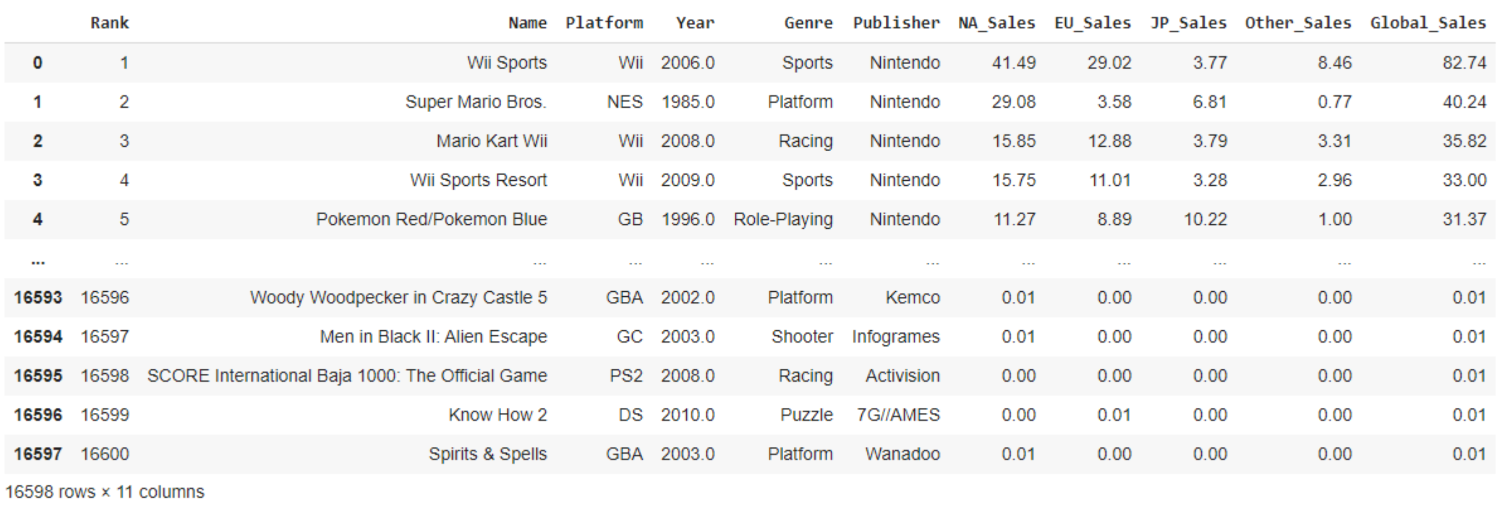
Video Game Sales

This dataset contains a list of video games with sales greater than 100,000 copies. It was generated by a scrape of vgchartz.com.

Fields include:

- Rank - Ranking of overall sales
- Name - The games name
- Platform - Platform of the games release (i.e. PC,PS4, etc.)
- Year - Year of the game's release
- Genre - Genre of the game
- Publisher - Publisher of the game
- NA_Sales - Sales in North America (in millions)
- EU_Sales - Sales in Europe (in millions)
- JP_Sales - Sales in Japan (in millions)
- Other_Sales - Sales in the rest of the world (in millions)
- Global_Sales - Total worldwide sales.

```
The dataset contains 16598 Rows and 11 Columns.
```



	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
...	...	...	...	...	...	...	...	...	...	...	...
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.00	0.00	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.00	0.00	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.00	0.00	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.00	0.00	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.00	0.01

16598 rows × 11 columns

Data preprocessing steps:

