Modeling Binary Social Medial Image Data

William F Lamberti

STAT 663

Dr. Carr

Abstract

We wish to correctly categorize two sets of image data from the social media service, Instagram. We utilized multivariable logistic regression model. We compared this model to alternative models that utilized techniques such as system vector machines and k-means. We compared the models utilizing appropriate methods such as ROC curves. We found that the logistic model was the best model, and it correctly categorized about 75% of the observations.

Introduction

The goal of this report is to correctly classify 2 sets of image data from the social media website, Instagram. In general, if a set of individual images, Q, either belong to set A or set B, we desire to accurately categorize each image of Q into its appropriate grouping of A or B. It could be possible to manually look at each image and correctly categorize them. For example, if we had 4 images of clouds and 6 images of trees, we could go through all 10 images and categorize them into 2 separate subsets of cloud images and tree images. However, we want to be able to have a computational statistical process to correctly categorize these images for us. This would enable another set of images that may have 400 cloud images and 600 tree images to be processed much quicker than by manually categorizing them. Being able to classify sets of images would not only be useful for social media websites to better understand their users' actions, but for a variety of other topics as well. For example, if a user could take a picture of an injury, upload it to a website for analysis, then the program could diagnose the injury instead of a doctor. This has broad social and medical applications. In places in the world where doctors are sparse, this type of solution could provide a cheaper alternative to a doctor. While this may remove the human component of a medical diagnosis, an important part of the medical process, it is still a viable option in many cases.

In our case, we will start the process of analyzing images in our simpler case. Each set of images has a unique tag that categorizes the images. The two tags were "fallleavesfall" and "winterbeachday". Each data set has 613 and 1171 observations in their populations respectively as of November 11, 2015. However, the total number of publically available images was 300 and 571 respectively. Thus, we analyzed 871 total images in our analysis. We obtained this data set by using a Firefox extension called "Save Images" by Lumox. We then converted each image

into its grayscale counterpart and obtained the histogram of the intensities with 256 variables. We performed this conversion with ImageJ, a powerful software program that can analyze images. We streamlined this process by writing a Javascript that would make the conversion faster than by hand. We then labeled all those images belonging to "fallleavesfall" with a 1, and all those images belonging to "winterbeachday" with a zero. Additionally, we assume that all the images are 8 bits. We do this because ImageJ must be told the number of bits in all the images.

Intuitively, we might picture two distinct images in our imaginations of what each tag might imply the image would include. For "fallleavesfall", we would imagine that their might be trees and leaves. The leaves would generally not be green, but they would be colors like red, orange, and yellow. There could also be browns for the trees themselves, or possibly the leaves as well. For "winterbeachday", we could imagine a significant amount was light tan for the color of the sand. Of course, sand does come in a variety of different colors, but light tan seems to be the most common. There would also be significant amount of blue or dark gray for the ocean water. A noticeable feature is that we would believe that very few people would be present on the beach itself. While it is certainly true that people will walk on the beaches during the winter or colder seasons, but it will be significantly less populated than images of the beach during the summer or warmers seasons. Nonetheless, we must try to image what the grayscale version would be like. This is certainly more difficult in comparison to imaging what colors would be present in for many people.

However, even if we believe that these fictional representation of these images might be true, the real data for these images had a significant amount of noise. For example, many of the images contained people in them. Therefore, there presence creates a similarity between the

images and will make the analysis much more difficult. Additionally, some of the images of those with the beach tag were during what an American would imagine as a typical beach image as there were people in swimwear. Additionally, the image quality and size varied. This is due to the fact that many of the users on Instagram upload their images from smart phones. Smart phones' cameras varies greatly in variables such as quality, clarity, and the way in which they are stored. Additionally, Instagram promotes the use of "filters" for the pictures. These filters can alter the images in about an infinite number of ways. Thus, the images are not original. Additionally, our data set contains the exact same image twice. It is possible that the user uploaded the image twice on accident. Thus, these are important factors to consider when performing this analysis.

Methods

We performed a variety of different statistical analyses on the data. The most successful was multivariable logistic regression. We also performed system vector machine (SVM) on the data as well, but it was not nearly as successful. We also explored other options such as Ward's clustering method and k-means, but none were nearly as successful as SVM. Thus, we will primarily limit the result of our analysis to multivariable logistic regression. We will also briefly discuss the results from SVM, k-means and Ward's clustering method.

We performed the analysis on the data by using R. The data itself was initially saved as JPEGs. It was then all converted to a .CSV file format with each image having 256 variables. We used Excel to confirm the format was correct as well as correctly stored.

For the logistic regression model, we utilized a partial least squares multiple regression fit to entire publicly available data. Dependent variable correlation and variance inflation were checked to ensure that no multicollinearity existed. This lead to the removal of 204 variables.

Following that, the validity of the model was checked by means of the log likelihood of the model and the X^2 value. While interactions were explored by means of the visual residual checks, only the first order model was utilized. Additionally, a Lasso method was utilized to try and further refine the model. However, the first λ utilized was extremely small. Thus, we deduced that keeping all the variables in the model was permissible. Lastly, the first order model's overall logistic regression model and confidence intervals were estimated using 95% confidence intervals. Thus, the final model resulted in a first order model with 51 regressors. Since there were no missing observations, no analysis on missing observations was required.

