Data 151 Project: Video Game Sales

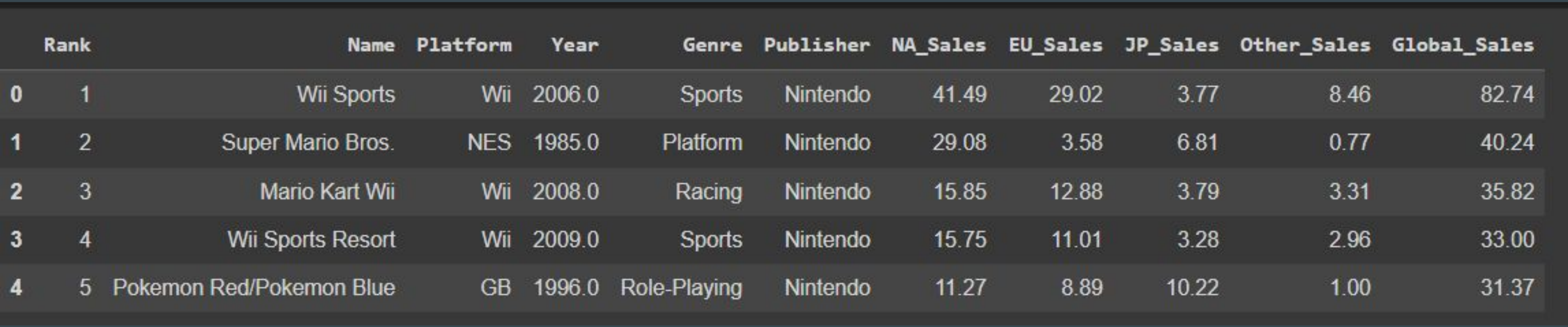
•••

Cheyann Odle, Ricky Gajeski

Original Intentions of the Project

- Originally had a different topic for the project
  - Data set was not useful
- Found new data set that piqued our interest, that being video game sales
- After discussing, decided that new data set had an applicable use for the real world

General View

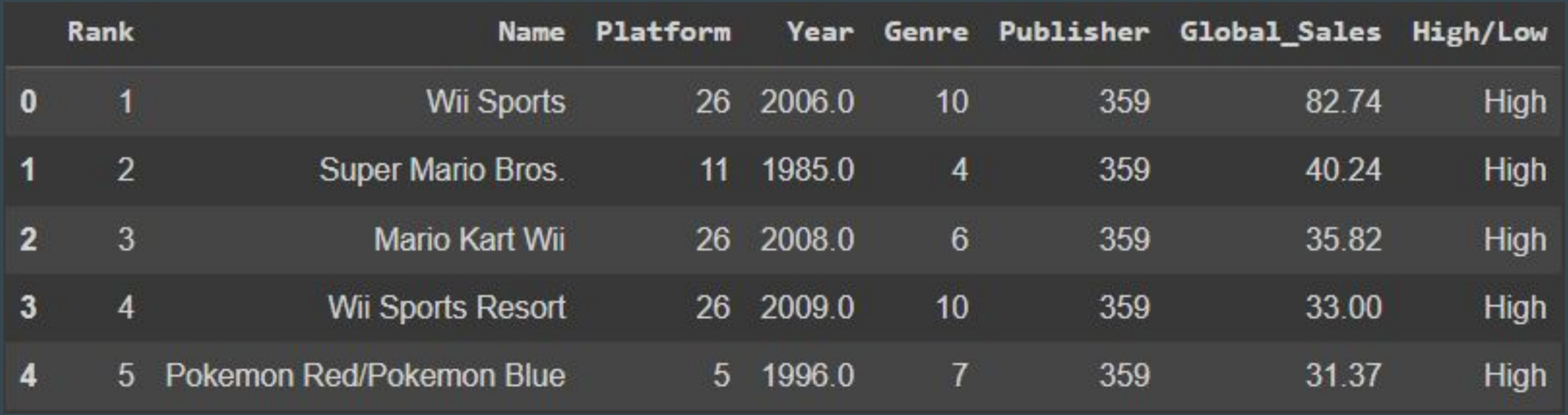


	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	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

