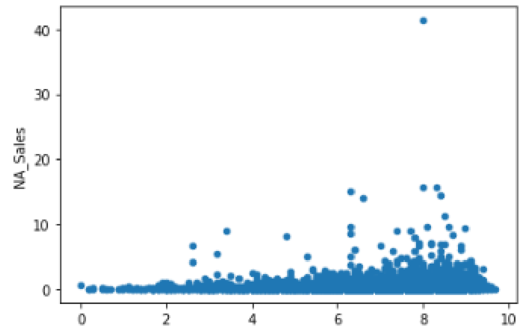
```python
# scatter matrix of vgsales 2017
df_vgsales_2017.plot.scatter('Critic_Score', 'NA_Sales')
```

```
<AxesSubplot:xlabel='Critic_Score', ylabel='NA_Sales'>
```



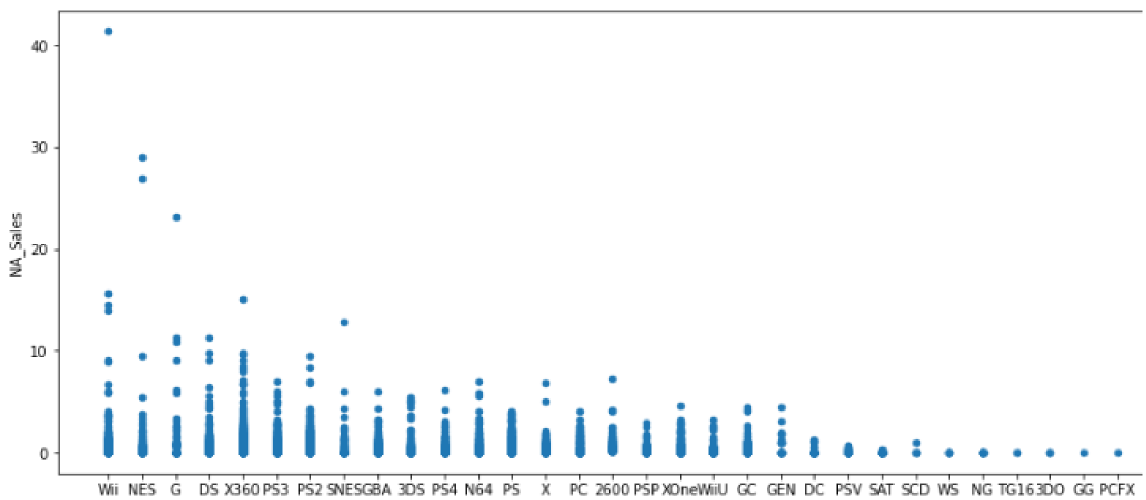

```python
# scatter matrix of vgsales 2017
df_vgsales_2017.plot.scatter('User_Score', 'NA_Sales')
```

```
<AxesSubplot:xlabel='User_Score', ylabel='NA_Sales'>
```




```python
# scatter matrix of vgsales 2017
df_vgsales_2017.plot.scatter('Platform', 'NA_Sales',figsize=(14,6))
```

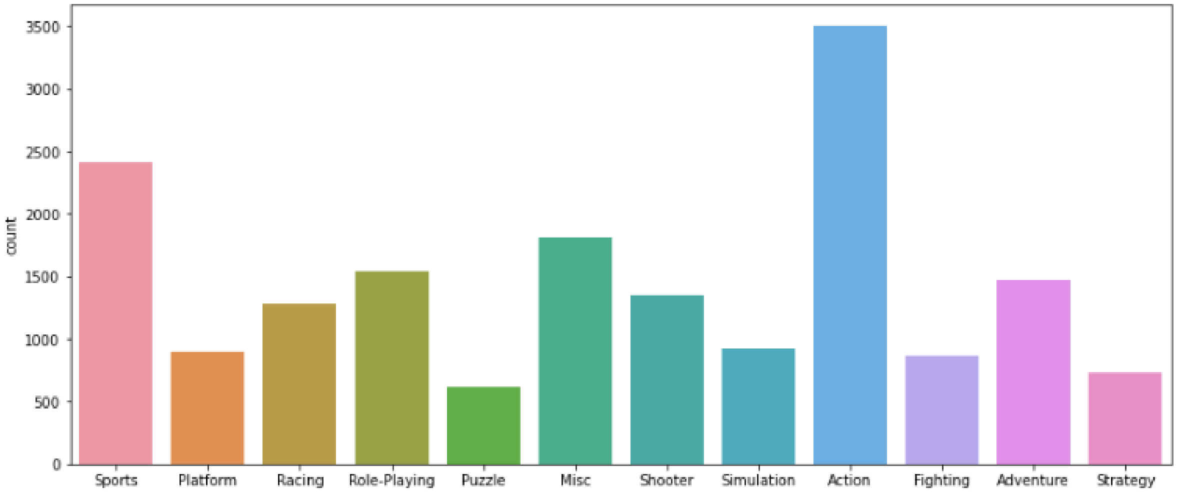
```
<AxesSubplot:xlabel='Platform', ylabel='NA_Sales'>
```