For the SVM, we split the data into testing and training data. The training data had 608 observations. In this approach, we retained all of the variables. The training data had similar proportions between the two groups in comparison to the original complete data set. We attempted various different kernels such as linear and polynomial with differing parameters. However, the best shape was a sigmoid. We further refined this by attempting different variables for the cost and coefficient variables. The best model was had a coefficient value of 0.5 and a cost value of 0.1. We then used the testing data to compare results. For k-means, we standardized the data and obtained the centroid of each cluster. For Ward's clustering method, we converted the data to Euclidean distances and then found the dissimilarity matrix. We then compare the percentage of correctly categorized to the logistic regression model.

We compared the logistic and SVM models by means of ROC curves. SVM was

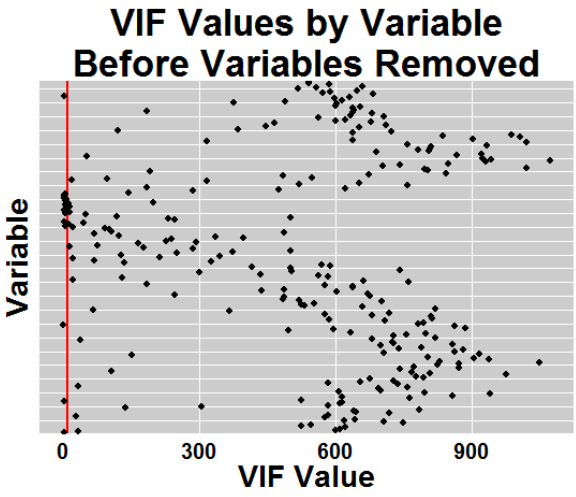

Figure 1: The VIFs of all the variables. The red line indicated the threshold value of 10 that the variables should be under.

significantly worse of in regards to both sensitivity and specificity for the training and testing data sets in comparison to the multiple logistic first order model. Thus, we believe the logistic model was the most appropriate approach.

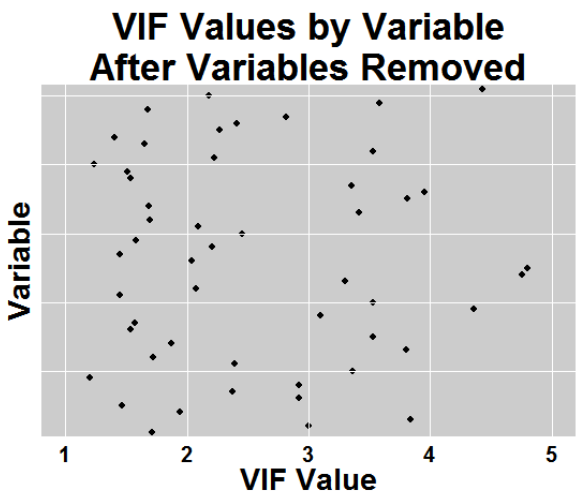

Figure 2: The VIFs of the remaining 51 variables. From the plot, we can see that all of them are below 5, which is more than adequate for the given threshold value.

Logistic Regression Results

Variance inflation factors (VIF) were compared to check for multiculinearity. The first VIF analysis was shown in Figure 1. The red line indicates the VIF value of 10. It is very clear, that many of the variables are significantly beyond that cutoff value. Initially, the variable with the highest VIF was "X47" with 1070.263. Thus, we begin by removing the variable with the highest VIF and continue this process until the maximum VIF is safely under 10. The final results with their presented values are presented in Figure 2. These 51 values are all below 5, so they are well under the general threshold value of 10.

With the remaining variables, we ran the first order model. The first order model's coefficients were summarized in Figure 3. The Log Likelihood value was -416.6125 with 52 degrees of freedom. The p value of the χ^2 was $2.407\text{X}10^{-34}$. Therefore, using this statistic, we have evidence that the model is significant. We then performed visual logistic regression diagnostics on the model. The residual plots can be seen in Figures 4 through 7. All figures have 868 unique covariate patterns. Since the number of covariate patterns is extremely close to the total 869 observations, this is a positive sign that the analysis can proceed. Figure 4 shows the leverage and estimate probability plot. Most of the points show are in a balanced position.

The black and white triangles indicate that those values are much higher than what the plot's limits indicate. Since the data is noisy, we find this plot permissible. Figure 5 shows the change in Chi-squared and the estimated probability plot. We note that there appears to be two quadratic shapes. Some of the observations fail to fit the model. However, since the data was noisy, we were not surprised that some of the observations failed to fit the model. Nonetheless, we retained those observations. Figure 6 shows the change in deviance and the estimated probability plot. Similar to the previous figure, we note that there are two quadratic shaped curves. Additionally, some points fall away from these curves. This also indicated that some of the points were not fit by the model very well. However, since the data is noisy, we are not surprised to have a few very large values and some that do not lay upon the curves. Thus, this plot is permissible. Figure 7 shows Cook's Distance and the estimated probability. The black and white triangle have similar representational meaning as to those in Figure 4. Those triangles indicate that those points have much higher values than indicated by the y axis. In particular, the last observation, which was also the last covariate pattern, had a leverage value of 400. One of the solutions we attempted was jackknifing the observation with the highest leverage one at a time until all observations had leverages below 0.4. However, despite jackknifing 39 times, we could not improve this plot. Consistently, we typically found that one covariate pattern always appeared to have at least a leverage value of 400. Thus, we decided to retain all the observations and covariate patterns.