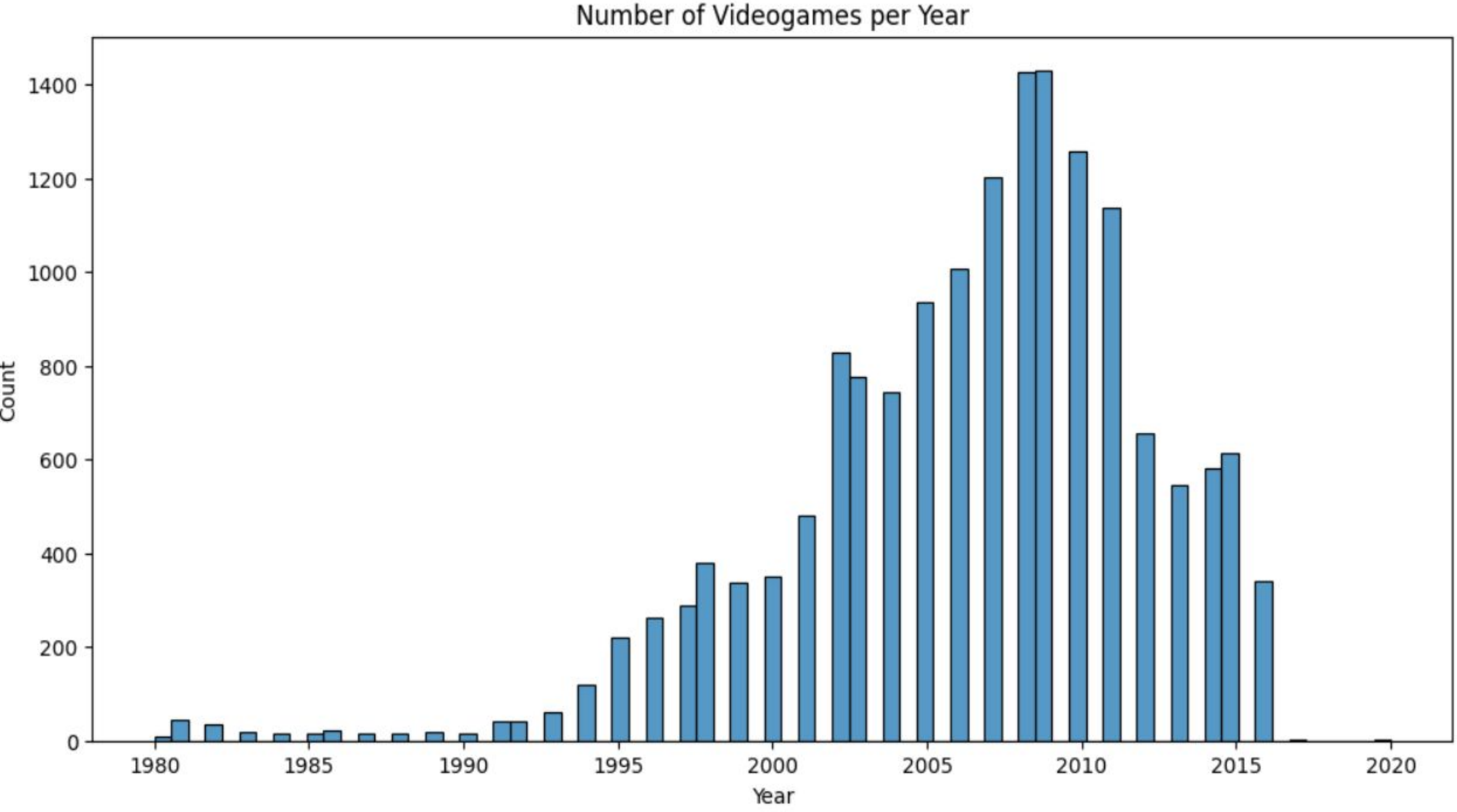
Cleaning Up/Altering the Code

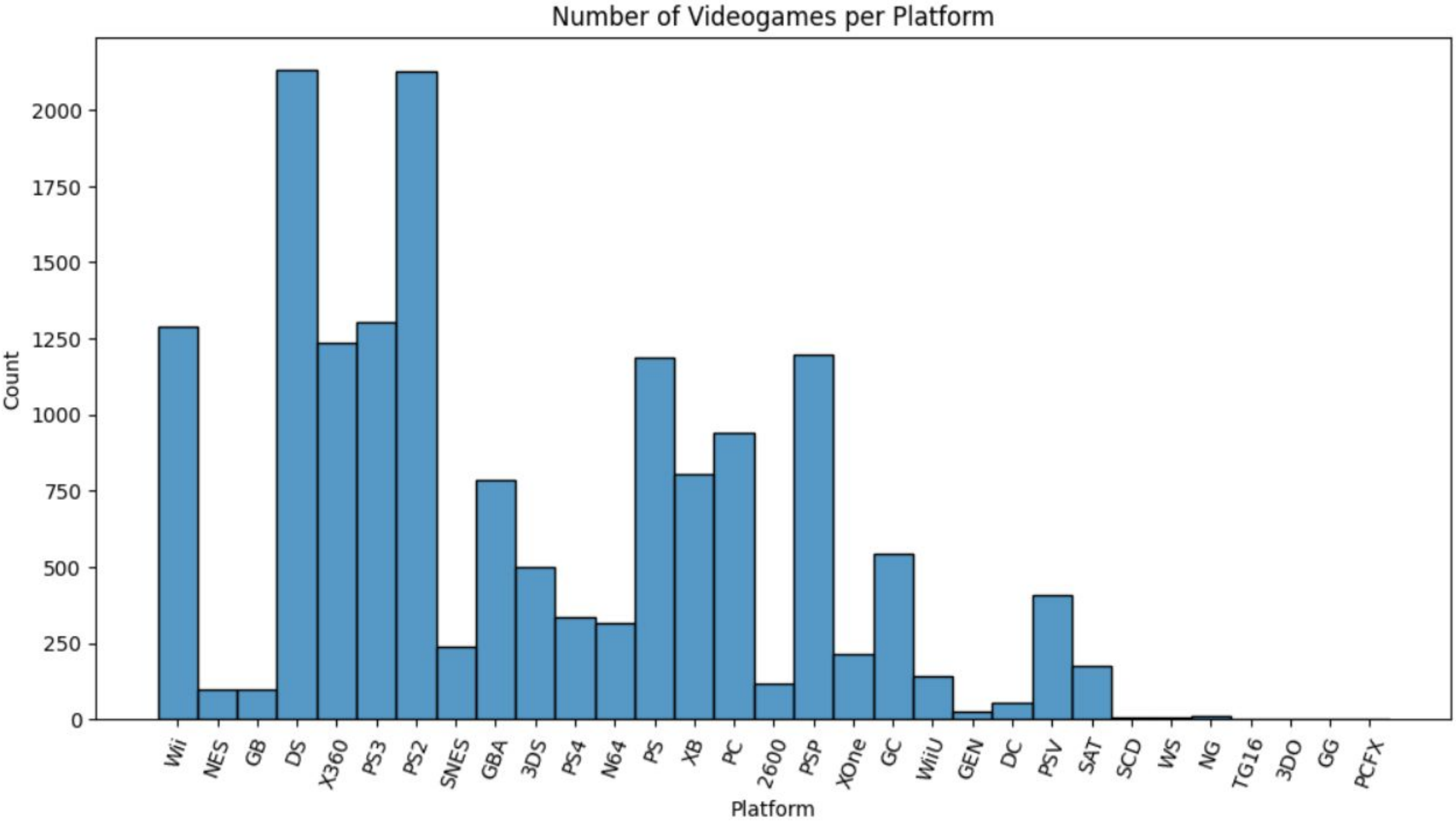
- Our dataset had a few issues
    - Had null values that needed to be removed
    - Had to drop the multiple sales column, leaving just global sales
    - Had to encode and add dummy columns
    - Had to add a High/Low column to better identify good selling video games

