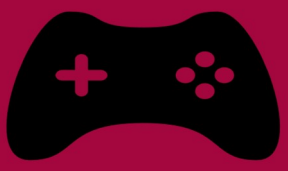
METACRITIC SCORE PREDICTION MODEL



Introduction

Background:

The video game industry continues to grow in popularity across the globe as our world becomes more technologically advanced. Year after year, video games become more accessible to the average person than at any previous time in history. With the continued saturation of video games by software companies, it has become increasingly difficult for a single game to gain notoriety. Metacritic is a website millions of consumers visit per day to view the best-reviewed games. Metacritic review score offers a way to gain notoriety in the mind of video game consumers.

The Problem:

The Metacritic score is only received after the game has already been released. By that time, it is to late to make changes to a video game that would impact its commercial success. In my report, I plan to give a detailed analysis of Metacritic review scores and to develop a predictive model for Metacritic review scores. This predictive model will allow company executives to anticipate how well received a game would be by industry critics before the launch of their title.

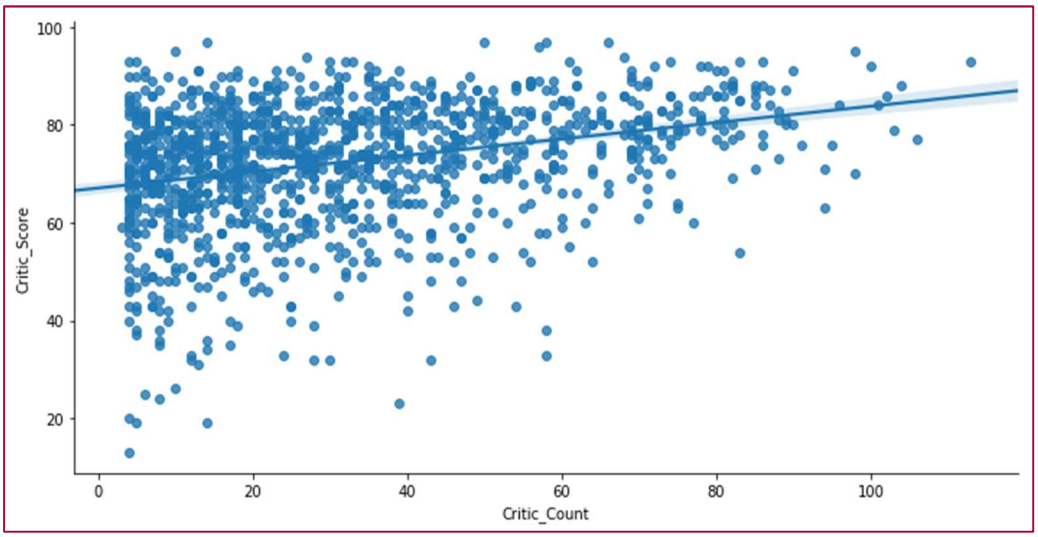
The Problem:

I began my analysis with a dataset available from Kaggle found at the following link: “https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings/data”. To build the best predictive model, my data set will only include video games that have been released after 2012. Video games review scores consider more aspects of a game now than in recent years. With that in mind, I decided to keep the data used to construct the model as current as possible. After importing the data set from Kaggle, I noticed several columns contained sales data which is irrelevant in terms of the project goal. I proceeded to drop all columns containing sales data from the data frame. Next, I discovered that there were several missing values in the user count and the user rating columns. I felt the best way to quickly deal with this problem would be to calculate the mean of each column and replace those missing values with the mean of all know

values. The 'ratings' column contained some missing values as well. The 'ratings' column contains data about each games ESRB rating. After some brainstorming, I couldn’t come up with a method to fill the missing values that would not jeopardize the model. The best thing to do in this situation is to drop the video games that have missing values in the rating column.