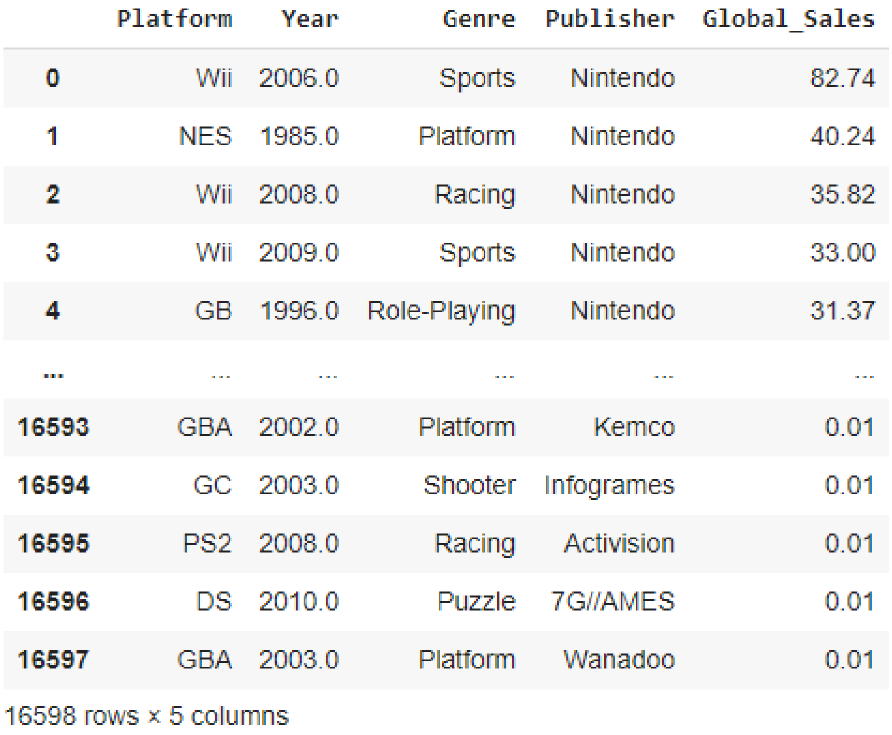
1- **Attribute Selection.**

After the process of collecting and confirming it, we delete the columns that do not have much importance in the data, and deleting them does not affect the rest of the columns.

Drop Irrelevant Attributes

```
dropped_list = ['Rank','Name', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales
']
```

```
Dataset after drop process contains 16598 Rows and 5 Columns.
```



	Platform	Year	Genre	Publisher	Global_Sales
0	Wii	2006.0	Sports	Nintendo	82.74
1	NES	1985.0	Platform	Nintendo	40.24
2	Wii	2008.0	Racing	Nintendo	35.82
3	Wii	2009.0	Sports	Nintendo	33.00
4	GB	1996.0	Role-Playing	Nintendo	31.37
...	...	...	...	...	...
16593	GBA	2002.0	Platform	Kemco	0.01
16594	GC	2003.0	Shooter	Infogrames	0.01
16595	PS2	2008.0	Racing	Activision	0.01
16596	DS	2010.0	Puzzle	7G//AMES	0.01
16597	GBA	2003.0	Platform	Wanadoo	0.01

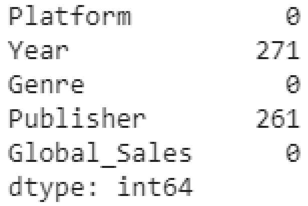
16598 rows × 5 columns

2- Data Cleaning:

We dropped all NON and Unknown data because the number of NON record is small based on the number of all records.

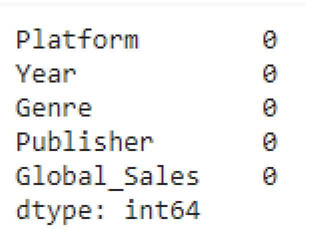
a) **Find null and Unknown record**

The total number of NON values in dataset are 532



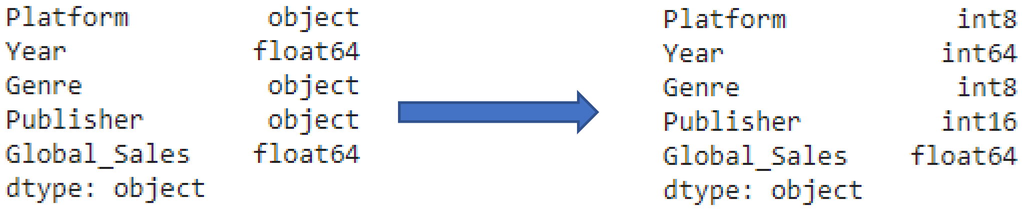
```
Platform          0
Year            271
Genre             0
Publisher       261
Global_Sales      0
dtype: int64
```

b) **Delete null and Unknown record**



```
Platform        0
Year            0
Genre           0
Publisher       0
Global_Sales    0
dtype: int64
```