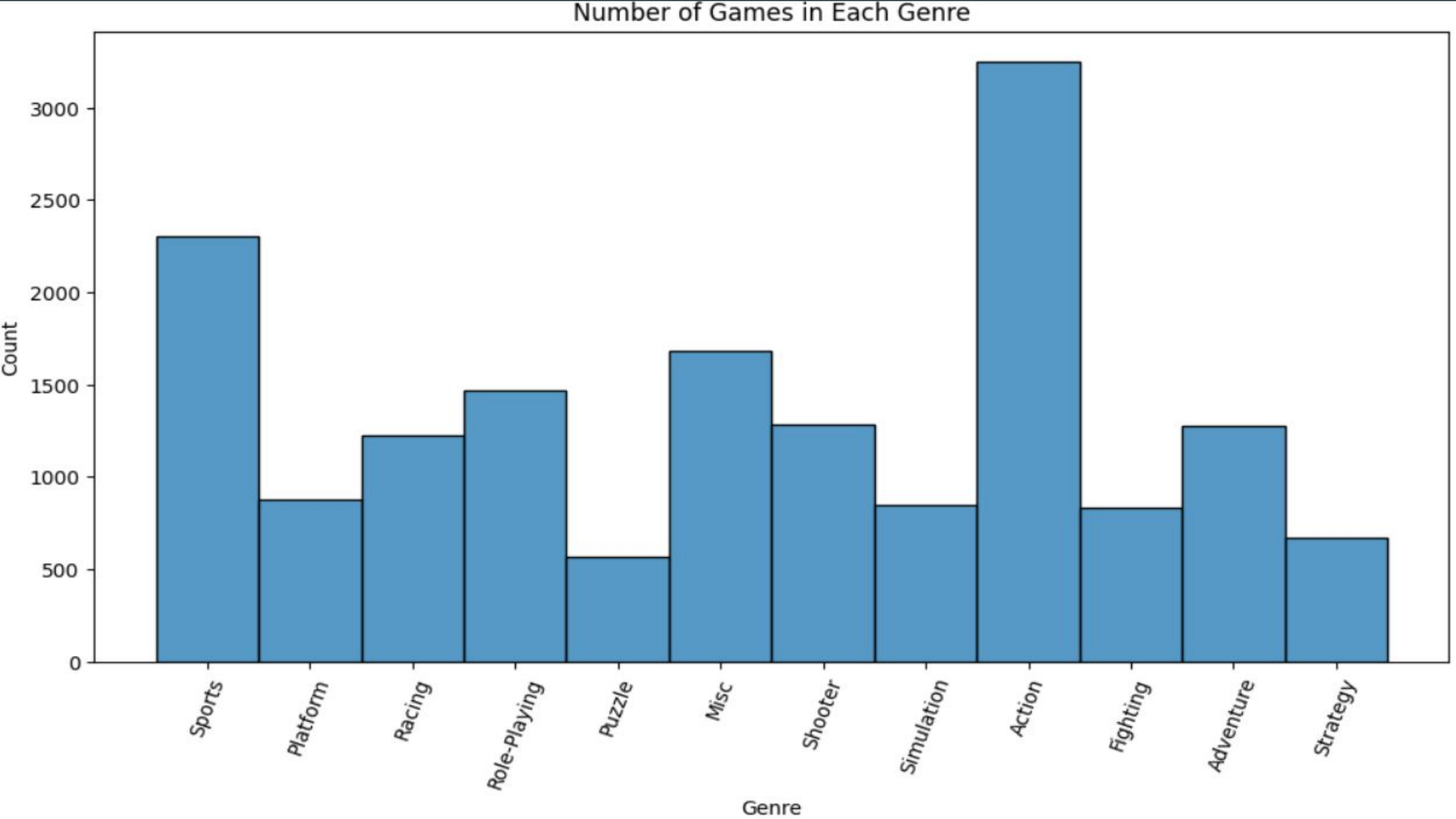
After Cleaning



	Rank	Name	Platform	Year	Genre	Publisher	Global_Sales	High/Low
0	1	Wii Sports	26	2006.0	10	359	82.74	High
1	2	Super Mario Bros.	11	1985.0	4	359	40.24	High
2	3	Mario Kart Wii	26	2008.0	6	359	35.82	High
3	4	Wii Sports Resort	26	2009.0	10	359	33.00	High
4	5	Pokemon Red/Pokemon Blue	5	1996.0	7	359	31.37	High


Number of Videogames per Year
Count
1400
1200
1000
800
600
400
200
0
1980
1985
1990
1995
2000
2005
2010
2015
2020
Year

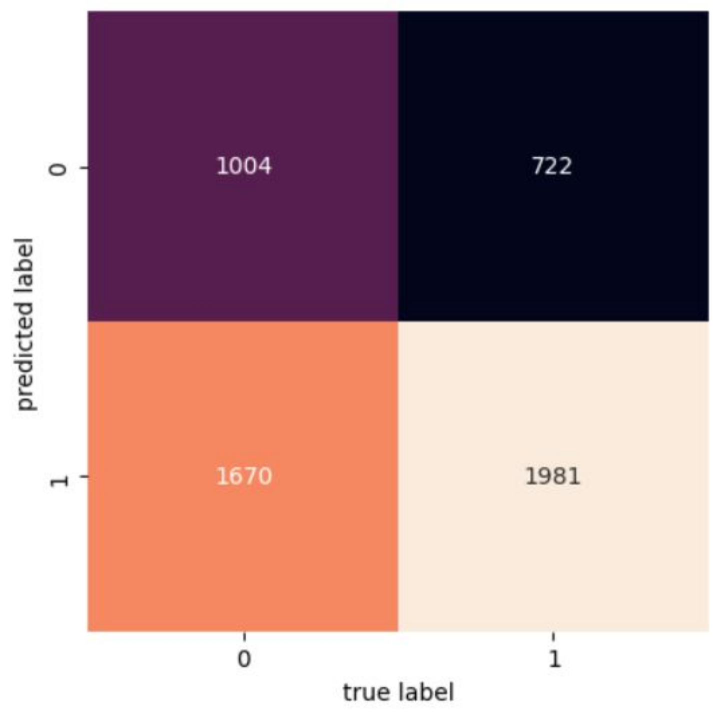

Number of Videogames per Platform
Count
2000
1750
1500
1250
1000
750
500
250
0
Platform
Wii
NES
GB
DS
X360
PS3
PS2
SNES
GBA
3DS
PS4
N64
PS
XB
PC
2600
PSP
XOne
GC
WiiU
GEN
DC
PSV
SAT
SCD
WS
NG
TG16
3DO
GG
PCFX