Thus, we proceed with the summary analysis. To measure how well our model performed, we will utilize an ROC curve. We used this measure instead of a pseudo R^2 because we are most interested in correctly categorizing the 2 sets of images. The final model's ROC

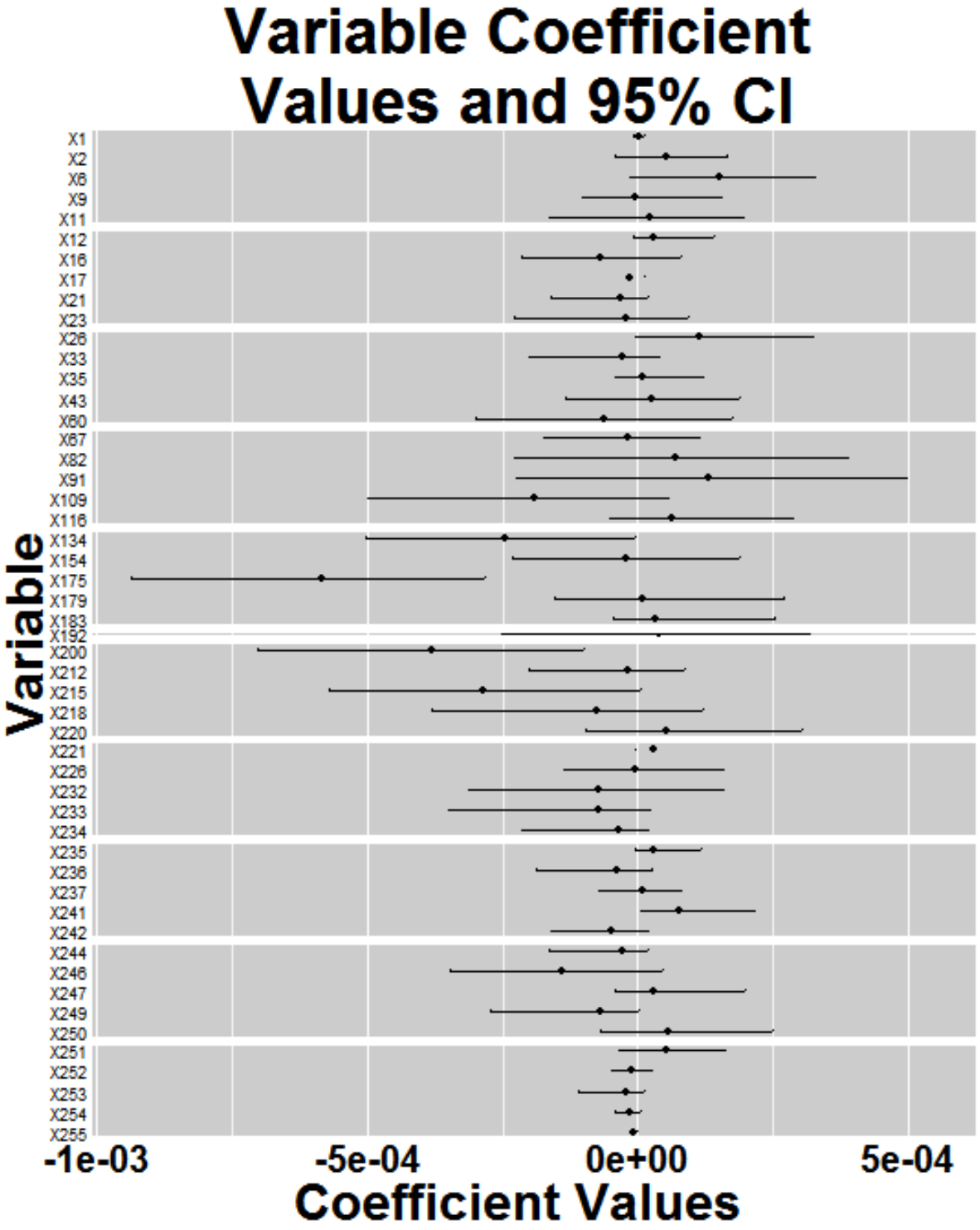

Figure 3: This plot shows the coefficient values and 95% confident intervals around those values

