Automated Art Analysis

Eric Evje

05/06/2020

Introduction

In 1933 George D. Birkhoff published a book entitled "Aesthetic Measure"[1] in an attempt to describe a mathematical theory of aesthetics. The notion he proposes revolves around the measure of order and complexity of an image or other work of art. The basic equation can be written as:

$$M = f(\frac{O}{C})$$

Where M is aesthetic measure, O is the measure of order in the object, and C is the measure of complexity in the object. Birkhoff's interest in aesthetic measure began with music, and continued into simple shapes, tiling, vases, and other works of art [2].

Though there are ways to analytically measure order and complexity of simpler shapes and objects, these become impossible for more complex works of art, in the case of this paper, paintings. With the advent of digitized images and information theory, a simple metric developed for order and complexity of images based upon comparing the maximum entropy of an image vs the compressed entropy of an image[3]. In practice this is accomplished by comparing the file size of an image after compression using a standard compression algorithm (such as .jpeg) and comparing to the maximum entropy which related to the possible color space (or palette) of the image. This paper uses two metrics adopted from the paper, "Conceptualizing Birkhoff's Aesthetic Measure Using Shannon Entropy and Kolmogorov Complexity", one incorporating an understanding of Shannon entropy in terms of the color space of the image weighted against the maximum entropy of the color space, and another method relying on Kolmogorov's complexity. This is defined as the maximum theoretical entropy of the image against the actul entropy of the image. The computations are as follows:

For aesthetic measure based on Shannon entropy, the equation used herein is:

$$M_b = \frac{H_{max} - H_p}{H_{max}}$$

Where M_b is the aesthetic measure from [0, 1], and H_{max} is the maximum theoretical entropy of the color space. Since we are converting the images to greyscale prior to analysis, the maximum entropy is $log_2(256) = 8$ [3]. H_p is the entropy of the image, which is defined as:

$$H_p = -\sum p(x)log(p(x))$$

Where p(x) is based upon the distribution of the pixel values in grayscale [3].

For aesthetic measure based on Kolmogorov complexity we used the equation:

$$M_k = \frac{NH_{max} - K}{NH_{max}}$$

Where M_k is the Kolmogorov complexity from [0,1], N is the number of pixels, H_{max} is the maximum information per pixel. In this case, since the full color image was used, $H_{max} = 8 * 3 = 24$. K is Kolmogorov complexity. For this analysis, this was the file size of the compressed image in bytes.

The goal of this paper is to use image metadata as well as the aesthetic measures of a variety paintings and compare them to the average art rating of the painting. The hope is to find a method to accurately predict the art rating of new paintings.

Data Generation

This paper leverages survey data collected by Saif Mohammed through the National Research Council Canada [4]. The dataset is comprised of 4,105 annotated works of art available through wikiart.org. The variables from the paper included in the model building were as follows:

- *Rating* - The average rating given by the study participants on a scale from -3 to 3, the higher the number the more positive the reaction to the art.
- *Style* - A 6 factor variable with styles ranging from renaissance art to modern art. These were encoded as 5 dummy variables.
- $Year$ - The year the painting was painted. Some art work had a range, or were narrowed down to a century. For the ranges, the start year was chosen. The artwork with only a century as year were removed from the analysis.
- $Face$ - Whether or not the artwork contained a human and/or animal face and/or body. 1 indicates presence, 0 indicates no presence.

Data Cleanup

A python script was used to download the images to a local drive for faster analysis (see appendix for script). Some images failed during analysis or download and were removed from the dataset. An important note here is that greyscale images broke the analysis pipeline, so all black and white paintings were removed. As stated above, some of the artworks did not have correctly formatted year data. These were either cleaned up or removed.

Image Analysis

Once all of the images were local they were processed using R to find the following statistics:

- $mean_{red}$ - the mean value from the red channel of the image's histogram
- $mean_{green}$ - the mean value from the green channel of the image's histogram
- $mean_{blue}$ - the mean value from the blue channel of the image's histogram
- $median_{red}$ - the median value from the red channel of the image's histogram
- $median_{green}$ - the median value from the green channel of the image's histogram
- $median_{blue}$ - the median value from the blue channel of the image's histogram
- sd_{red} - the standard deviation value from the red channel of the image's histogram
- sd_{green} - the standard deviation value from the green channel of the image's histogram
- sd_{blue} - the standard deviation value from the blue channel of the image's histogram
- per_{red} - the percent of the image intensity from the red channel
- per_{green} - the percent of the image intensity from the green channel
- per_{blue} - the percent of the image intensity from the blue channel

- per_{edges} - the percent of the image after thresholding that is 1, indicating an edge in the image
- M_k - The aesthetic coefficient based upon Kolmogorov complexity
- M_{kedges} - The aesthetic coefficient based upon Kolmogorov complexity completed on an edges only image (see Figure 2)
- M_b - The aesthetic coefficient based upon Shannon Entropy

A script was used to automated the process (see appendix for script). The mean, median, and standard deviation of the color channels is a straighforward mean, median, or standard deviation calculation on all of the values in the respective color channel. The percent values are the ratio of the sum of the respective channel against the other two channels. Edge finding was performed by applying a high pass filter to the image followed by a thresholding routine. The ratio of high and low pixels was calculated for per_{edges}. The aesthetic coefficients were calculated as described in the introduction. The three images below show an example of an original image, the result of the edge finding algorithm for the image, and the histogram of the color channels of the original image.