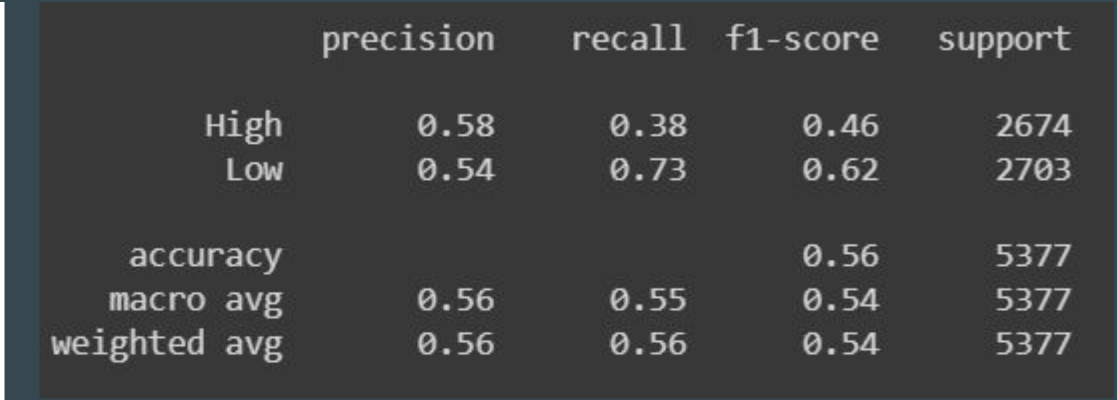

Number of Games in Each Genre
Count
3000
2500
2000
1500
1000
500
0
Sports
Platform
Racing
Role-Playing
Puzzle
Misc
Shooter
Simulation
Action
Fighting
Adventure
Strategy
Genre

Splitting the Data and Creating Models

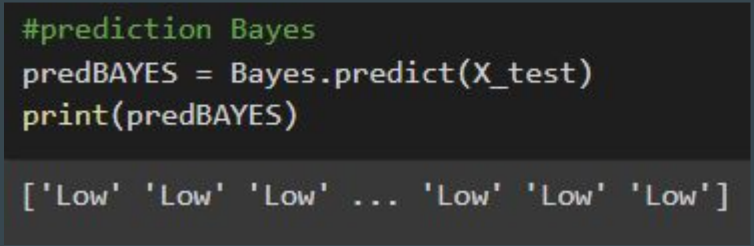
- We decided to split our data 33% test and 66% training
    - More data in training the better
- Created four Classification Models overall
    - Naive Bayes
    - Decision Tree
    - Random Forest
    - AdaBoost

Confusion Matrix and Scores for Bayes Model





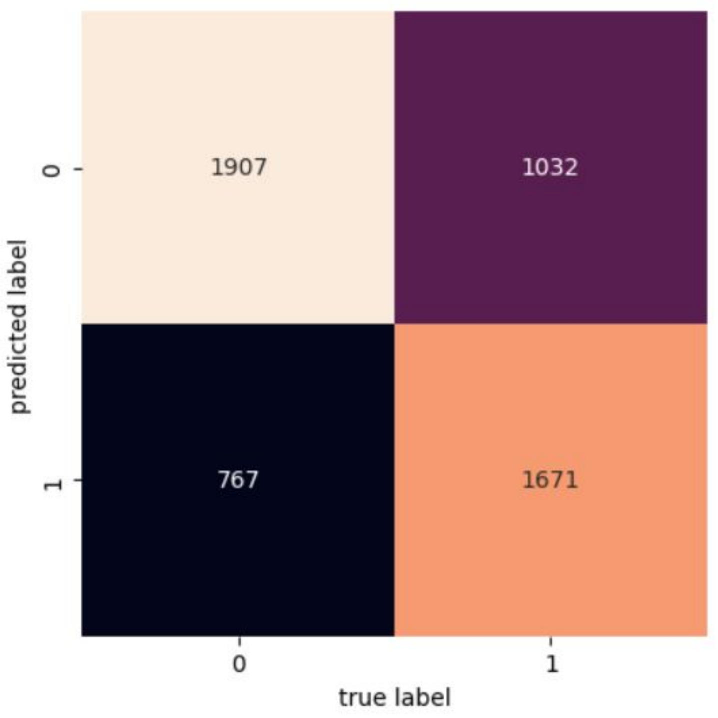
```
              precision    recall  f1-score   support

        High       0.58      0.38      0.46      2674
         Low       0.54      0.73      0.62      2703

    accuracy                           0.56      5377
   macro avg       0.56      0.55      0.54      5377
weighted avg       0.56      0.56      0.54      5377
```

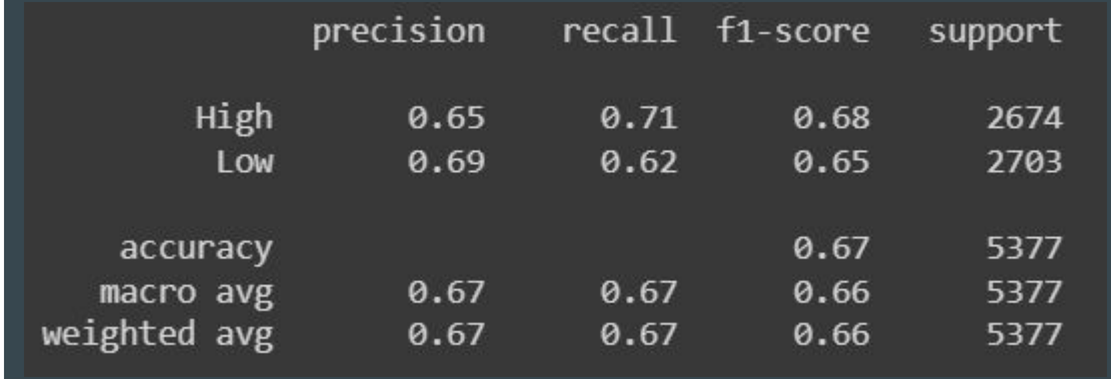


```
#prediction Bayes
predBAYES = Bayes.predict(X_test)
print(predBAYES)
```