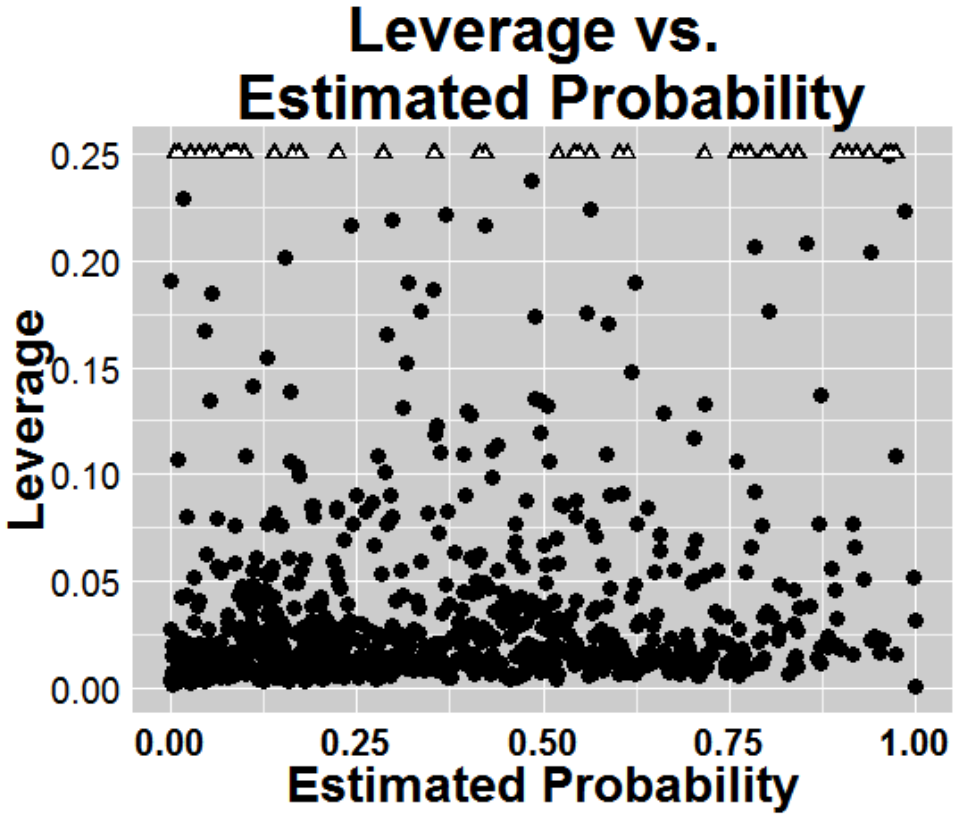

Figure 4: This plot shows 868 unique covariate patterns. The black and white triangles indicate that those values are much higher than what the plot's limits indicate.

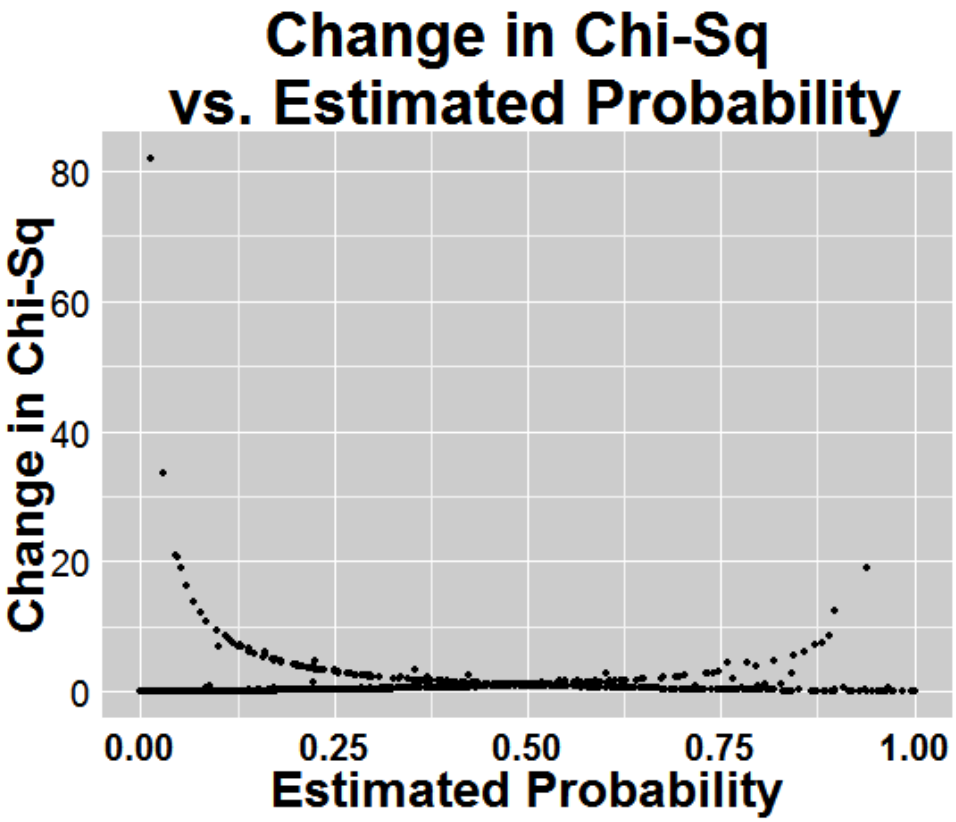

Figure 5: This plot has the change in Chi Squared and estimate probability. It appears to have two quadratic lines. There are a few poorly fit points. However, a majority of the points follow one of the two lines. This plot shows 868 unique covariate patterns.