3- Convert Categorial Attributes for int:

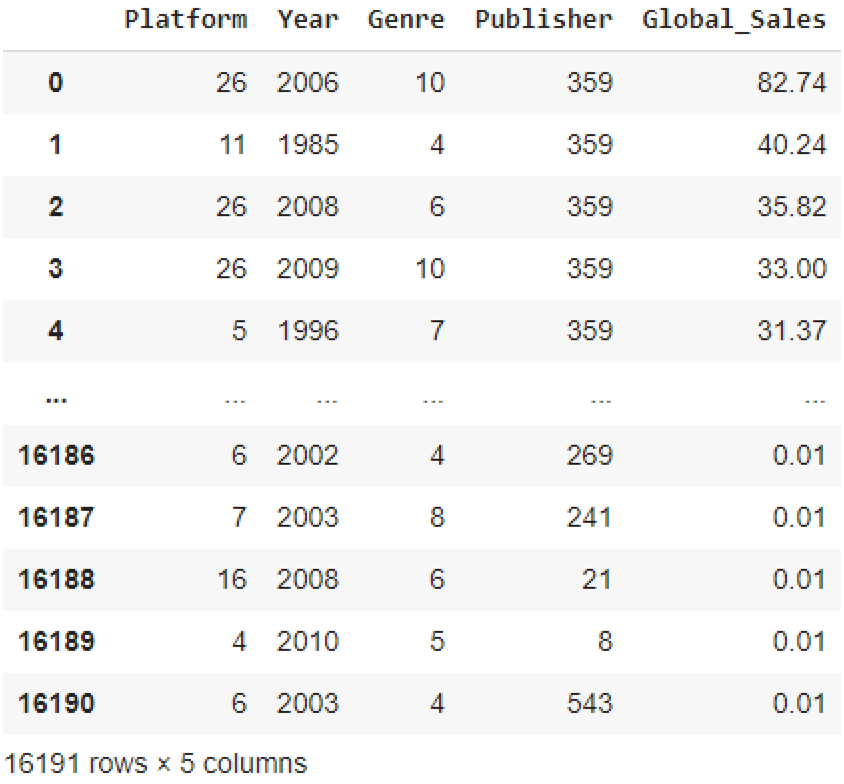


```
Platform         object
Year            float64
Genre            object
Publisher        object
Global_Sales    float64
dtype: object
```

```
Platform           int8
Year              int64
Genre              int8
Publisher         int16
Global_Sales    float64
dtype: object
```

4- **Reset index** (The dataset has index doesn't used). And show data
The Final results for the Dataset:

```
Dataset contains 16191 Rows and 5 Columns.
```