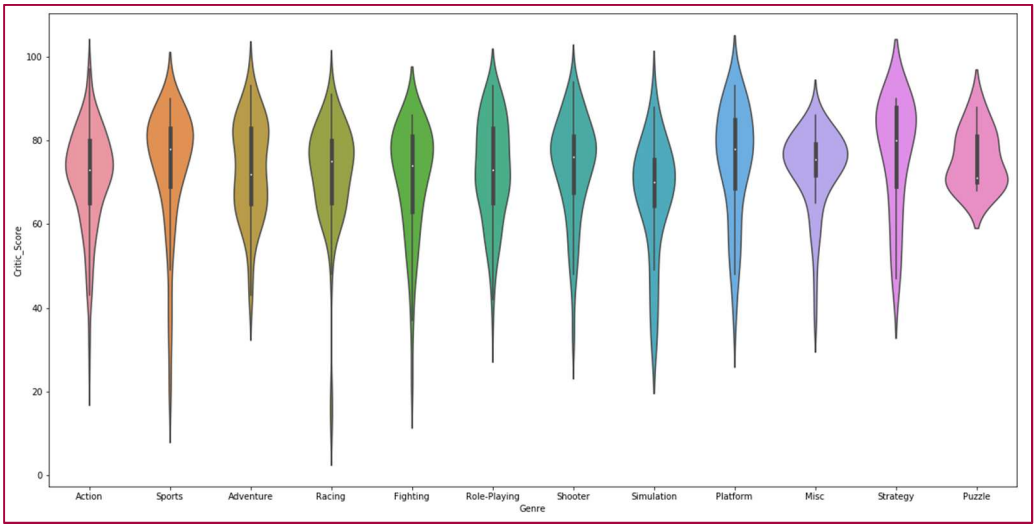
Exploratory Data Analysis:

CRITIC SCORE AND CRITIC COUNT



First, I wanted to explore the relationship between critic score (the target variable) and critic count. I came up with a scatter plot that displays a slightly positive relationship between critic count and critic score. Most games have between 5 - 60 game critic reviews. Next, I wanted to get a sense of which genre was on average rated the highest.

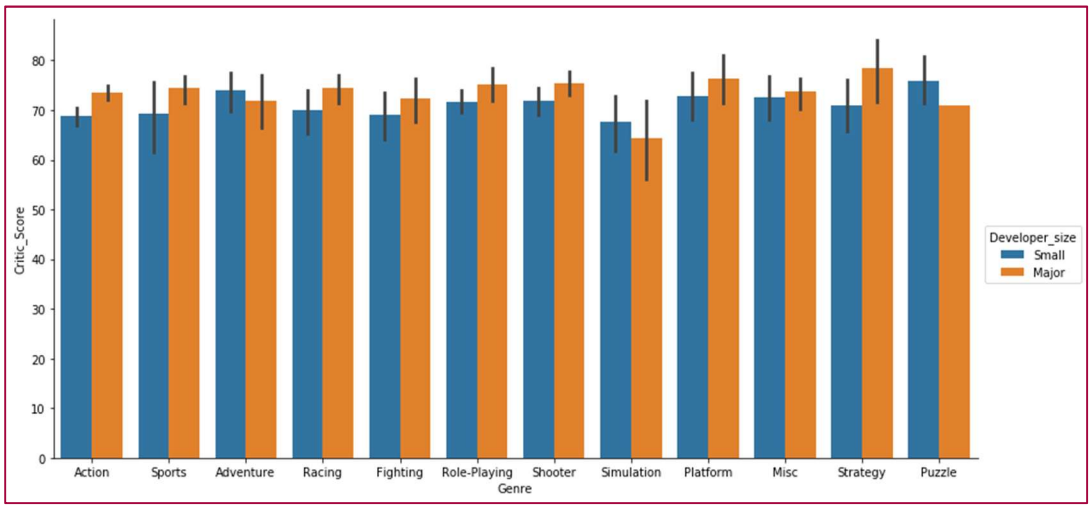
VIOLIN PLOT OF CRITIC SCORE AND GENRE



Looking at games by genre, there isn't any drastic difference in the average review score by genre. Racing games have the greatest variance in review score, while