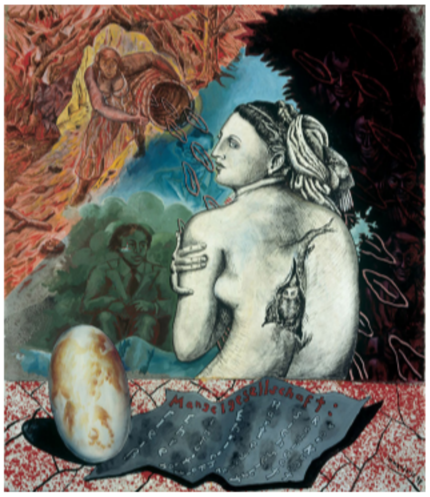


Figure 1: Original example image

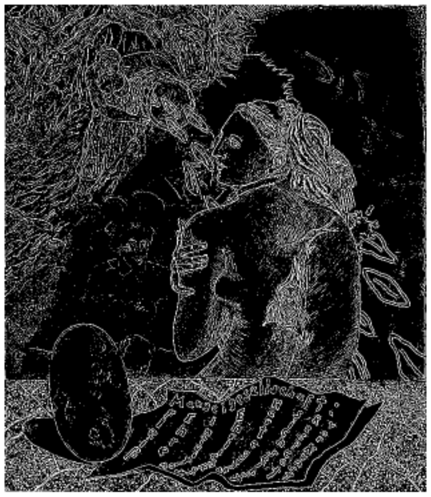


Figure 2: Results of image edge finding algorithm

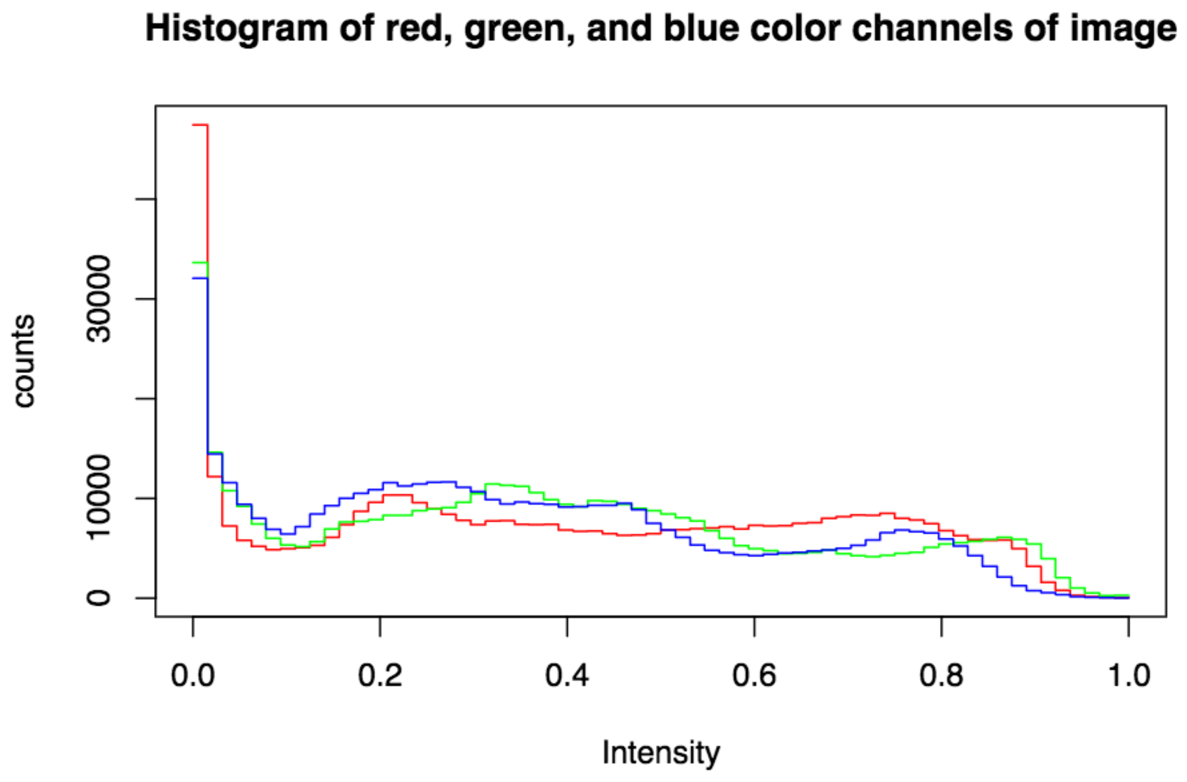


Figure 3: Redm, green, blue channel histogram for the above example artwork

Model Building

OLS Linear Regression Using All Possible Regression Algorithms

The first model fitted used OLS linear regression by searching all possible regression models as well as the interaction between all variables and M_b. The notion for the inclusion of the interaction term was the hypothesis that expectation of art is an important aspect of the viewer's rating of a work of art, and thus their expecation on the ratio of order over complexity may change (e.g. a viewer of modern art may expect less entropy in the art than compared to a renaissance piece). Before fitting, the data set was broken up into a training and test data set at 75% and 25%, respectively. After all possible regressions was complete, the lowest cross validation score was found, and that model was fit to the training data. This was then used to predict values for the test data set and find RMSE of the model.

After the first full model fit, it was evident there were several variables with multicollinearity, and in fact one that was aliased. We first removed per_{blue} from the model since it was an alias of $per_{red} + per_{green}$. After removal of per_{blue} and the dummy variables for analysis, the variance inflation factors were run for the full model.

Table 1: VIF values for the full model

	VIF
Year	1.43
face	1.47
mean_red	70.50
mean_green	147.62
mean_blue	73.59

	VIF
median_red	26.48
median_green	47.63
median_blue	31.55
per_red	13.24
per_green	11.50
sd_red	3.86
sd_green	6.87
sd_blue	4.18
per_edges	5.25
k_vals	1.26
k_edges	6.06
Benses	1.59

Eventually, $mean_{green}$, $mean_{blue}$, $median_{green}$, and $mean_{red}$ were removed from the model as well, VIFs were recalcaulted and shown below.

Table 2: VIF values for the reduced mdoel

	VIF
X	1.00
Year	1.42
face	1.46
median_red	6.29
median_blue	7.23
per_red	5.62
per_green	1.58
sd_red	3.27
sd_green	5.98
sd_blue	3.98
per_edges	5.25
k_vals	1.26
k_edges	6.05
Benses	1.58

The data was then split into the test and training datasets and all possible regression models were run to find the minimum CV.