For better understanding of the data, I plotted some visual representations.

Frequency of Genre:

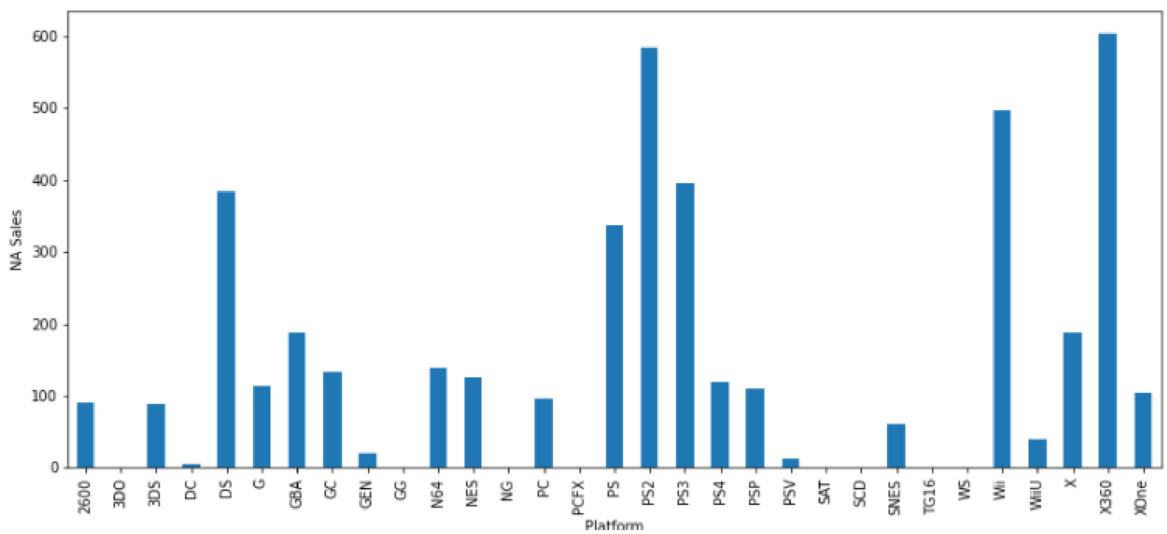
```python
# frequency of genre of vgsales 2017
plt.figure(figsize=(14,6))

ax = sns.countplot(x='Genre', data=df_vgsales_2017)
```



Sum of sales by Platform:

```python
# Platform vs total North America Sales of vgsales 2017
plt.figure(figsize=(14,6))
plt.ylabel('NA Sales')
#plt.bar(df_vgsales_2017['Platform'], df_vgsales_2017['NA_Sales'], width=0.8)
df_vgsales_2017.groupby('Platform').NA_Sales.sum().plot(kind='bar')
```