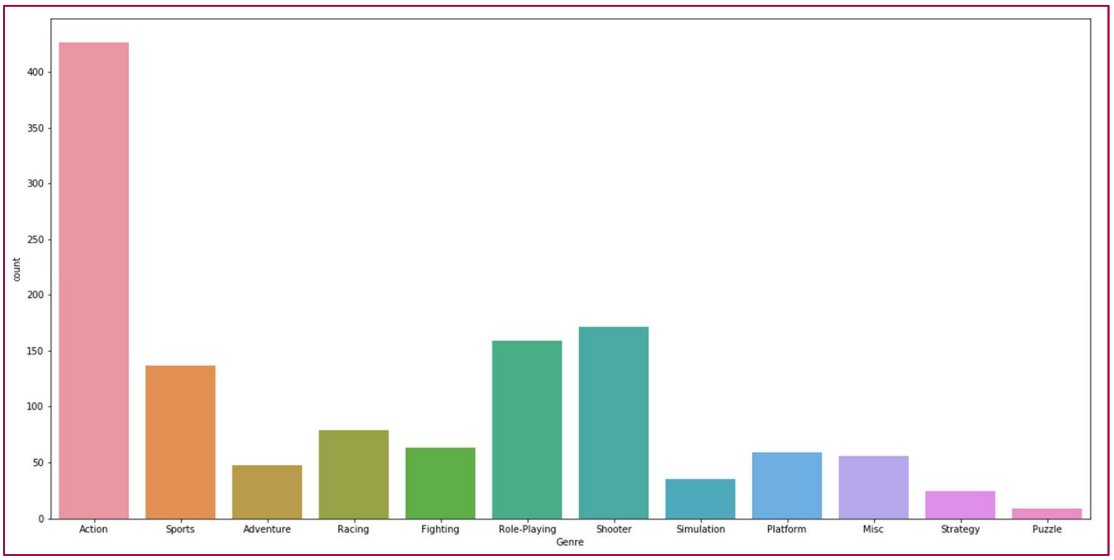
puzzle games have the lowest. The plot does show that games that fall into the Platform and Strategy genres are rated on average slightly higher than other categories.

CRITIC SCORE AND GENRE GROUPED BY DEVELOPER



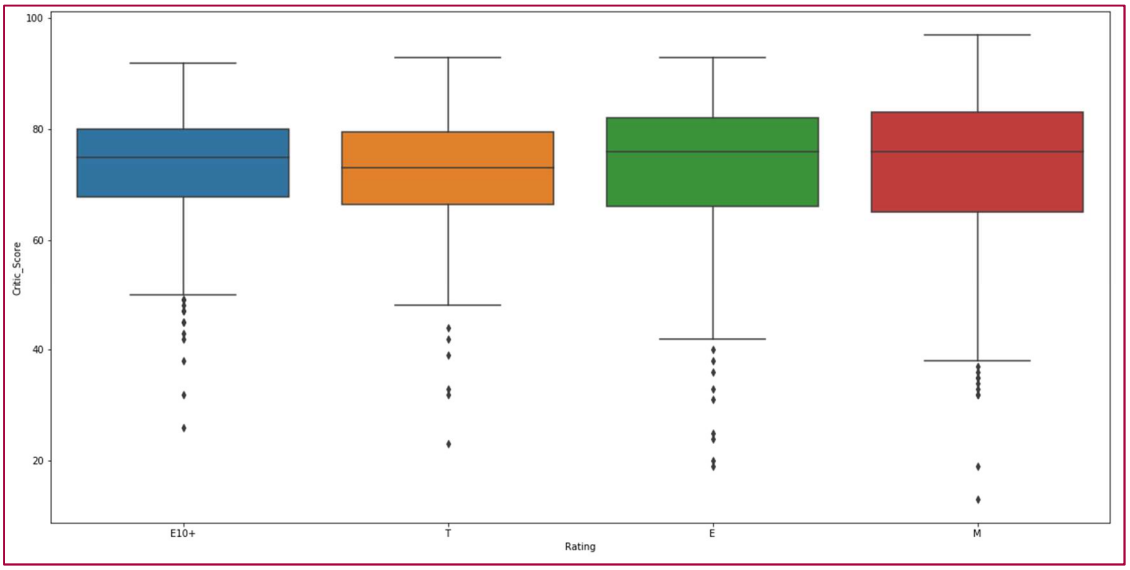
Taking a deeper look at critic scores by genre, I tried to split the games made by the major and smaller developers to see if there is a trend in the scores. The plot reveals that games developed by major developers are rated higher on average than games made by smaller developers excluding games in Adventure, Simulation, and Puzzle genres. I suspect that major developers have a bigger budget to work on games which would lead to higher quality games. If we were able to get access to the number of developers working on the game or the budget for development, we would be able to get a better understanding of what exactly causes major developer's games to be rated higher.