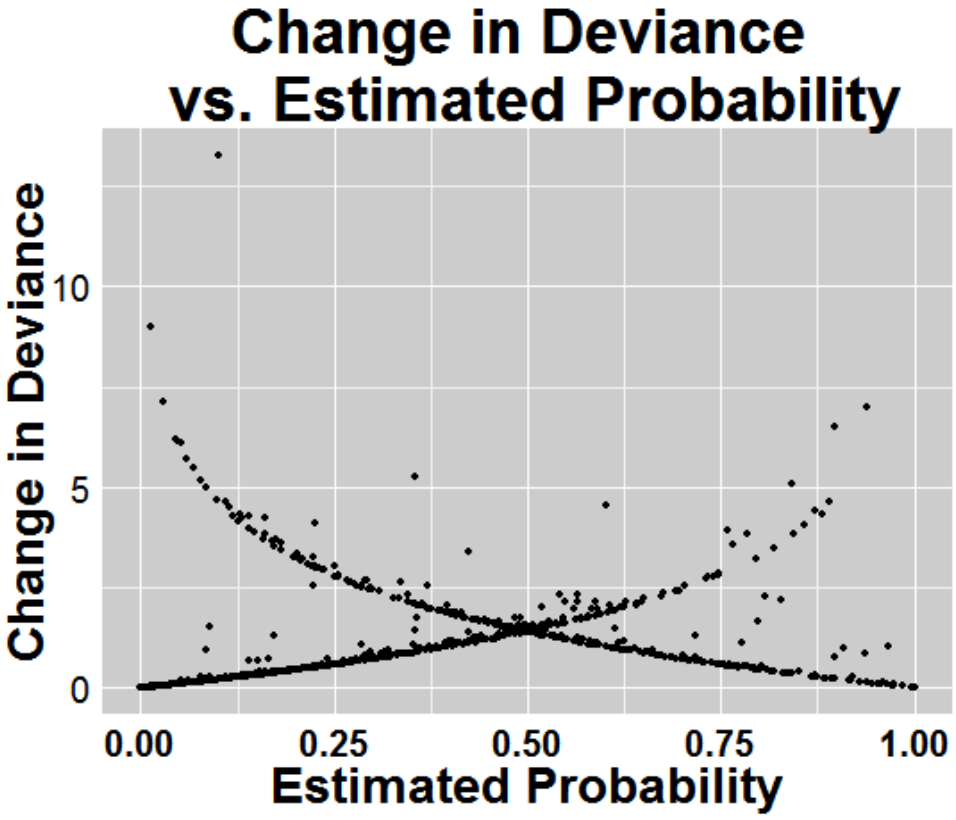

Figure 6: This plot shows the change in deviance and estimate probability. There appears to be two quadratic lines. While some points greatly deviate from these lines, most of the points follow these lines. This plot shows 868 unique covariate patterns.

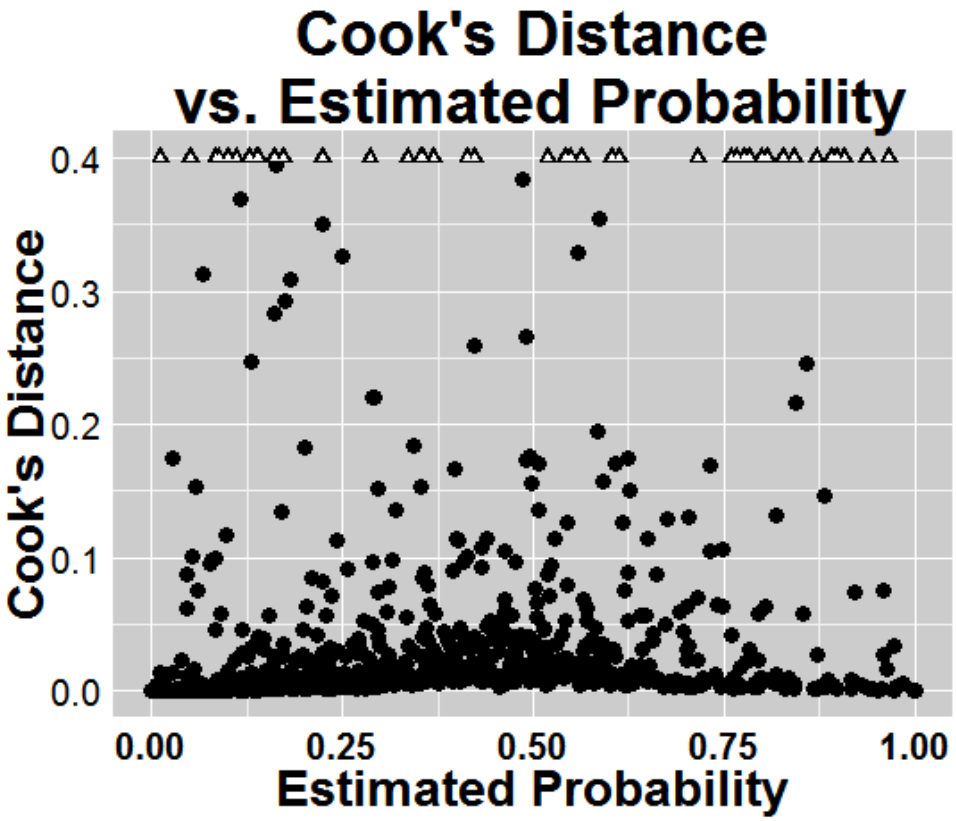

Figure 7: This plot shows Cook's Distance and estimated probability. The black and white triangles indicate that those values are much higher than what the plot's limits indicate. This plot shows 868 unique covariate patterns.

lines indicate our choice of the curve is displayed in Figure 8. The area underneath the curve (AUC) is .8255. We find this model to be a fairly excellent procedure for discriminating between the 2 sets of images. The red optimal balance between specificity and sensitivity. The values were .75 and .76 respectively. The white and blue dot indicates the point in which these 2 red lines intersect. We desire to find a balance point between the sensitivity and specificity