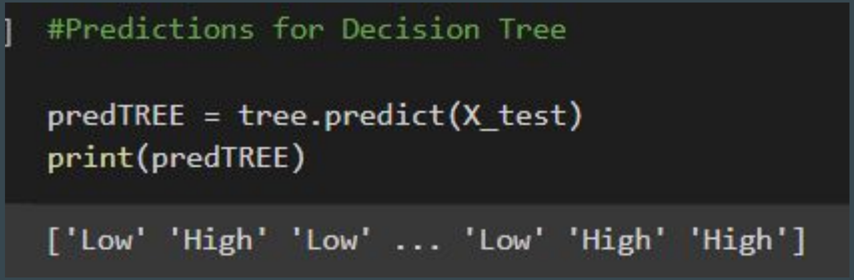
```
['Low' 'Low' 'Low' ... 'Low' 'Low' 'Low']
```

Confusion Matrix and Scores for Decision Tree Model



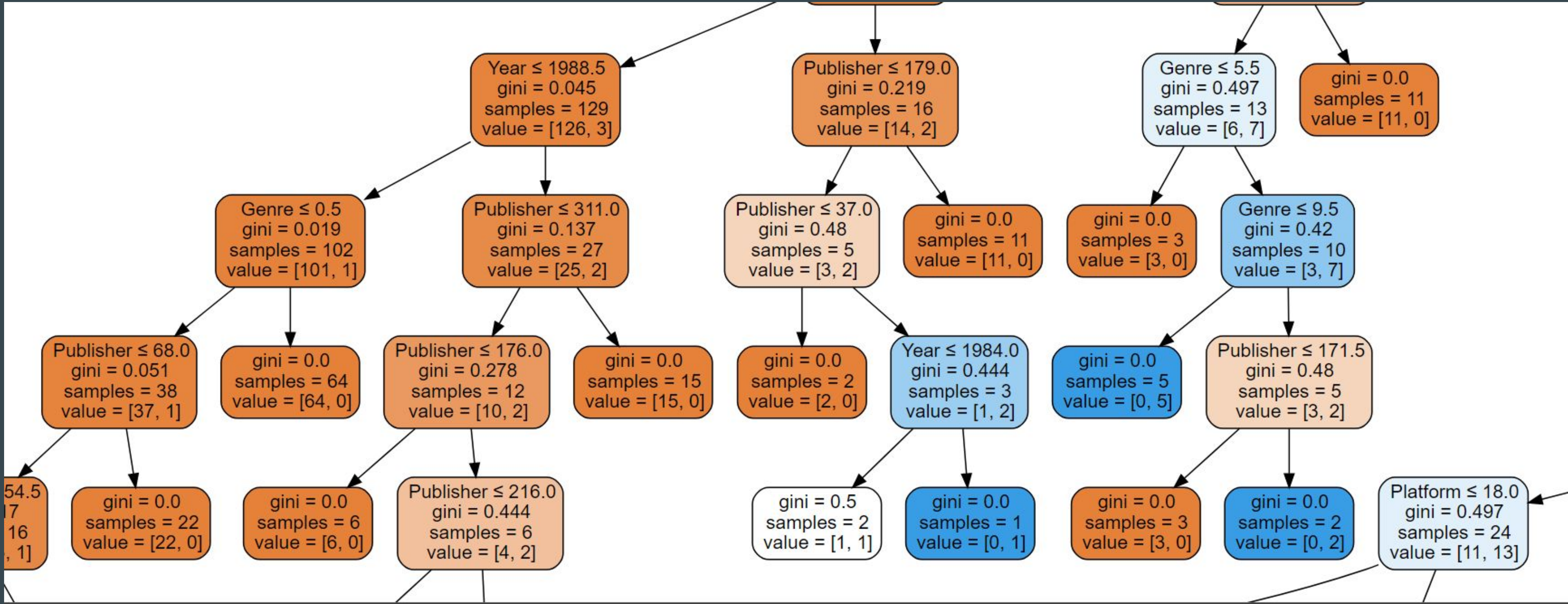


```
              precision    recall  f1-score   support

        High       0.65      0.71      0.68      2674
         Low       0.69      0.62      0.65      2703

    accuracy                           0.67      5377
   macro avg       0.67      0.67      0.66      5377
weighted avg       0.67      0.67      0.66      5377
```