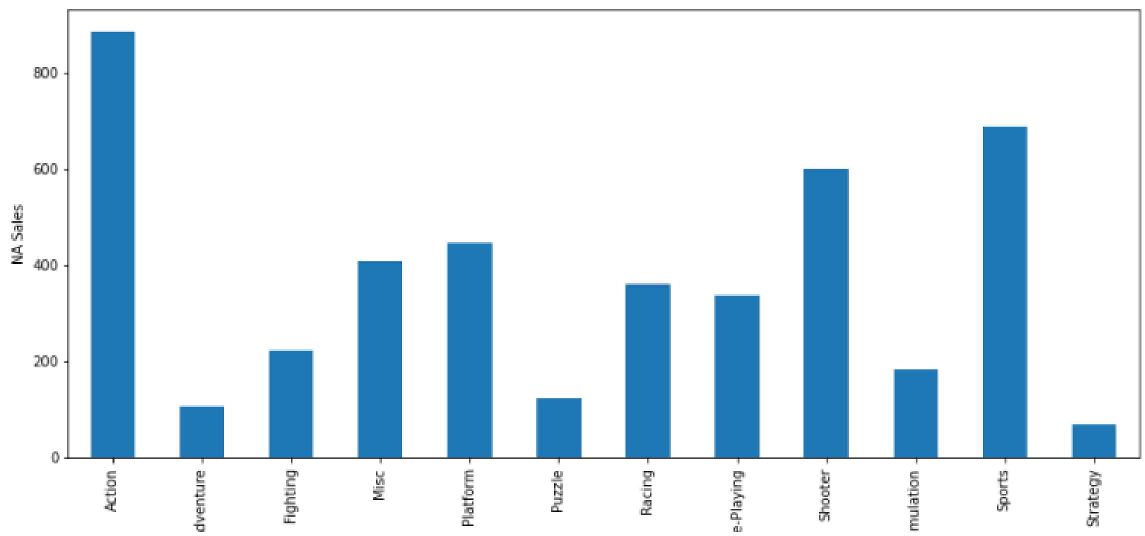
```
<AxesSubplot:xlabel='Platform', ylabel='NA Sales'>
```



Sum of sales by Genre:

```python
# # Genre vs total North America Sales of vgsales 2017
plt.figure(figsize=(14,6))
plt.ylabel('NA Sales')
df_vgsales_2017.groupby('Genre').NA_Sales.sum().plot(kind='bar')
```

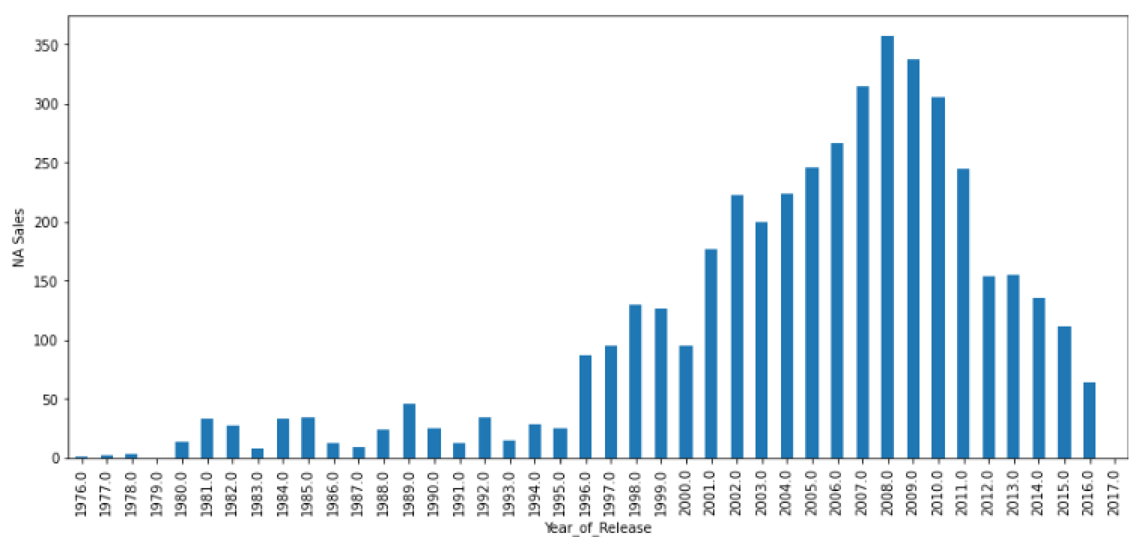
```
<AxesSubplot:xlabel='Genre', ylabel='NA Sales'>
```



Sum of sales by Year of Release:

```python
# scatter matrix of vgsales 2017
plt.figure(figsize=(14,6))
plt.ylabel('NA Sales')
df_vgsales_2017.groupby('Year_of_Release').NA_Sales.sum().plot(kind='bar')
```