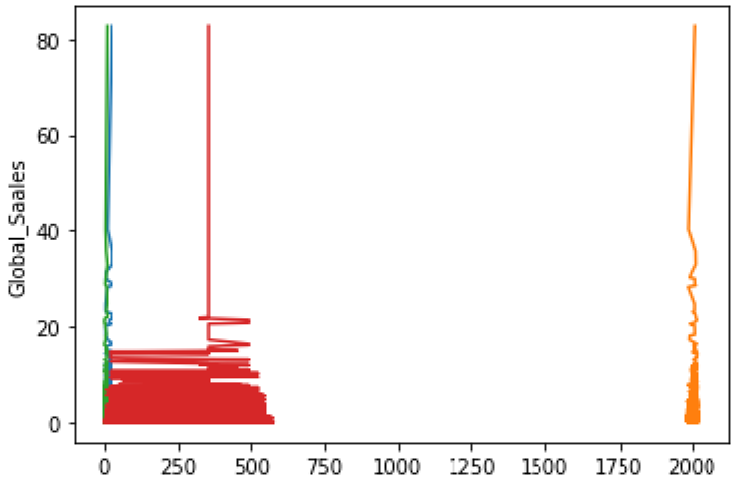


	Platform	Year	Genre	Publisher	Global_Sales
0	26	2006	10	359	82.74
1	11	1985	4	359	40.24
2	26	2008	6	359	35.82
3	26	2009	10	359	33.00
4	5	1996	7	359	31.37
...	...	...	...	...	...
16186	6	2002	4	269	0.01
16187	7	2003	8	241	0.01
16188	16	2008	6	21	0.01
16189	4	2010	5	8	0.01
16190	6	2003	4	543	0.01