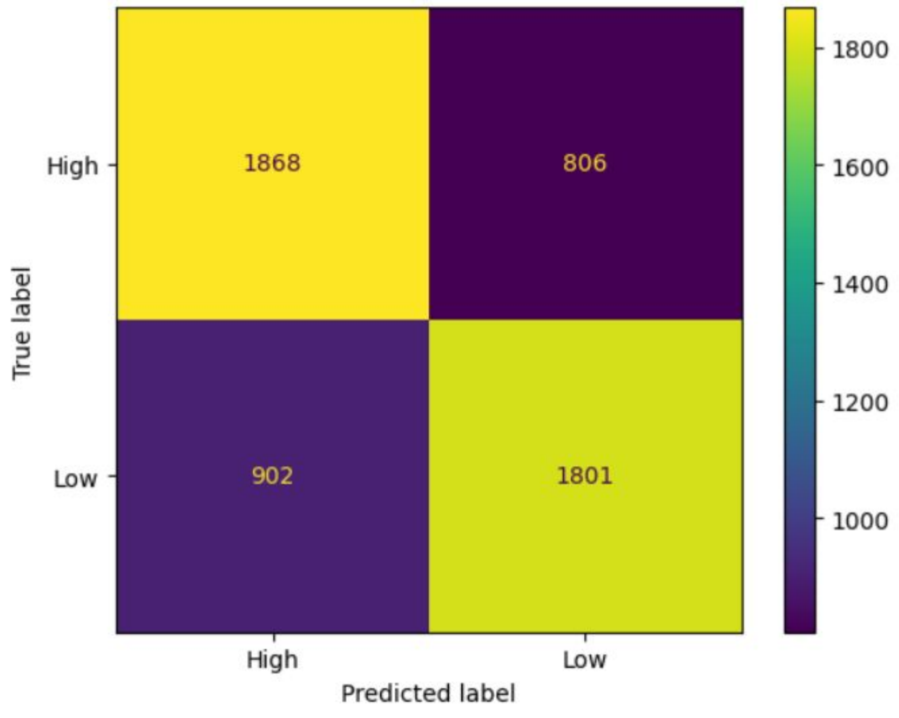


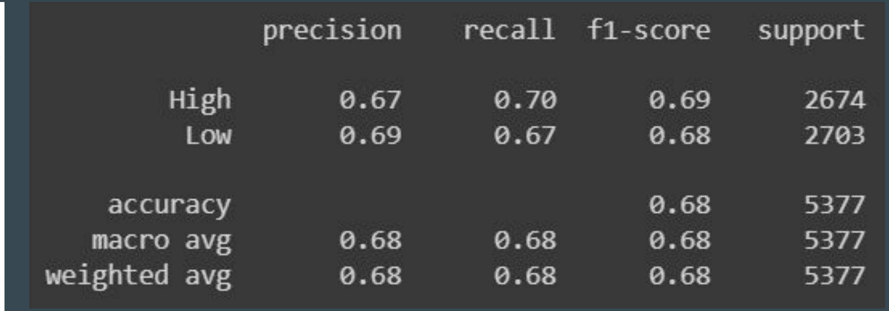
```
#Predictions for Decision Tree

predTREE = tree.predict(X_test)
print(predTREE)
```

```
['Low' 'High' 'Low' ... 'Low' 'High' 'High']
```

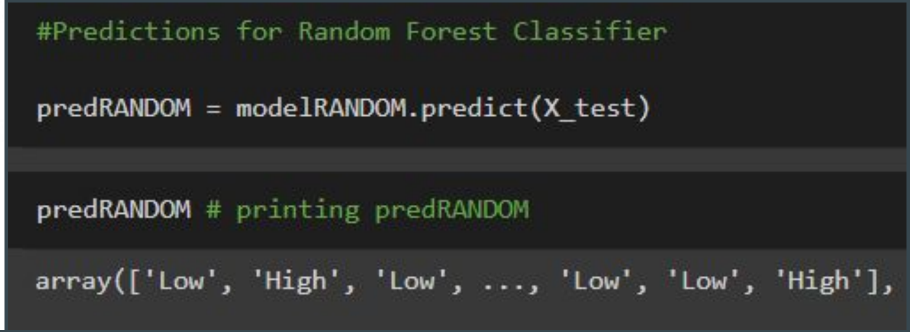

Year ≤ 1988.5 gini = 0.045 samples = 129 value = [126, 3]
Publisher ≤ 179.0 gini = 0.219 samples = 16 value = [14, 2]
Genre ≤ 5.5 gini = 0.497 samples = 13 value = [6, 7]
gini = 0.0 samples = 11 value = [11, 0]
Genre ≤ 0.5 gini = 0.019 samples = 102 value = [101, 1]
Publisher ≤ 311.0 gini = 0.137 samples = 27 value = [25, 2]
Publisher ≤ 37.0 gini = 0.48 samples = 5 value = [3, 2]
gini = 0.0 samples = 11 value = [11, 0]
gini = 0.0 samples = 3 value = [3, 0]
Genre ≤ 9.5 gini = 0.42 samples = 10 value = [3, 7]
Publisher ≤ 68.0 gini = 0.051 samples = 38 value = [37, 1]
gini = 0.0 samples = 64 value = [64, 0]
Publisher ≤ 176.0 gini = 0.278 samples = 12 value = [10, 2]
gini = 0.0 samples = 15 value = [15, 0]
gini = 0.0 samples = 2 value = [2, 0]
Year ≤ 1984.0 gini = 0.444 samples = 3 value = [1, 2]
gini = 0.0 samples = 5 value = [0, 5]
Publisher ≤ 171.5 gini = 0.48 samples = 5 value = [3, 2]
54.5 17 16 , 1]
gini = 0.0 samples = 22 value = [22, 0]
gini = 0.0 samples = 6 value = [6, 0]
Publisher ≤ 216.0 gini = 0.444 samples = 6 value = [4, 2]
gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.0 samples = 2 value = [0, 2]
Platform ≤ 18.0 gini = 0.497 samples = 24 value = [11, 13]