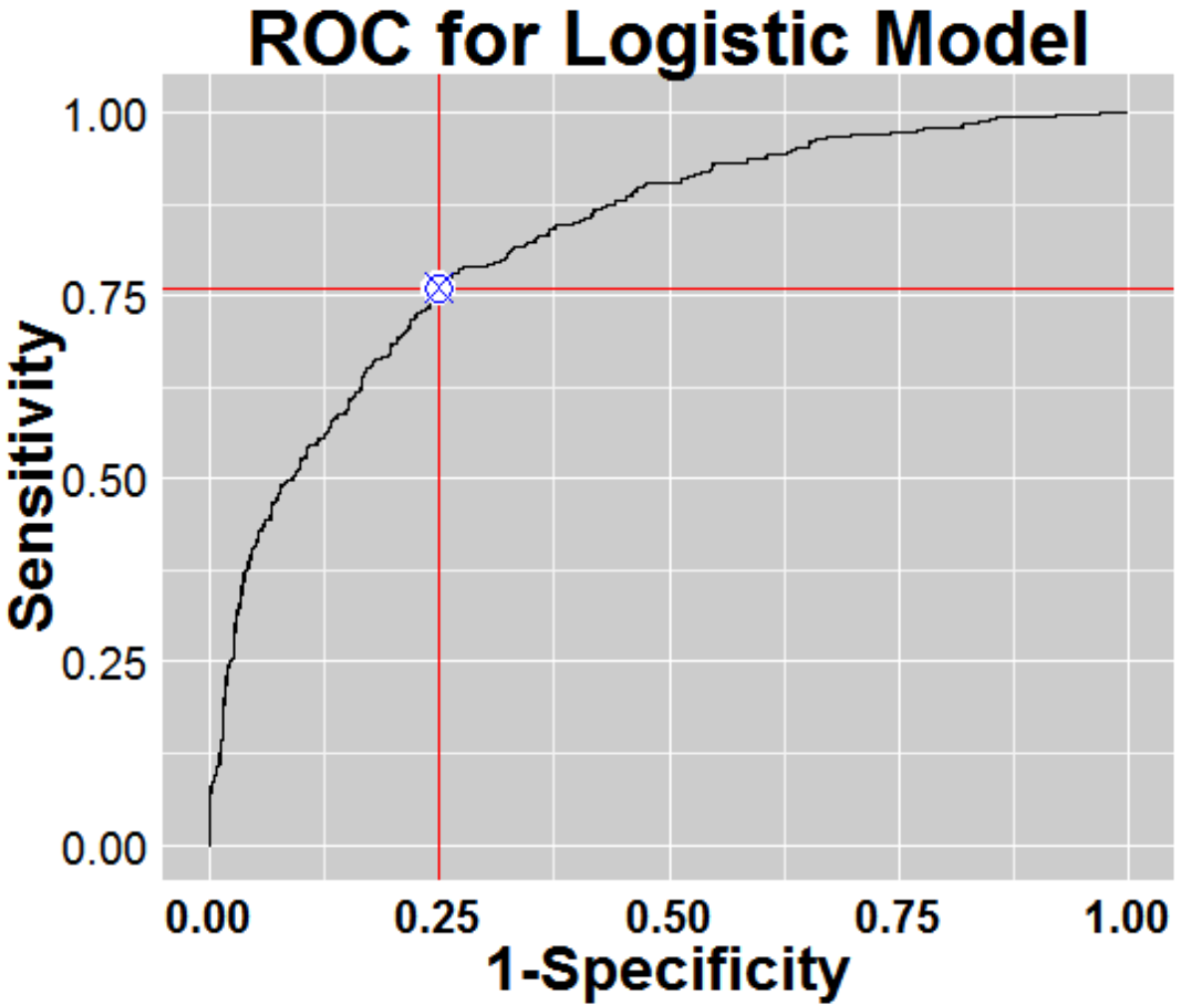

Figure 8: This plot shows the final logistic model's ROC curve. The red lines indicated our choice of the optimal balance point between Sensitivity and Specificity with values of .76 and .75 respectively. The white and blue point indicates the point at which the red lines intersect. The AUC is 0.8255. We find this to be a fairly excellent model for discrimination.

because correctly categorizing both are equally important. To obtain this balance between specificity and sensitivity, we need a threshold value of 0.364. If the estimated probability goes beyond this value, we will predict the observation to be a 1, or a picture belonging to the fall set. Otherwise, we will assign the observation a 0, or we will predict that it belong to the set of winter pictures. We summarize these results in Table 1. We note that about 75% of the observations were correctly classified. This provides further evidence that the model accurately predicted the observations.

Table 1: Using a Threshold of 0.364, this table has Predicted vs. Actual Groups Counts.