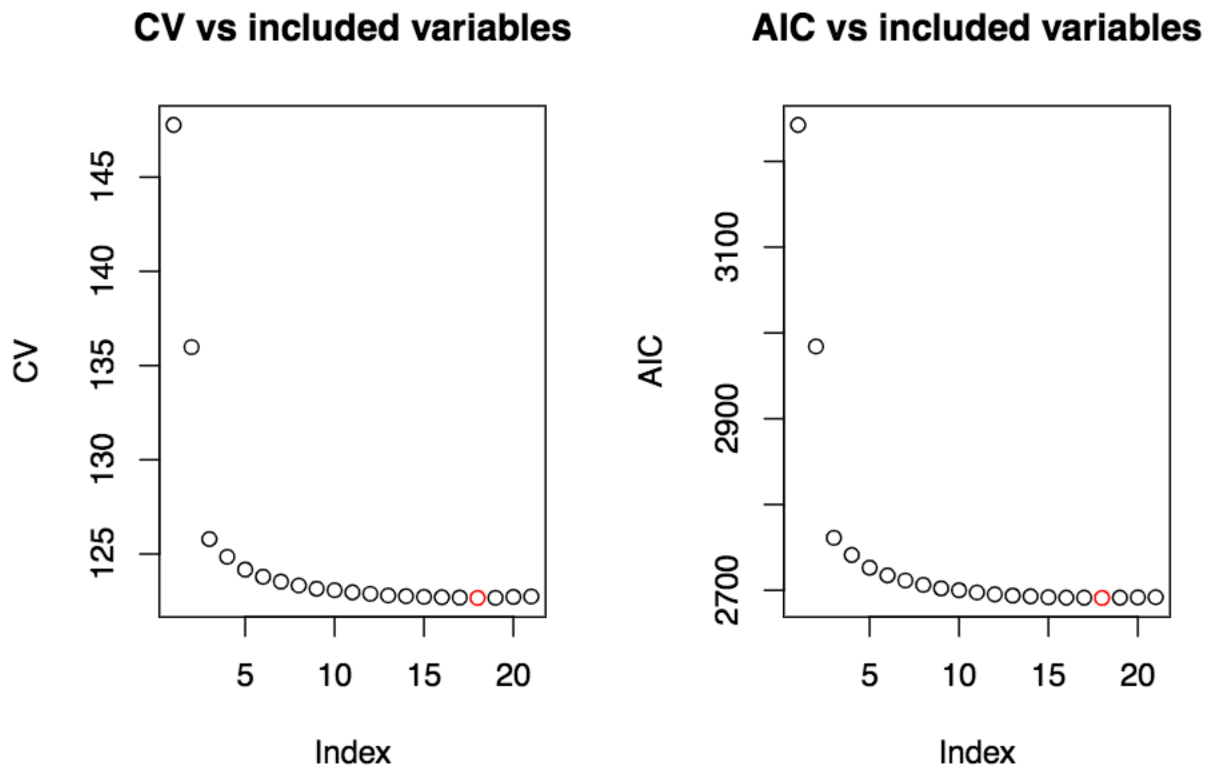


Figure 4: Plots of CV and AIC for the best model at each level of included variable.

Both metrics (CV and AIC) point to models at index 18 (in this case a model with 18 predictor variables) as the model with the best predicitve capabilities (see the red data point in Figure 4). This model was then used to predict *Rating* from the test data set and compare the actual values. The figure below shows a plot of fitted vs actual values.

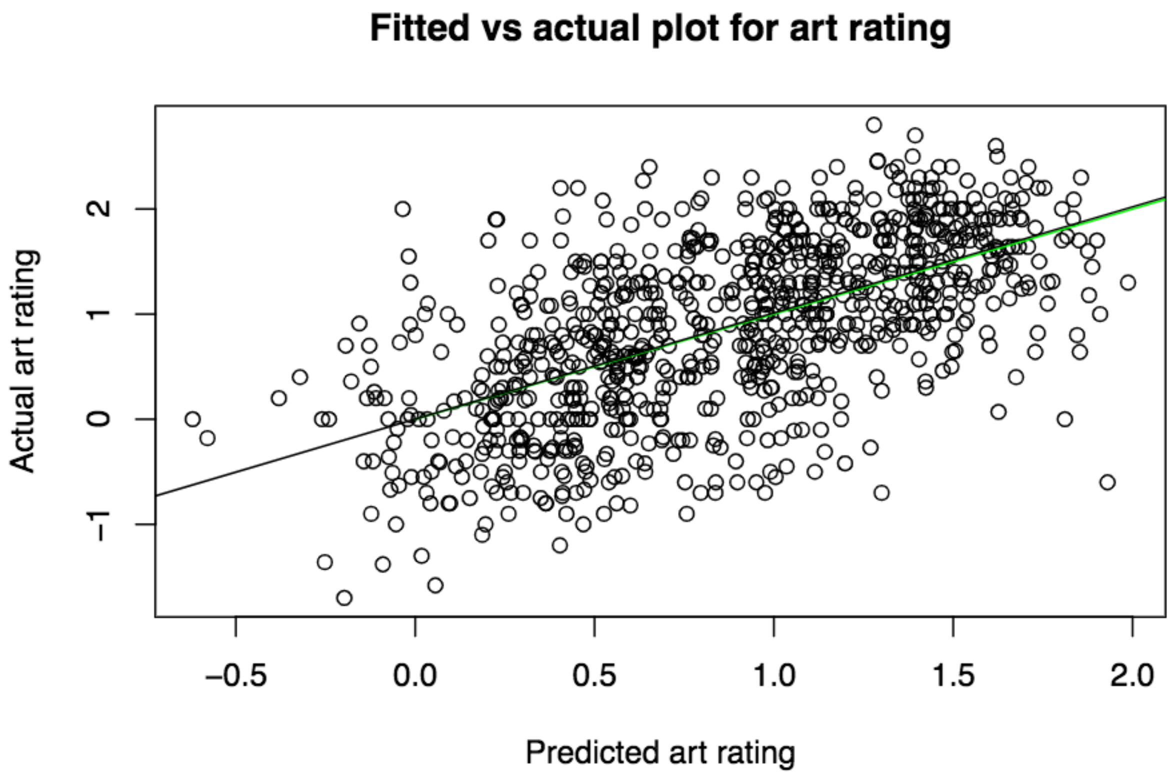


Figure 5: Fitted vs actual plot for best OLS regression model

The plot seen above are the predicited values of average art rating plotted against the actual values. The black line is the fit line and the green line is a line with slope 1 indicating good correlation across the range. The RMSE was $RMSE = 0.6717211$. The coefficients were:

Table 3: Beta terms used in model

	Beta Coefficients
(Intercept)	1.3574
Year	-0.0016
face	0.4048
modern	0.7457
postrenaissance	0.7217
per_green	6.8655
k_vals	0.4801
k_edges	-0.5806
Benses	3.9568
modern:Benses	-1.0523
per_green:Benses	-14.4273
Benses:per_edges	-2.7976

Next, we look at regression diagnostics for the model, as seen in the plots below.