Confusion Matrix and Scores for Random Forest Model





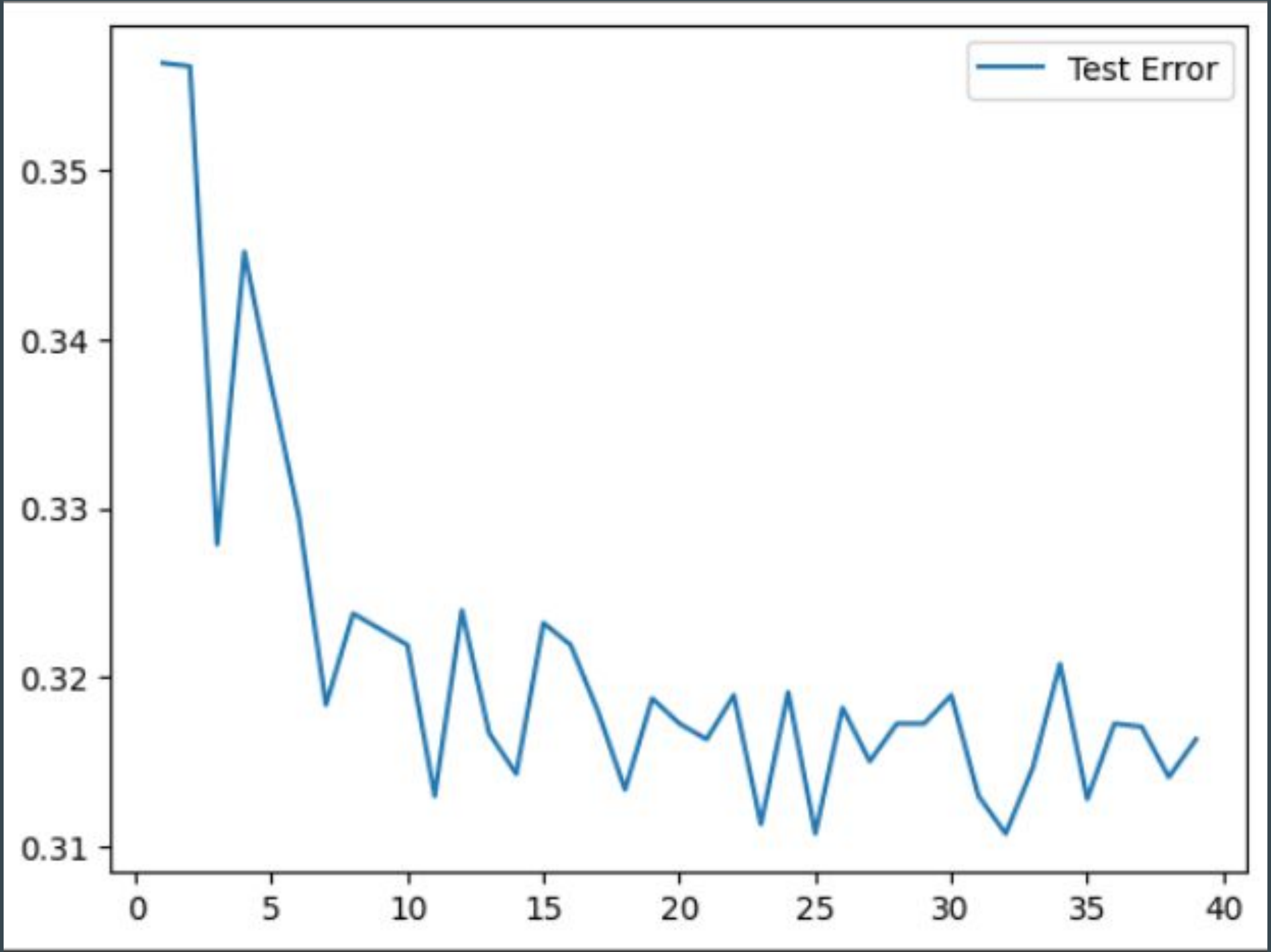
```
              precision    recall  f1-score   support

        High       0.67      0.70      0.69      2674
         Low       0.69      0.67      0.68      2703

    accuracy                           0.68      5377
   macro avg       0.68      0.68      0.68      5377
weighted avg       0.68      0.68      0.68      5377
```



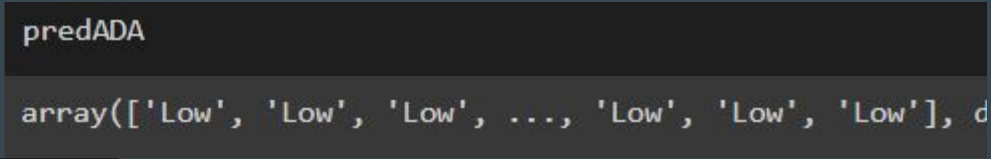
```
#Predictions for Random Forest Classifier

predRANDOM = modelRANDOM.predict(X_test)
```

```
predRANDOM # printing predRANDOM
```

```
array(['Low', 'High', 'Low', ..., 'Low', 'Low', 'High'],
```

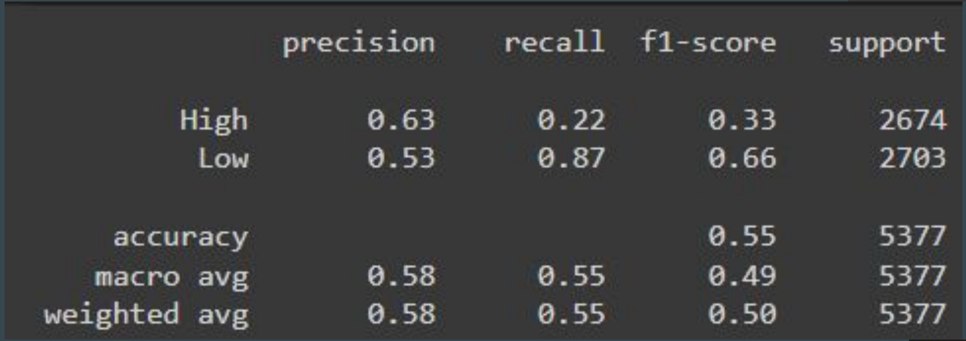

Test Error
0.35
0.34
0.33
0.32
0.31
0
5
10
15
20
25
30
35
40

AdaBoost

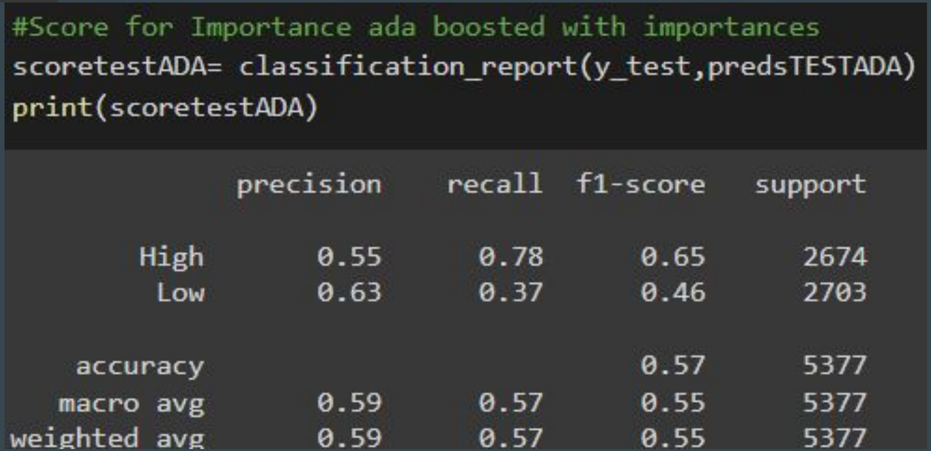


```
predADA
```

```
array(['Low', 'Low', 'Low', ..., 'Low', 'Low', 'Low'], d
```



```
              precision    recall  f1-score   support

        High       0.63      0.22      0.33      2674
         Low       0.53      0.87      0.66      2703

    accuracy                           0.55      5377
   macro avg       0.58      0.55      0.49      5377
weighted avg       0.58      0.55      0.50      5377
```