Percent Correct: 75%	Predicted 0	Predicted 1
Actual 0	431	139
Actual 1	75	224

SVM Results:

We then performed system vector machines to try and split the data into two separable groups using all 256 variables. We first split the data into a training and test data sets. The training data set had 608 observations while the test data set had 261 observations. About 64.7% of the observations came from the winter images while about 35.3% of the observations came from the fall set of images. A variety of kernels and parameters were selected, attempting to find the best model possible. The best model utilized a sigmoid kernel with parameters 0.1 and 0.005 for the cost and coefficient respectively. By using a 10 fold cross validation method, this model had the lowest error of 0.255. However, the SVM did utilize 430 support vectors. We recognize this is a large number of support vectors. The ROC curve for the training and test data set is provided below in Figure 9. We indicated the ROC curve for the training data in red and the for the test data in purple. While the test data does general perform better than the testing data, both

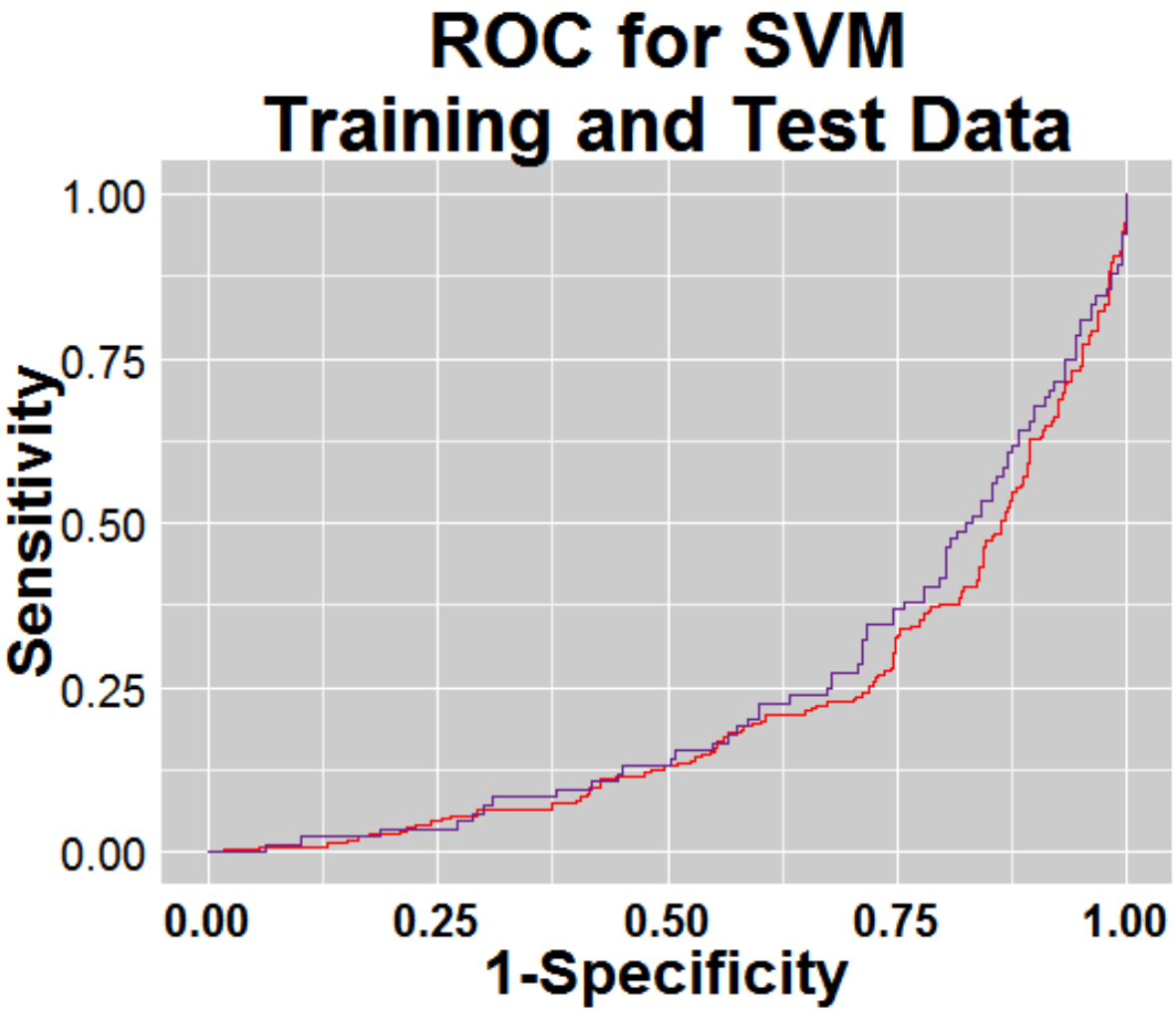

Figure 9: This is the ROC plot for the SVM model using the training and test data. The colors for each are red and purple, respectively. The AUC for each curve is less than 0.50.

ROC curves have AUC's that are less than 0.50. Thus, we recognize that this model fails to discriminate between the two data sets well.

k-means Results and Comparison:

We then performed a k-means analysis on the data. We first tried to determine how many clusters were present in the data. Using the plot in Figure 10, we compared the number of clusters vs the within groups sum of squares. The plot seems to suggest that the optimal number of clusters is two as the slope between the points with the number of cluster values of 1 and 2

had the largest slope. This appears promising as the data has two groups. We standardized the data, and obtained the cluster centroids. We then aggregated the data and converted it back into its original metrics for comparison. Additionally, we manually calculated in R all the numbers of clusters from 2 to 8. Again, 2 clusters had the best partition of the data. Table 1 shows the comparison between the clusters created by k-means and the actual groups created. The table shows that a majority of observations were combined into the first k-means cluster despite their