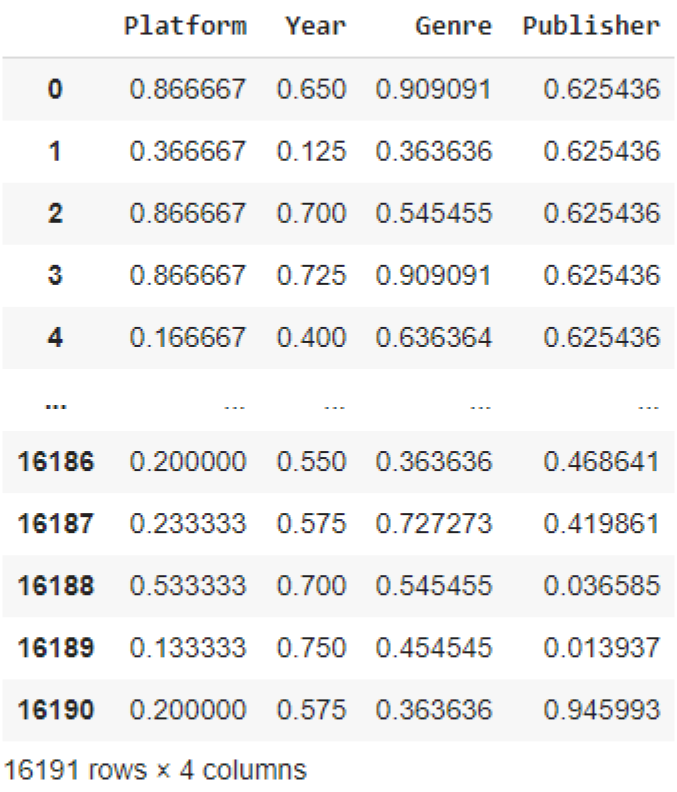
16191 rows × 5 columns

5- Data Visualization

5.1- **Data Visualization - Before Scaling**



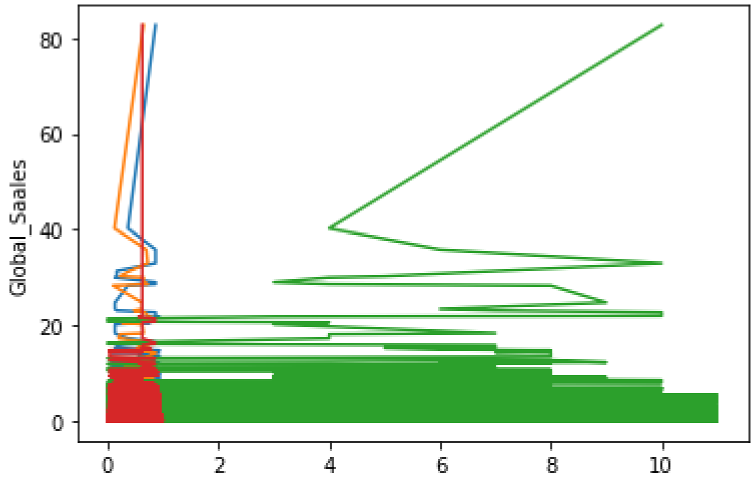
6- Scaling -with MinMaxScaler



	Platform	Year	Genre	Publisher
0	0.866667	0.650	0.909091	0.625436
1	0.366667	0.125	0.363636	0.625436
2	0.866667	0.700	0.545455	0.625436
3	0.866667	0.725	0.909091	0.625436
4	0.166667	0.400	0.636364	0.625436
...	...	...	...	...
16186	0.200000	0.550	0.363636	0.468641
16187	0.233333	0.575	0.727273	0.419861
16188	0.533333	0.700	0.545455	0.036585
16189	0.133333	0.750	0.454545	0.013937
16190	0.200000	0.575	0.363636	0.945993

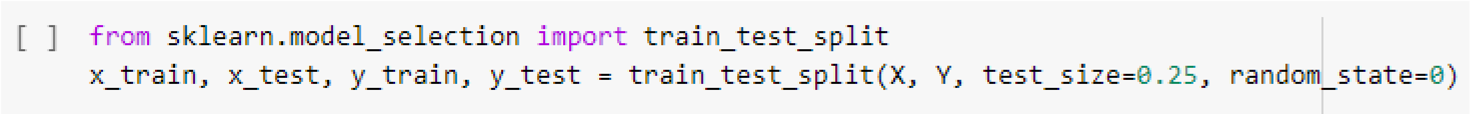
16191 rows × 4 columns

5.2- **Data Visualization - After Scaling**



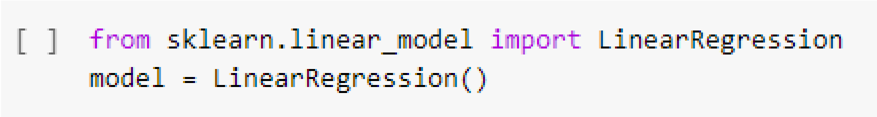
7- Modeling

a) Split into training and test set



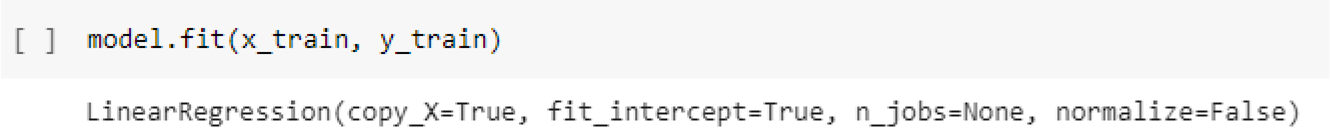
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=0)
```

b) Initiate Model



```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

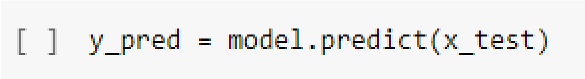
c) Train Model



```
model.fit(x_train, y_train)
```

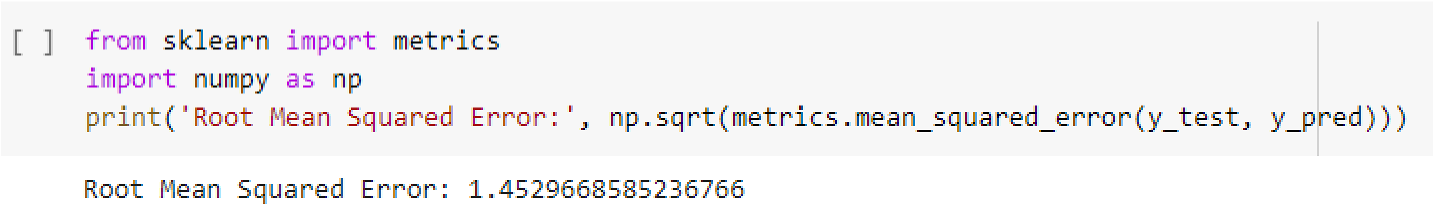
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

d) Test Model



```
y_pred = model.predict(x_test)
```

Evaluate Model



```
from sklearn import metrics
import numpy as np
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Root Mean Squared Error: 1.4529668585236766
```