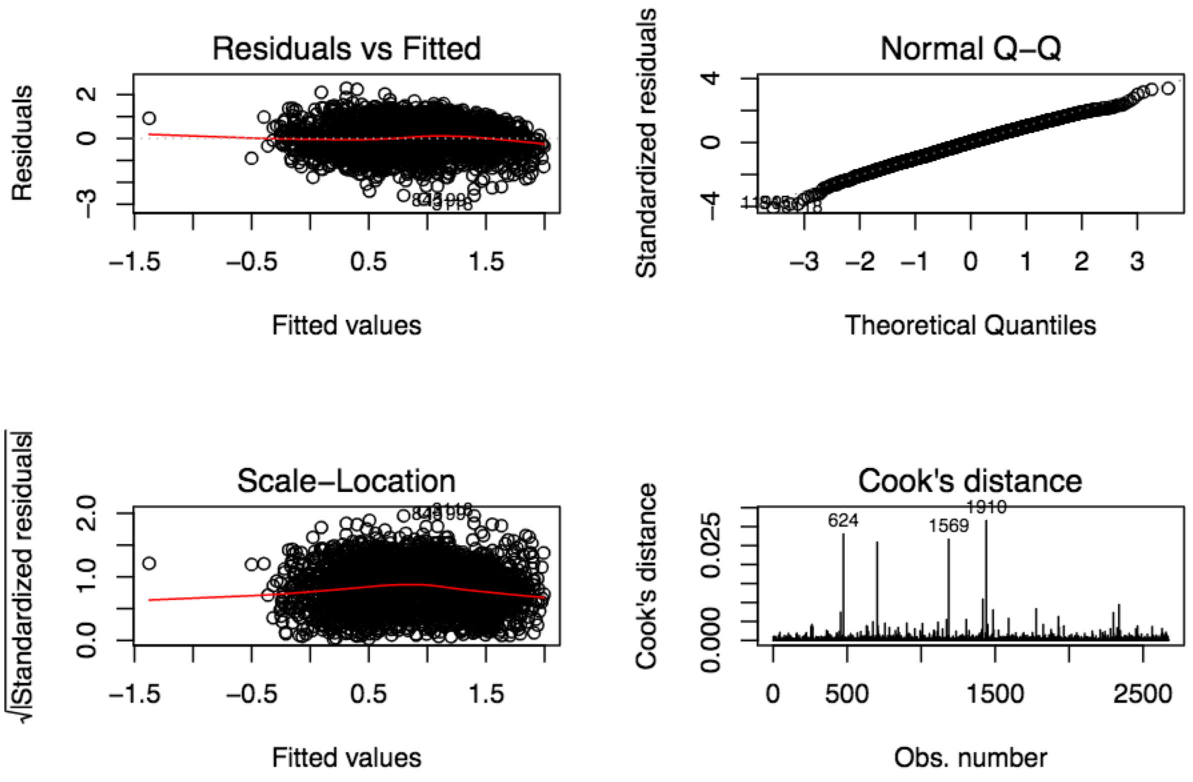


Figure 6: Regression diagnostics for the best fit linear model found with OLS

Looking at Cook's distance, there are 3 flagged data points, 624, 1569, 1910. Their data is presented below:

Table 4: Data of outliers included in the model

	Rating	Year	Face	Modern Art	Post Renaissance Art	median red	% green	M_k	M_b
624	-0.45	1971	0	1	0	1.0000	0.4472	0.9680	0.9135
1569	-1.40	1962	0	1	0	0.0510	0.6828	0.9671	0.5762
1910	-0.40	1968	1	1	0	0.7137	0.1039	0.9514	0.6391

It does appear that 624 has a $median_{red}$ value of 1. Looking at the painting, it actually has no red in it, as seen below. The M_b value is also unusually high, most likely due to the lack of red in the image.