COUNT OF GAMES RELEASED BY GENRE



The most popular type of game released between 2012 and 2016 were action games. The number of action games released during that period is significantly higher than any other category. Following the action game genre, Shooter, Role-playing, and Sports are the next highest genres released.

CRITIC SCORE AND RATINGS CHART



The box plot of Critic Scores and Ratings shows ‘E’ and ‘Mature’ rated games are rated the highest and have similar mean critic scores. Although, the variance of ‘Mature’ rated games critic score is larger than the variance of ‘E’ rated games.

Inferential Statistical Analysis for Prediction Model:

To develop the best model for predicting Metacritic review scores on video games, I need to verify which variables have a statistically significant relationship with Critic Scores. A chi-squared goodness of fit test allows us to test the statistical significance of a relationship between categorical variables. I will run the test on several variables within the data frame to see which variables should be included when creating models. To be clear, for each variable that is tested our null and alternate hypothesis is as follows:

H_0: There is no relationship between the variable and Game Quality (Categorized Critic Scores).
H_A: There is a relationship between the variable and Game Quality (Categorized Critic Scores).

```
def Chi_test(data,c1,c2):
    #combine columns into matrix
    table = pd.crosstab(data[c1], data[c2])

    colsum = table.sum(axis=0)
    colpct = table / colsum
    cstable = stats.chi2_contingency(table)


    return print('Variable:', c1,'|',
                 'Chi Statistic:' ,cstable[0],'|',
                 'P-Value:', cstable[1],'|',
                 'Expected Value:', cstable[2], '|')
```

Variable	Chi Statistic	P-Value	Expected Value
Genre	61.9599558931	0.00166336164951	33
Platform	26.4503350982	0.493729754709	27
Rating	33.5884026466	0.00010543964857	9
Developer size	11.1890509966	0.0107463201464	3
Critic Count	436.95798475	3.28801958791e-08	288
Publisher size	14.2250362929	0.00261430484993	3

After running a couple of chi-squared goodness of fit tests, I found that the variables 'Genre', 'Rating', and 'Critic Count' have a statistically significant relationship with 'Game Quality'(Categorized Critic Scores). The 'Platform' variable doesn't have a statistically significant relationship with 'Game Quality' based on the p-value of 'Platform' surpassing .05 alpha threshold. The test of 'Developer size' (Whether a game was made by a major developer or not) and 'Publisher size'(Whether a game is published by a major publisher or not) relationship with 'Game Quality' isn't valid since the expected value for both variables is 3 and need to be greater than 5 to be considered valid.

Next, I wanted to explore the relationship between critic score and YSR (years since platform release). I predict that there should be a positive correlation between YSR. As consoles get older, scores should increase as developers become more familiar with each platform capability. Surprisingly, I found almost no correlation between Critic Score and YSR. The calculated correlation from the data set is -0.04164. This leads me to believe that YSR should be left out of the variables used to construct my predictive model.

Lastly, I wanted to determine if the difference in proportion of high quality games developed by major developers and smaller developers is statistically significant. I decided to run a two-sample proportion test. I predict that there will be a significant difference in the number of high quality games created by major developers and smaller developers.

H_0: The null hypothesis is that there is no difference between the two proportions.
H_A: The alternative is that there is a statistical difference between the two proportions.

The test resulted in a z-statistic of 9.2869213766 and a p-value of 1.58815467642e-20. The P-value from out proportion test is extremely low. We can conclude that there is a statistically significant difference between the number of high quality games generated by major and small developers. There is sufficient evidence to include this variable in our model development.