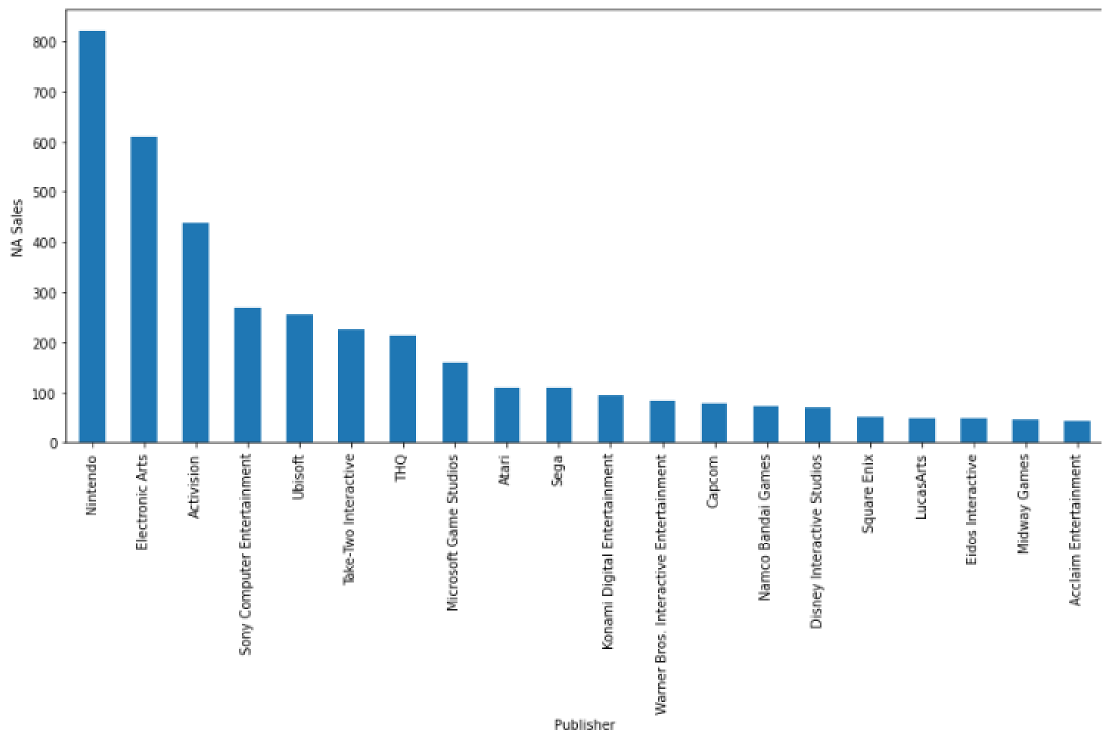
```
<AxesSubplot:xlabel='Year_of_Release', ylabel='NA Sales'>
```



Sum of Sales by Publisher:

```python
# Publisher vs total North America Sales of vgsales 2017 (top 20 sellers)
plt.figure(figsize=(14,6))
plt.ylabel('NA Sales')
df_vgsales_2017.groupby('Publisher').NA_Sales.sum().nlargest(n=20).plot(kind='bar')
```

```
<AxesSubplot:xlabel='Publisher', ylabel='NA Sales'>
```



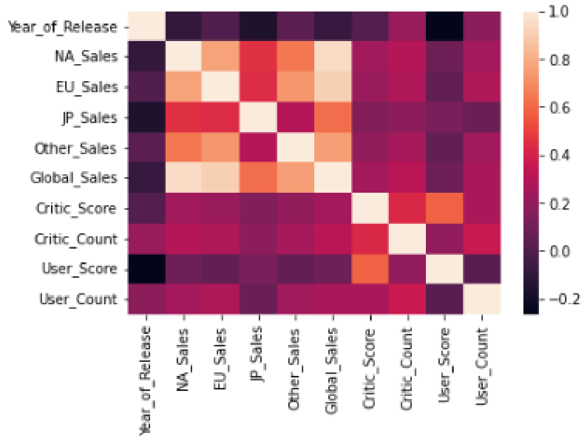
I went ahead and did a correlation matrix of the raw data without removing the other sales first, and displayed it via heat map:

```python
# Correlation matrix of vgsales 2017
correlation_matrix = df_vgsales_2017.corr()
correlation_matrix
```

	Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count
Year_of_Release	1.000000	-0.096994	0.000802	-0.166241	0.033849	-0.079628	0.010899	0.214983	-0.266821	0.174309
NA_Sales	-0.096994	1.000000	0.765520	0.451668	0.640798	0.941072	0.241139	0.299011	0.085027	0.247640
EU_Sales	0.000802	0.765520	1.000000	0.436560	0.725495	0.901681	0.220343	0.280763	0.054562	0.285077
JP_Sales	-0.166241	0.451668	0.436560	1.000000	0.293111	0.613325	0.152174	0.182793	0.125728	0.076720
Other_Sales	0.033849	0.640798	0.725495	0.293111	1.000000	0.751348	0.198686	0.255152	0.056474	0.241878
Global_Sales	-0.079628	0.941072	0.901681	0.613325	0.751348	1.000000	0.245523	0.307431	0.087241	0.266895
Critic_Score	0.010899	0.241139	0.220343	0.152174	0.198686	0.245523	1.000000	0.424108	0.582705	0.264277
Critic_Count	0.214983	0.299011	0.280763	0.182793	0.255152	0.307431	0.424108	1.000000	0.193619	0.361092
User_Score	-0.266821	0.085027	0.054562	0.125728	0.056474	0.087241	0.582705	0.193619	1.000000	0.028059
User_Count	0.174309	0.247640	0.285077	0.076720	0.241878	0.266895	0.264277	0.361092	0.028059	1.000000

```python
# Correlation matrix of vgsales 2017 - visual Heatmap form
sns.heatmap(correlation_matrix, xticklabels=correlation_matrix.columns, yticklabels=correlation_matrix.columns)
```

```
<AxesSubplot:>
```



The EU_Sales, NA_Sales, JP_Sales, Other_Sales and Global_Sales are all highly correlated.

```python
# Correlation above a certain threshold
threshold = 0.6
pd.DataFrame(np.abs(correlation_matrix.values) > threshold)
```

	0	1	2	3	4	5	6	7	8	9
0	True	False	False	False	False	False	False	False	False	False
1	False	True	True	False	True	True	False	False	False	False
2	False	True	True	False	True	True	False	False	False	False
3	False	False	False	True	False	True	False	False	False	False
4	False	True	True	False	True	True	False	False	False	False
5	False	True	True	True	True	True	False	False	False	False
6	False	False	False	False	False	False	True	False	False	False
7	False	False	False	False	False	False	False	True	False	False
8	False	False	False	False	False	False	False	False	True	False
9	False	False	False	False	False	False	False	False	False	True

As you can see, the problems are mostly just in the sales. So let's remove them down to just 'NA_Sales'

But I realized some data was missing – they needed to be numerical values.

I used the pandas.factorize method to achieve this

```
numerical, categoriesPlat = pd.factorize(df_vgsales2017_edit["Platform"])
df_vgsales2017_edit["Platform"] = numerical
#abs(df_vgsales2017_edit["Critic_Score"].corr(df_vgsales2017_edit["Platform"]))

numerical, categories = pd.factorize(df_vgsales2017_edit["Genre"])
df_vgsales2017_edit["Genre"] = numerical

numerical, categories = pd.factorize(df_vgsales2017_edit["Publisher"])
df_vgsales2017_edit["Publisher"] = numerical

numerical, categories = pd.factorize(df_vgsales2017_edit["Rating"])
df_vgsales2017_edit["Rating"] = numerical


display(df_vgsales2017_edit)
```

	Platform	Year_of_Release	Genre	Publisher	NA_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Rating
0	0	2006.0	0	0	41.36	76.0	51.0	8.0	324.0	0
1	1	1985.0	1	0	29.08	NaN	NaN	NaN	NaN	-1
2	0	2008.0	2	0	15.68	82.0	73.0	8.3	712.0	0
3	0	2009.0	0	0	15.61	80.0	73.0	8.0	193.0	0
4	2	1996.0	3	0	11.27	NaN	NaN	NaN	NaN	-1
...	...	...	...	...	...	...	...	...	...	...
17411	3	2007.0	10	34	0.00	64.0	7.0	NaN	NaN	0
17412	3	2007.0	7	11	0.00	NaN	NaN	NaN	NaN	-1
17413	6	2002.0	9	12	0.00	NaN	NaN	NaN	NaN	-1
17414	6	2009.0	10	64	0.00	NaN	NaN	NaN	NaN	-1
17415	16	2007.0	0	33	0.00	NaN	NaN	NaN	NaN	-1

17416 rows × 10 columns

I also removed missing values-

```
# Check NA values
sales_withNA = df_vgsales2017_edit.columns[df_vgsales2017_edit.isna().any()].tolist()
sales_withNA
```

```
['Year_of_Release', 'Critic_Score', 'Critic_Count', 'User_Score', 'User_Count']
```

```
# Check NA values
df_vgsales2017_edit = df_vgsales2017_edit.dropna()

df_vgsales2017_edit