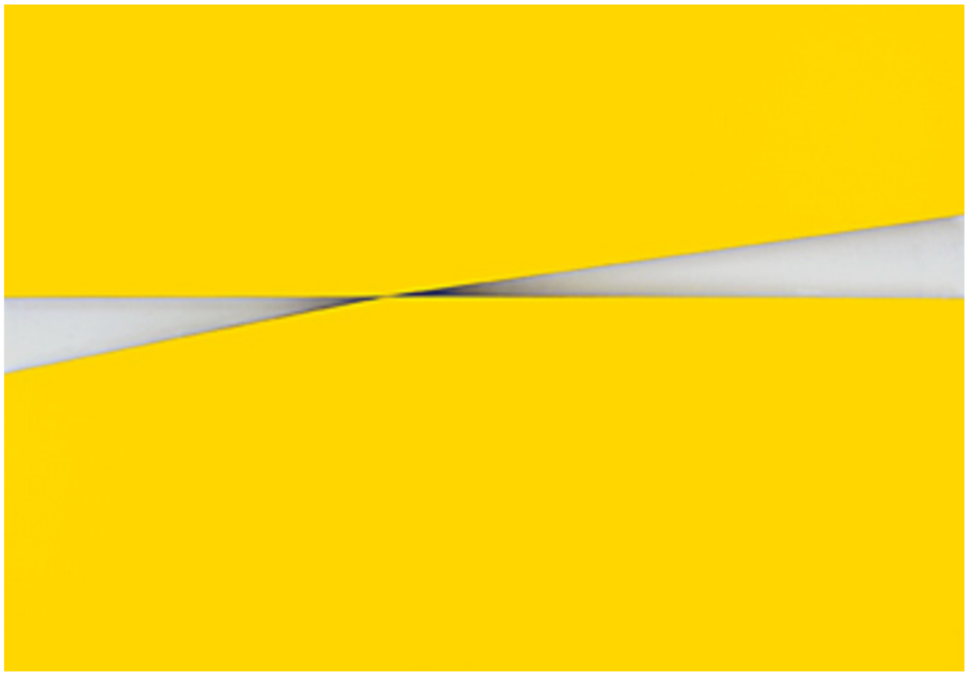
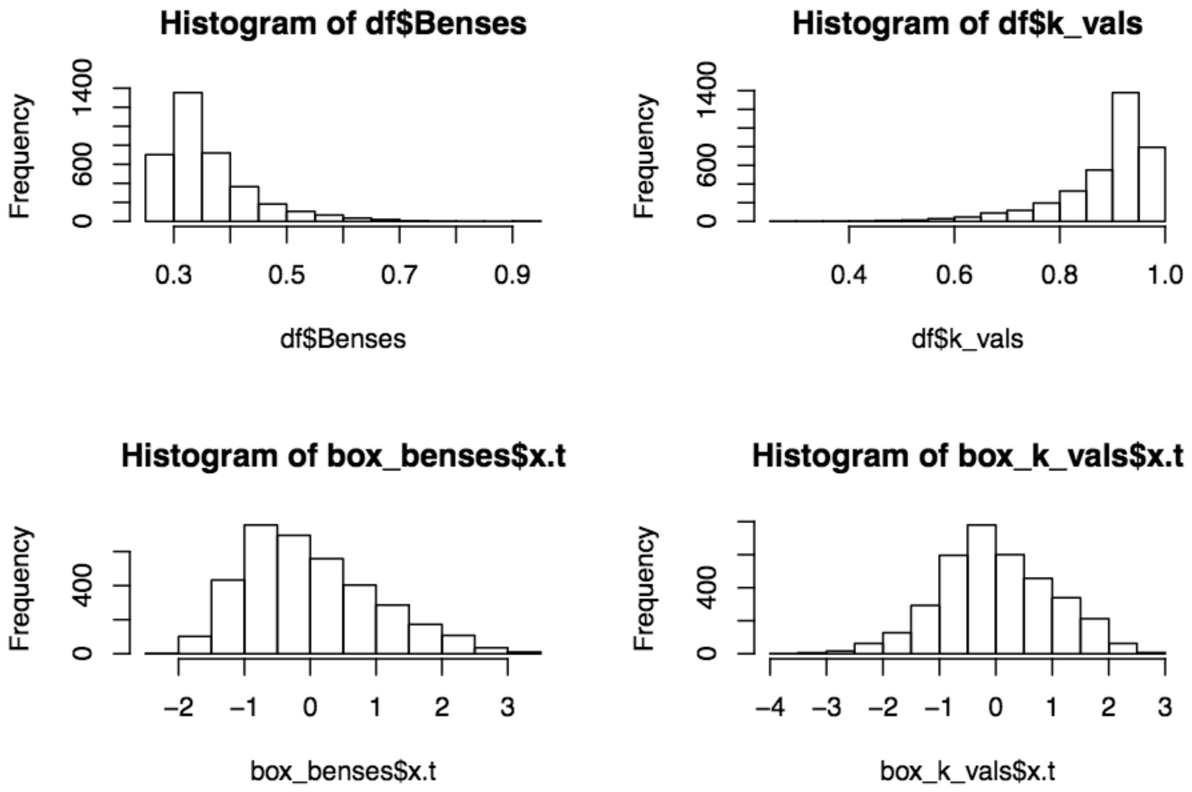


Figure 7: Outlier image 624

The model finding was re-run with the outliers removed. The new model without outliers had an RMSE of $RMSE = 0.6899344$, which is greater then the original RMSE with outliers. It is unlikely removing outliers would improve the performance of the model further.

Using a methodology of computing all possible regressions for the full model plus interaction effects with the Benses aesthetic measure, M_b, we were able to achieve an RMSE of $RMSE = 0.6717211$. However, this was time intensive, still has some linear dependcies within the model, and is sensitive to influential points, of which there are several. Another issue is the absence of transformed data. For example, the Benses measure distribution and the Kolomogorov measure distribution are not normal, concentrated at the edges of the range of the datasets. Using the $bestNormalize$ package, optimal transforms were taken to normalize the data as best possible. Ordered quantile technique was not used since it is difficult replicate an ordered transform with new data, although possible with the $bestNormalize$ package. Box-cox transformations were chosen for both measures, although the kolomogorov value was transformed as $1 - M_k$ to skew the data correctly for the transform to work. The figures below show the transformed data.



Model building using Lasso Regression

For performing regression using OLS and all possible regression algorithms to compute the best predicitive model it was clear there were many instances of multicollineariy, and the variable space was too large to compute all possible regression models for a full model with all interaction terms. Therefore, lasso regression was tried next to find a more predicitively powerful model. Ridge regression was not attemped since it was already apparent from the previous model building exercise that many of the variables were not significant and should not appear in the final model. Lasso regression can completely remove variables from the model, unlike ridge regression. Elastic-net was also not attempted since the computation time would have been extremely long with 25^2 variables.

To obtain a regression model using lasso, a λ value is required, which can be found automatically using the *glmnet* R package. In order to verify the predicitve perofmance of the model, a test and training group are created with 75% of the data and 25% of the data, repectively. A Lasso model is then generated of all first and second order interaction of the model. This results in 300 variables. Finally, the model is generated and checked against the test set for RMSE, which we will use to compare against the model found in the previous section. To the data we allso added the box-cox transforms of M_b and M_k as described in the previous section. The figure below shows the outputs of the lasso regression, namely the plot showing the search for the optimal value of λ.