Model Construction:

Before we start to fit the model, we will need to do some feature selection to select the relevant features. I used the function “variance_threshold” to identify feature with low variance among our chosen predictors. Since we are dealing with mostly categorical data, I created dummy variable for each column of categories. The data frame was fed through a function that looks for columns with low variance and deletes them from the data frame called “get_low_variance_columns”. The function didn’t delete any columns from the data frame since none of the features had near zero variance. In the first attempt to create a model, I decided to give each developer their own feature column. The result from the first model construction are below:

OLS Regression Results

Dep. Variable:	Critic_Score	**R-squared:**	0.714
Model:	OLS	**Adj. R-squared:**	0.575
Method:	Least Squares	**F-statistic:**	5.132
Date:	Wed, 07 Feb 2018	**Prob (F-statistic):**	1.90e-90
Time:	21:00:50	**Log-Likelihood:**	-4223.1
No. Observations:	1267	**AIC:**	9276.
Df Residuals:	852	**BIC:**	1.141e+04
Df Model:	414		

There are a couple of minor issues with our first model. Although, the main problem is the huge discrepancy between our R-Squared and Adjusted R-squared. We also have to many variables that are correlated in the model. This problem signals that we may have to many features in our model that don’t add that much predictive power. Although, we know from our statistical analysis that developers play a key role in predicting Metacritic review scores. We need to find a way to preserve the information that developers provide in predicting review scores while reducing the number of columns. I came up with an idea to find the median critic score for each developer and categorize the median score into categories based on the bins from the Metacritic website. The categorized medians scores for developers were then mapped to each game and placed in the column called ‘Game_Hist’.

```
# group by developer and calculate median critic score:
Devs = pd.DataFrame(VGR.groupby(by='Developer').median()['Critic_Score'].sort_values()).reset_index()

Devs = Devs.rename(index=str, columns={"Critic_Score": "D_median_cs"})

# Create Bin Names for Analysis :
g_names = ['Poor','Average', 'Good', 'Great']

#Bins Limits
bins = [0, 50, 75, 90, 100]

# Sort Critic_Score into Game_Hist:
Devs['Game_Hist'] = pd.cut(Devs['D_median_cs'], bins, labels=g_names)

VGR = VGR.merge(Devs, on='Developer')
```

By including 'Game_Hist' in our model, we use developer's median critic score (Which isn't influenced by outliers) to help predict the score of their next game. For example, developer 'Naughty Dogs' has a track record for making outstanding triple A games. One could be certain that next game naughty dog develops will have close to or the same standard of their previous games.

The 2nd model's data frame includes the variable 'Game Hist' and removes developer columns. Before we train our model and predict new values, we must split the data we have on hand into a training and test set. Once the Data has been split, we fit our model to the training set and predict on the test set and training set.

```
Score on training set: 0.538210092853
Score on test set: 0.487923291325
```

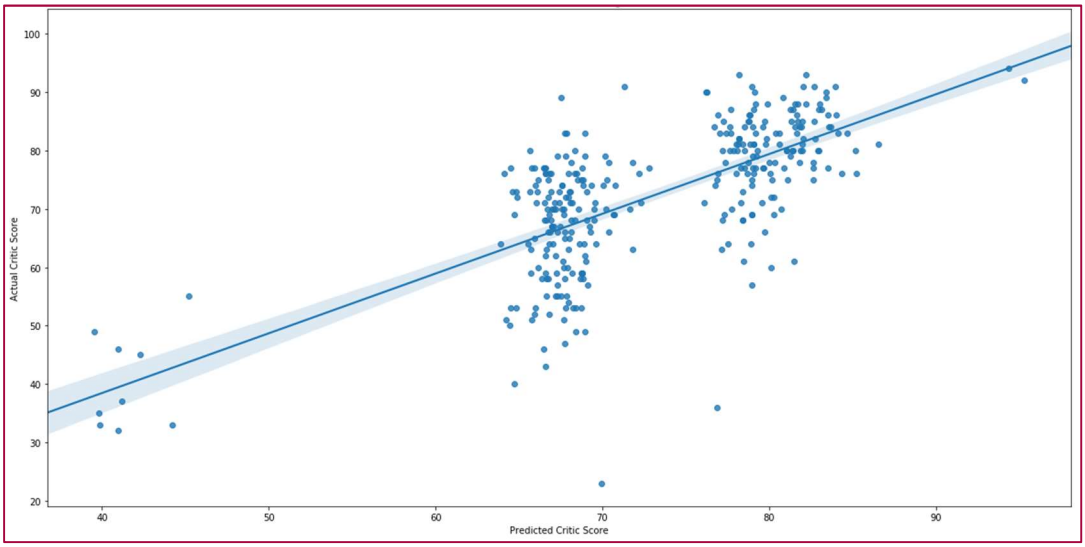
As you can see above, the results show that the R-squared dropped slightly but we were able to get rid of our multicollinearity problem. The R-squared metric for our model increased on the training set. For the test data, the R-squared value is .487.