```

	Platform	Year_of_Release	Genre	Publisher	NA_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Rating
0	0	2006.0	0	0	41.36	76.0	51.0	8.0	324.0	0
2	0	2008.0	2	0	15.68	82.0	73.0	8.3	712.0	0
3	0	2009.0	0	0	15.61	80.0	73.0	8.0	193.0	0
6	3	2006.0	1	0	11.28	89.0	65.0	8.5	433.0	0
7	0	2006.0	5	0	13.96	58.0	41.0	6.6	129.0	0
...	...	...	...	...	...	...	...	...	...	...
17394	14	2003.0	8	5	0.00	91.0	20.0	8.5	291.0	2
17401	14	2007.0	6	38	0.00	60.0	20.0	4.9	42.0	2
17402	14	2009.0	0	8	0.00	68.0	8.0	6.5	19.0	0
17404	14	2006.0	11	2	0.00	67.0	46.0	6.9	32.0	3
17407	10	2016.0	1	626	0.00	85.0	7.0	7.0	114.0	2

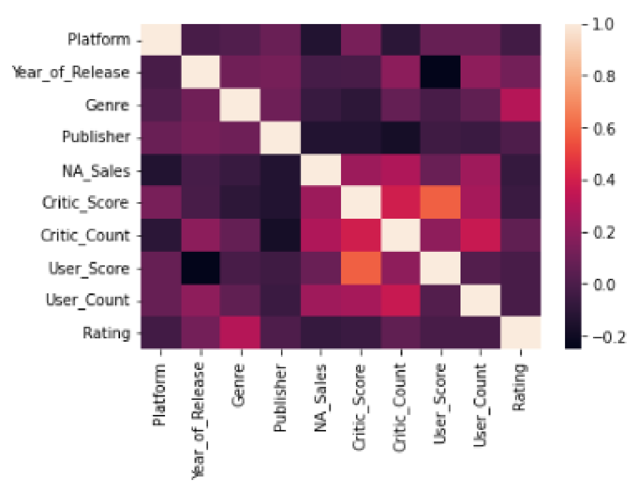
7191 rows × 10 columns

In retrospect, I likely didn't need to remove each row with ANY missing values, since Rating and User count were throwing a lot of missing segments. However the dataset is still quite large, at 7191, so I figured the models will still have enough to work with.

Correlation heat map after processing and transforming:

```
# Correlation matrix of vgsales 2017 - visual Heatmap form
correlation_matrix_edit = df_vgsales2017_edit.corr()
sns.heatmap(correlation_matrix_edit, xticklabels=correlation_matrix_edit.columns,
            yticklabels=correlation_matrix_edit.columns)
```

```
<AxesSubplot:>
```



So now it was time to split into test data and train data and fit/train the model:

```
# train and test split
from sklearn.model_selection import train_test_split

VG_X= df_vgsales2017_edit.drop('NA_Sales', axis=1)
VG_T= df_vgsales2017_edit['NA_Sales']

X_train, X_test, t_train, t_test = train_test_split(VG_X, VG_T, test_size=0.2)

print("Train data shape: {}".format(X_train.shape))
print("Train target shape: {}".format(t_train.shape))
print("Test data shape: {}".format(X_test.shape))
print("Test target shape: {}".format(t_test.shape))
```

```
Train data shape: (5752, 9)
Train target shape: (5752,)
Test data shape: (1439, 9)
Test target shape: (1439,)
```

```
# Create logistic regression, train the model and evaluate
from sklearn.linear_model import LinearRegression

model = LinearRegression()


model.fit(X_train, t_train)

train_score = model.score(X_train, t_train)
test_score = model.score(X_test, t_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score, test_score))

```

```
Train Accuracy: 0.19434746378954904, Test Accuracy: 0.09302078849256623
```

```
print('Coefficients:')
model.coef_