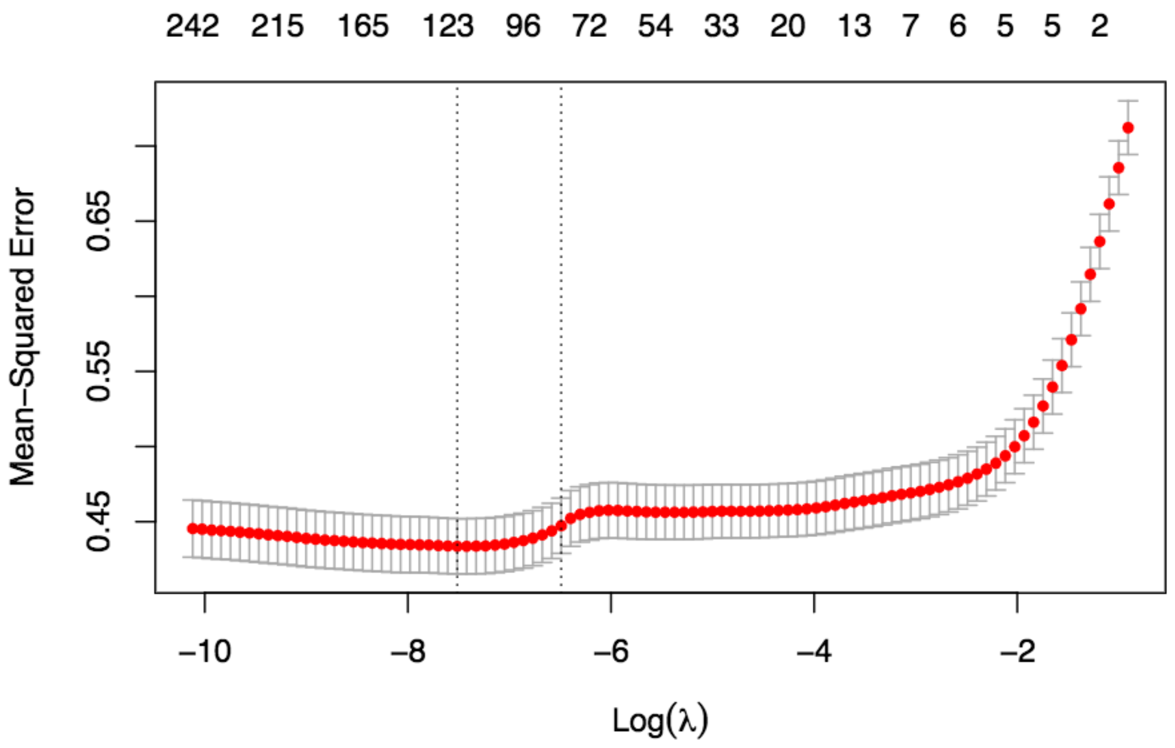


Figure 8: Cross validation plot to find ideal value of λ

The RMSE for the second order interaction model found using lasso regression was $RMSE = 0.6472936$ while the RMSE for the full model with one interaction term was $RMSE = 0.6717211$. It is is clear that the lasso regression model has better predictive capabilities than the all possible regressions method. Lastly, we attempted elastic-net models to see if performance can be increased further.

After running the training model to look for the optimal value of lambda, it was apparent that $\alpha = 1$ with $\lambda = 0.00042$ are the ideal values. Therefore, the model obtained from the lasso regression was the optimal model.

Logistic Regression

In the original data set, art rating is given by a real number between -3 and 3. The last analysis section of this paper looks at reducing this space to a binomial distribution, a 0 for disliked ($rating \leq 0$) and a 1 for liked ($rating > 0$). To do so, we ran lasso regression on the transformed Y variable. The figure and table below show the search for optimal value for λ as well as the value of the beta coefficients in the model.

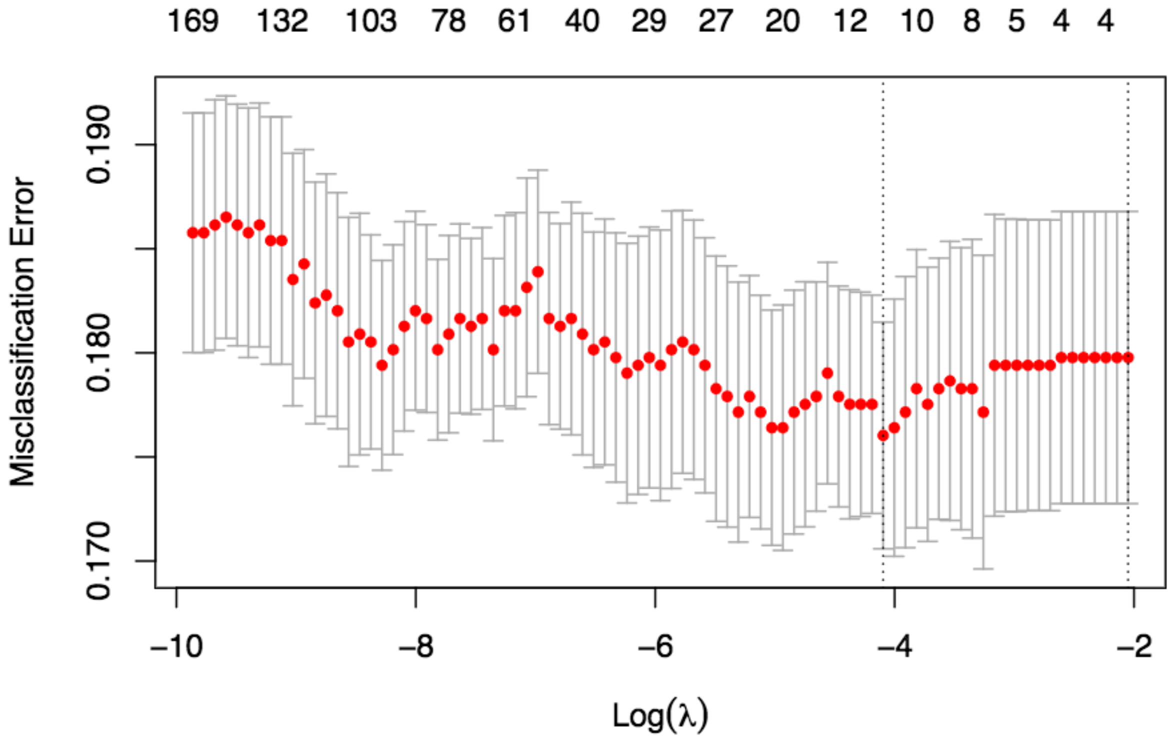


Figure 9: Cross validation plot to find ideal value of λ of logisitc regression model

Table 5: Coefficients included in logisitic model

	s0
(Intercept)	11.4246138
Year	-0.0047319
Year:postrenaissance	0.0002533
Year:median_red	-0.0000538
Year:median_green	-0.0000895
Year:Benses	-0.0016555
face:per_green	1.7764375
face:k_vals	0.2531022
contemporary:sd_green	-0.2603523
contemporary:k_edges	-0.2840705

To analyze the predictive capabilities of the model, we used a receiver-operator characteristic curve for analysis.