STRONGEST FEATURES

	features	estimatedCoefficients
19	Game_Hist_Great	50.364446
18	Game_Hist_Good	37.634153
17	Game_Hist_Average	25.606009
12	Genre_Strategy	2.455028
3	Genre_Fighting	1.471696
6	Genre_Puzzle	1.272746

Looking at the strongest positive predictor in our model, 'Game Hist' plays a crucial part in determining the critic scores. This shows that prior developer success in video games helps predict future success.

PREDICTED VS ACTUAL CRITIC SCORE



Looking at the plot of test set predicted critic scores vs. actual critic scores, we can see that most of our predicted values are clustered around certain areas of the plot. We would expect some variation between predicted and actual values. We will need to either transform the target variable or include a new variable in our model. Since our model predictive range is only between 0 – 100, our best approach is to try and include another predictor from our data set. After tirelessly trying over 15 different combinations of variables, I found that no other combination of variables offered a better fit to the training and test data then our previous combination. This leads me to believe that we don't have enough information in the data set to increase the predictive power of our model. Perhaps using random forest regressor algorithm to predict critic score, will yield a higher score.

Before the model is fitted, the hyper parameters need to be optimized using grid search cross validation. I found our optimal hyper-parameters to be max_depth: 8 and n_estimators: 500. Using the optimal parameters, I fit the model to the training set and predict on test set. The random forest regressor model didn't offer a significantly higher score on the test set. Leading me to believe that regardless of the algorithm used, there isn't enough information to create an improved prediction model.

```
The best parameter {'max_depth': 8, 'max_features': 'sqrt', 'n_estimators': 500} for model with a score of 0.53
```

```
N_est = grid.best_params_['n_estimators']
max_d = grid.best_params_['max_depth']
m_feat = grid.best_params_['max_features']

rf = RandomForestRegressor(n_estimators=N_est, max_depth=max_d, max_features='sqrt', random_state=15)
model = rf.fit(X_train, y_train)
```

```
from sklearn.metrics import r2_score
from scipy.stats import spearmanr, pearsonr

predicted_train = rf.predict(X_train)
predicted_test = rf.predict(X_test)
test_score = r2_score(y_test, predicted_test)
spearman = spearmanr(y_test, predicted_test)
pearson = pearsonr(y_test, predicted_test)

print(f'Test data R-2 score: {test_score:.3}')
print(f'Test data Spearman correlation: {spearman[0]:.3}')
print(f'Test data Pearson correlation: {pearson[0]:.3}')
```

```
Test data R-2 score: 0.496
Test data Spearman correlation: 0.685
Test data Pearson correlation: 0.709
```

After some careful consideration, I thought if I couldn't predict the exact critic score maybe I could predict the category the critic score would fall into. Information available from the Metacritic website states critic scores fall into 5 distinct categories. By predicting the category, the games critic scores would fall in based on known features, I could predict the range of the scores and be able to provide a better prediction. Using a linear support vector classification model, I train the model on the training set and predict on the test set.