```

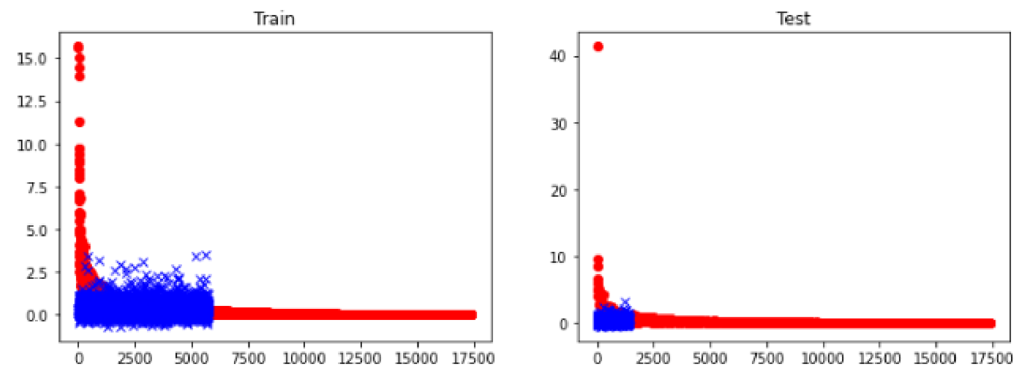
```
Coefficients:
```

```
array([-0.02282289, -0.01904455, -0.00858964, -0.00055469,  0.01105062,
        0.00876832, -0.0408606 ,  0.00025846, -0.04451696])
```

Plot train vs test predictions:

```python
y_train = model.predict(X_train)
y_test = model.predict(X_test)


plt.figure(figsize=(12,4))
plt.subplot(121)

# TODO 6: Plot the train labels using the 'ro' marker, y_train using the 'bx' marker.
plt.title('Train')
plt.plot(t_train, 'ro')
plt.plot(y_train, 'bx')


plt.subplot(122)

# TODO 7: Plot the test labels using the 'ro' marker, y_train using the 'bx' marker.
plt.title('Test')
plt.plot(t_test, 'ro')
plt.plot(y_test, 'bx')
```

```
[<matplotlib.lines.Line2D at 0x1ee36190c40>]
```



Test other algorithms:

```python
# Create regression models, train the models and evaluate
from sklearn.linear_model import Lasso, Ridge, SGDRegressor

model = SGDRegressor(alpha=0.5)

model.fit(X_train, t_train)

train_score = model.score(X_train, t_train)
test_score = model.score(X_test, t_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score, test_score))

```

```
Train Accuracy: -3.759572823987611e+30, Test Accuracy: -1.459424682260178e+30
```

```python
model = Lasso(alpha=0.1)

model.fit(X_train, t_train)

train_score = model.score(X_train, t_train)
test_score = model.score(X_test, t_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score, test_score))

```

```
Train Accuracy: 0.18588671743772067, Test Accuracy: 0.08742783025922907
```

```
model = Ridge(alpha=2.5)

model.fit(X_train, t_train)

train_score = model.score(X_train, t_train)
test_score = model.score(X_test, t_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score, test_score))

```

```
Train Accuracy: 0.19434746291118132, Test Accuracy: 0.09302068332904478
```

Conclusions-

Both the platform (console) the game is released on and the critic score were helpful indicators of how well the model could predict. The accuracy of the predictions of my models were worse than in the original paper. The differences come from me including other features and removing the sales features. However, in my opinion, using sales to predict sales is not a very useful or realistic way to use machine learning. I worked on selecting features and removing NA values, but I still feel as though the features in the dataset could use more tweaking. If I was more thoughtful with what to remove (missing values) and what not to remove, I think I could have gotten better predictions. In the future, I would like a more robust dataset, I'd like to make use of different models, and apply neural networks to the predictions, if the data allows it. Another thing I would like to try is to apply feature selection algorithms before running it through regression models. Another extra step could be- parameter testing/selection with Ridge, Lasso and SGD Regression. The paper actually didn't include code examples so I had to do the base work by scratch. However, since it was just Linear Regression, it wasn't difficult to compute and expand on it using SciKit learn and so on.

Please do share my report and presentation if you see fit.

Works Cited

MURILÃO. (2020, July 21). *EDA - VIDEO GAME SALES* . Retrieved from kaggle: https://www.kaggle.com/upadorprofzs/eda-video-game-sales

TM Geethanjali, R. D. (2020, May 5). *Video Games Sales Analysis: A Data Science Approach*. Retrieved from ijcrt.org: https://ijcrt.org/papers/IJCRT2005182.pdf

Trněný, M. (2017). *Machine Learning for Predicting Success of Video Games*. Retrieved from Masaryk University Faculty of Informatics: https://is.muni.cz/th/k2c5b/diploma_thesis_trneny.pdf

Github link: https://github.com/Jnavarra41/Jnavarra5156MachineLearning