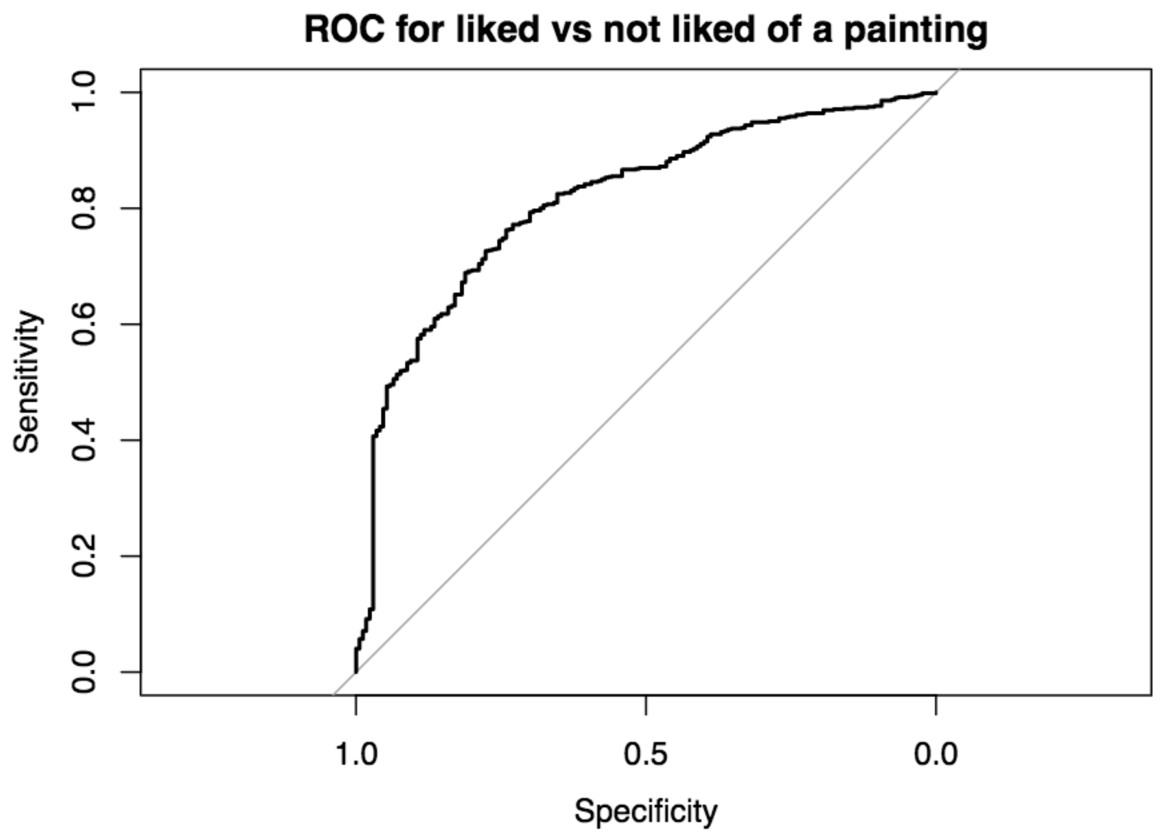


Figure 10: Receiver-operator chracteristic for the model to distinguish between art that is generally liked or generally disliked.

We used the area under the curve of a receiver operator charactertisic curve to rate the predictive capabilities of the logitic model. The 95% confidence interval of the area under the curve of the model is between 0.780371 and 0.8484852. This is generally considered to a good, but not great model for predictability. A discussion on thresholding is left out since there is no goal in the present paper for predicting art. For example, if an art collector was using this as a way to predict the rating of art and therefore it's selling price, you may want minimize false positives, artwork that is predicted to be liked, but in actuality is not, since you would be purchasing art with less of a chance of selling. Therefore you can set your threshold closer to 1, increasing your specificity.

Conclusion

In the preceding paper we developed 3 models to predict the art rating of paintings using ordinary linear regression, lasso regression, and logistic lasso regression. The models specified were created for the predictive capabilities, not for their actual modeling of the system. We combined metadata about the art, such as year painted, style of painting, and the presence or absence of a face, with image analysis data of the image, including percentage of the brightness of the image made up by each color channel, the percentage of the art that was considered an edge, as well as aesthetic measures that rate order over complexity in an image. These were compared to average art ratings taken during an onlice survey of more than 4,000 images of works of art.

The study found good predictive ability across the board, with the model found using lasso regression to be best for predicting the actual average art rating between -3 and 3. This may be due to the fact that we were able to perform regression on the full model as well as the interaction between the full model and itself. This was not possible for the OLS method using all possible regressions due to computation limitations.

Finally, we developed a logistic model to collapse the art ratings into two categories, liked and not liked. We found a fairly good model with a large AUC. However, we stopped short of refining the model threshold since there was no application in mind for the specificity and sensitivity of the model.

This shows that basic image analysis using aesthetic measure theory as well as simple data pertaining to works of art (in this case paintings) can inform predicitive models of the art rating of that art object. The reader should take caution when using this model though for the following reasons. First, the images studied were hosted on wikiart.org, and selected from the feature pages of each category of art hosted. The paintings were biased towards famous art, and art that has already been vetted by art critics and art historians to be of value. This is evident from that fact that much more of the art in the dataset was liked than disliked. To obtain a better model, one might add in amateur art, art from children, and other forms of painting to improve the predictive quality of the models. Second, the art studied was limited to western art, from renaissance period art to modern art. This model should not necessarily be used to predict the art rating of art from Asia or other time periods.