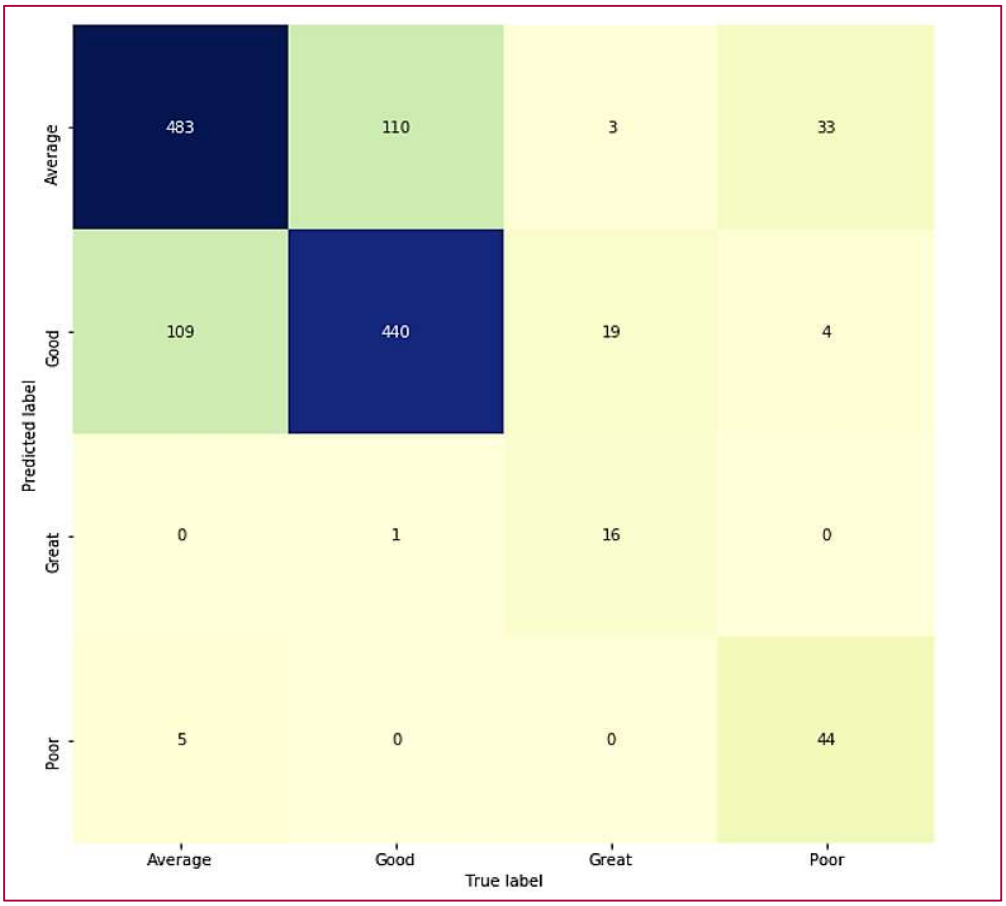
```
The best parameter {'C': 0.01, 'loss': 'squared_hinge'} for LinearSVC model with a score of 0.77
```

```
# Fit Model with Using optimized Parameters:
C = grid.best_params_['C']

# Create SVM classification object
model = svm.LinearSVC(C=C)

# Fit Data
model.fit(X_train, y_train)

#Print Score
print('Score on Training set:',model.score(X_train, y_train))
print('Score on Test set:',model.score(X_test, y_test))
```

```
Score on Training set: 0.782105263158
Score on Test set: 0.757097791798
```

The classification model works well when predicting on the training and test set. Looking at the predicted label vs true label plot below, it looks like our model has some problems predicting poor and great games. We can see from the plot a portion of great and poor games are miss classified but still works well.

PREDICTED VS TRUE LABELS:



Interpretation:

Our original goal for this project was to build a model that could predict Metacritic scores. The model with the highest score was our random forest regressor model when trying to predict critic score. I was able to use the information we have on hand to come up with a relatively good classifying model to predict the category a Metacritic score of a game would fall into. The classification model works better as a prediction model than our linear model for the goal of this project. This is to be expected since the model is only trying to predict which category of 'Game Quality' (Categorized Critic Scores) a game would fall into giving us a predicted range of critic scores. The classification model is just as valuable to executives because it still provides information needed to make key decisions. Executives could decide based on this model that a game which is predicted to fall in the poor category should spend a little more time in development. Thus, saving the game from an unfavorable review score.

Insights:

1. We need more information on the features of games to predict critic score. For example, features of a game such as presences of an online multiplayer and presence of microtransaction can have an impact on player experience which influences critic scores.

2. Higher Metacritic scores have a higher number of individual critic score involve in the calculation. Provide more critics before public release with a copy of you game for review.

3. We need more information about the Developers. Developers play a crucial part in determining critic score. If we had more information on the size of the teams working on a game or the time spent developing a game, we may be able to predict critic score with more precision.

4. By predicting the category (Great, Good, Average, and Poor), The classification model offers just as much insight as predicting the precise critic score would. An executive could decide games predicted to fall into the poor score range should spend more time in development refining feature to increase player experience.