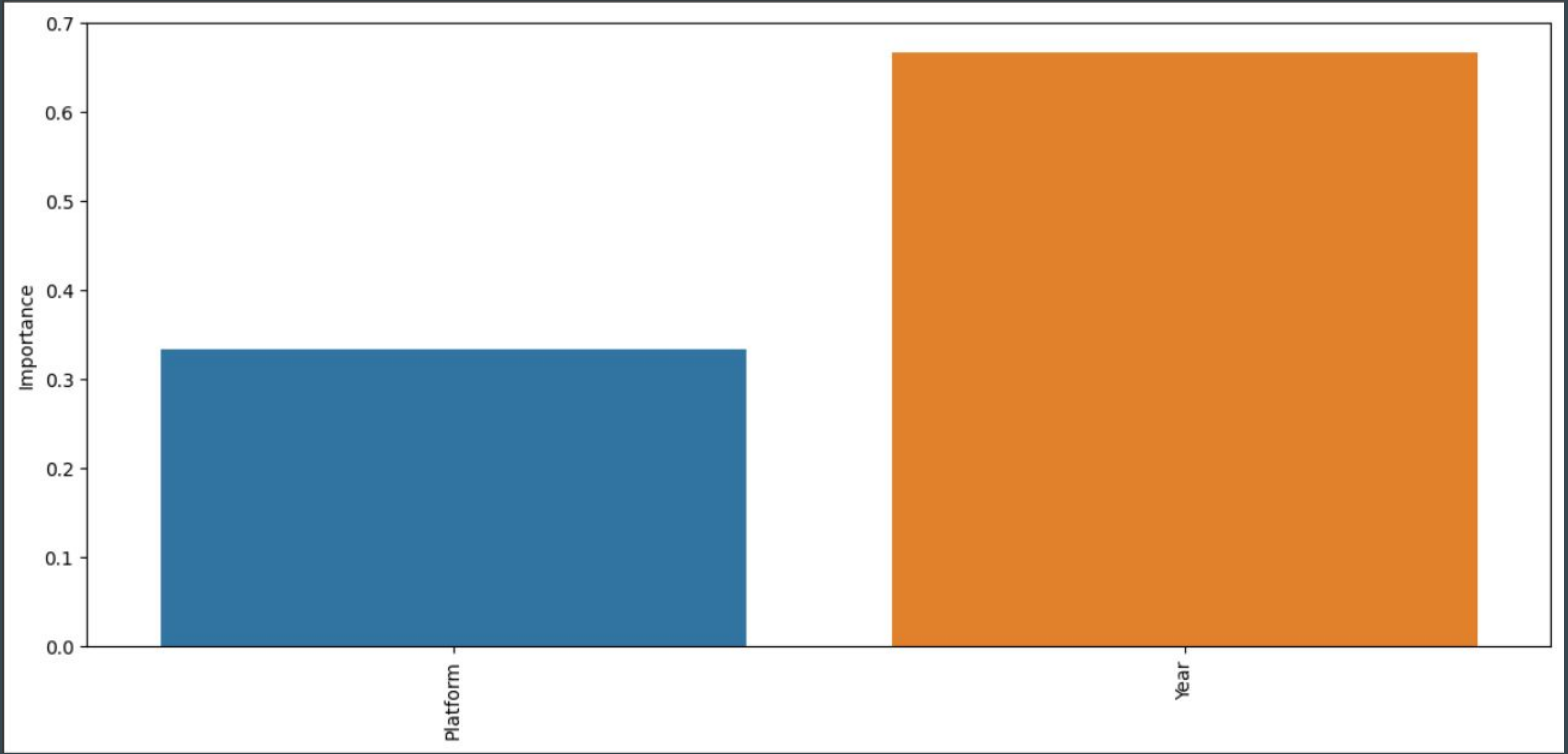
```
#Score for Importance ada boosted with importances
scoretestADA= classification_report(y_test,predsTESTADA)
print(scoretestADA)
```

```
              precision    recall  f1-score   support

        High       0.55      0.78      0.65      2674
         Low       0.63      0.37      0.46      2703

    accuracy                           0.57      5377
   macro avg       0.59      0.57      0.55      5377
weighted avg       0.59      0.57      0.55      5377
```

Feature Importances

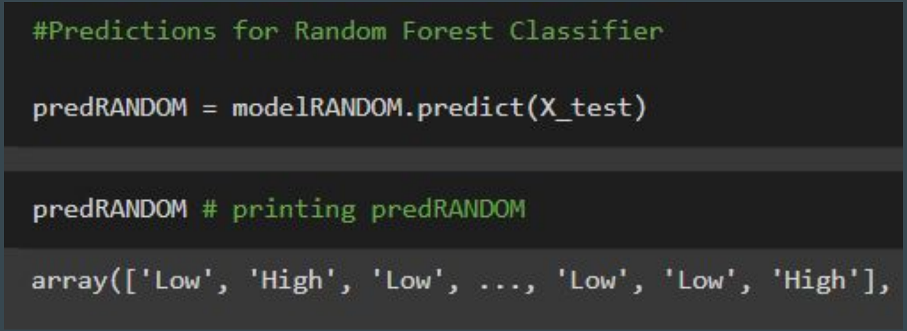
- For the Random Forest Model
  - Had Publishers as the highest importance
- Adaboost
  - Had year as the highest importance

AdaBoost Feature Importance



Conclusion

- Through our modeling, we found the Random Forest had the highest F1 score with 0.68



```
#Predictions for Random Forest Classifier

predRANDOM = modelRANDOM.predict(X_test)
```

```
predRANDOM # printing predRANDOM
```

```
array(['Low', 'High', 'Low', ..., 'Low', 'Low', 'High'],
```

Improvements

- Possibly hot-coding year
- Double Checking Dummies
- Further Extending Dataset
  - Adding Game Developers
  - Advertising budgets
  - etc.

Questions