In conclusion, information theory inspired concepts of the aesthetic measure of art, coupled with easily reported metadata about art, can be used to make decent predictions about the average art rating of paintings.

References

1. G. D. Birkhoff, *Aesthetic Measure*, Harvard University Press, 1933.
2. V. Douchova, “Birkhoff’s Aesthetic Measure”, *Acta Universitatis Carolinae. Philisophica et historica*, pp.39-53, Aug. 2016.
3. J. Rigau, M. Feixas, M. Sbert, “Conceptualizing Birkhoff’s Aesthetic Measure Using Shannon Entropy and Kolmogorov Complexity.”, *Computational Aesthetics in Graphics, Visualization, and Imaging*, Jan. 2007.
4. S. M. Mohammad, “The Wikiarts Emotion Dataset”, May 2018.

Appendix

Image Scraping Python Code

```python
import urllib.request
import pandas as pd

df = pd.read_csv("WikiArt-Emotions-Ag4.csv")
image = urllib.request.URLopener()
for row in range(3448, len(df["Image URL"])):
    url_secure = df["Image URL"][row]
    filename = df["ID"][row]
    url = url_secure.replace("https", "http")
    print(url)
    image.retrieve(url, "{}.jpg".format(filename))
```

Art Analysis R script

```r
library(EBImage)
library(readr)
library(stringr)
library(R.utils)
library(CulturalAnalytics)

img_data = read.csv("gitrepos/Art_Analyzer/cleaned.csv", stringsAsFactors = FALSE)
img_data = subset(img_data, Is.painting=="yes")

mean_red = vector()
mean_green = vector()
mean_blue = vector()

median_red = vector()
median_green = vector()
median_blue = vector()

per_red = vector()
per_green = vector()
per_blue = vector()

sd_red = vector()
sd_green = vector()
sd_blue = vector()

per_edges = vector()
k_vals = vector()
k_edges = vector()

Benses = vector()

paths = vector()
IDs = vector()
```

```r
fhi = matrix(1, nrow = 3, ncol = 3)
fhi[2, 2] = -8

for(ID in img_data$ID){
  path = paste0("/Users/ericevje/gitrepos/Art_Analyzer/", ID, ".jpg")
  print(path)
  img = readImage(path)

  #Basic image analysis
  if(length(dim(img)) == 3){
    IDs = c(IDs, ID)

    mean_red = c(mean_red, mean(img[,,1]))
    mean_green = c(mean_green, mean(img[,,2]))
    mean_blue = c(mean_blue, mean(img[,,3]))

    median_red = c(median_red, median(img[,,1]))
    median_green = c(median_green, median(img[,,2]))
    median_blue = c(median_blue, median(img[,,3]))

    total_red = sum(img[,,1])
    total_green = sum(img[,,2])
    total_blue = sum(img[,,3])
    total = total_red + total_green + total_blue

    per_red = c(per_red, total_red / total)
    per_green = c(per_green, total_green / total)
    per_blue = c(per_blue, total_blue / total)

    sd_red = c(sd_red, sd(img[,,1]))
    sd_green = c(sd_green, sd(img[,,2]))
    sd_blue = c(sd_blue, sd(img[,,3]))

    grey = channel(img, "gray")
    img_fhi = filter2(grey, fhi)
    img_thresh = img_fhi > 0.25
    per_edge = sum(img_thresh)
    per_edge = per_edge/(per_edge + sum(img_thresh==FALSE))
    per_edges = c(per_edges, per_edge)

    #Compression algorithm analysis
    #Compute compressed size of image
    command = paste0("ls -l ", path, " | cut -d \" \" -f 7")
    size = system(command, intern=TRUE)
    if(size == ""){
      command = paste0("ls -l ", path, " | cut -d \" \" -f 8")
      size = system(command, intern=TRUE)
    }
    #print(size)
    #Kolmogorov calculation
    size = as.integer(size)
    max_info = 3 * sum(img_thresh==F | img_thresh==T)
    k_val = (max_info - size) / max_info
    print(k_val)
    k_vals = c(k_vals, k_val)

    #Compute compressed size of edge images
    path_thresh = paste0("/Users/ericevje/gitrepos/Art_Analyzer/", ID, "-thresh.jpg")
    writeImage(img_thresh, path_thresh)
    command = paste0("ls -l ", path_thresh, " | cut -d \" \" -f 7")
    size_con = system(command, intern=TRUE)
    if(size_con == ""){
      command = paste0("ls -l ", path_thresh, " | cut -d \" \" -f 8")
      size_con = system(command, intern=TRUE)
    }
    #Compute Kolmogrov Mk for edge image
    size_con = as.integer(size_con)
    contour_val = (max_info - size_con) / max_info
    print(contour_val)
    k_edges = c(k_edges, contour_val)

    #Compute Bense aesthetic measure based on shannon entropy
    entropy = imageEntropy(hist(grey))
    max_entropy = 8
    Bense = (max_entropy - entropy) / max_entropy
    print(Bense)
    Benses = c(Benses, Bense)

    #delete any images saved during analysis
    command = paste0("rm ", path_thresh)
    system(command)
  }
  else{
    paths = c(paths, path)
  }
}

results_df = data.frame(IDs, stringsAsFactors = F)
results_df$mean_red = mean_red
results_df$mean_green =  mean_green
results_df$mean_blue =  mean_blue

results_df$median_red = median_red
results_df$median_green = median_green
results_df$median_blue = median_blue

results_df$per_red = per_red
results_df$per_green = per_green
results_df$per_blue = per_blue

results_df$sd_red = sd_red
results_df$sd_green = sd_green
results_df$sd_blue = sd_blue

results_df$per_edges = per_edges
results_df$k_vals = k_vals
```

```r
results_df$k_edges = k_edges

results_df$Benses = Benses
```