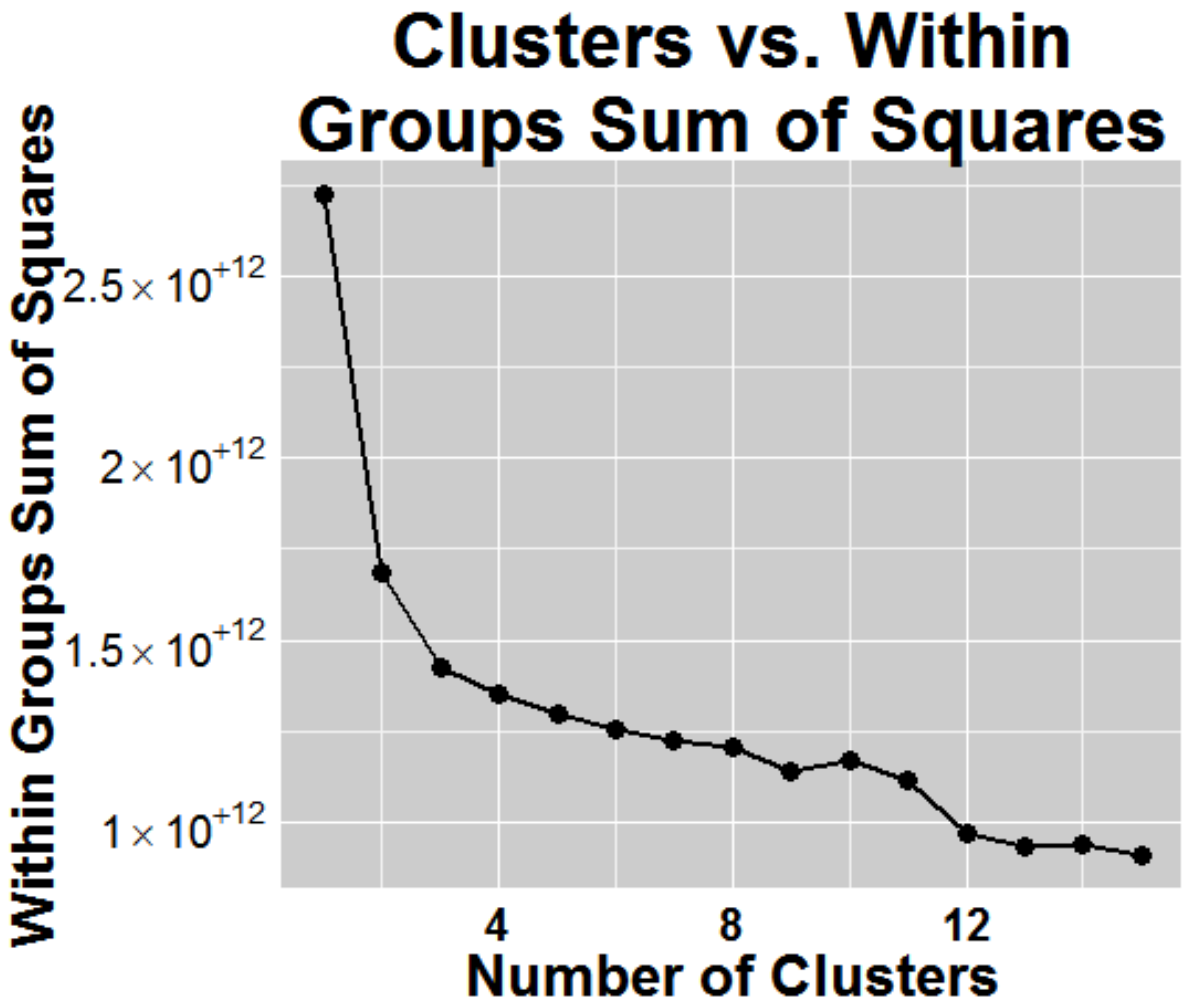

Figure 10: This plot has the numbers of cluster and the within groups sum of squares. We decide that 2 clusters was appropriate since that has the largest drop.

Table 1: Comparing k-means Clusters to Actual Groups		
Cluster/Group	**k-means:1**	**k-means:2**

original group. What we desired to observe is a large number from the fall pictures in one of the k-means cluster, and the second winter pictures to be discriminated into the other k-means group.

However, this was not the case. Due to the large error of the method, we quickly recognize that this is an inferior partitioning method than in comparison to the logistic regression model. While we do not know which cluster created was intended for which actual group, if we assume, just for comparison, that the cluster allocation with the smaller error rate was correct, then the percent correctly allocated was about 65%. Thus, since the percent correctly allocated for the logistic regression model was higher with 75%, we can determine that the logistic model did significantly better.

Table 1: Comparing k-means Clusters to Actual Groups

Cluster/Group	k-means:1	k-means:2
Actual:1	525	45
Actual:2	285	14

Ward's Clustering Methods and Comparison:

We then attempted Ward's Method to try group the observations into clusters. We converted that data into Euclidean distances and found their dissimilarities. The dendrogram is shown in Figure 10. The two red boxes shows the two clusters formed. We note that the percentage of correctly classified images was about 39%. However, as we noted previously, the logistic regression model correctly classifies about 75% of the observations. Thus, we conclude that Ward's method did poorer than the logistic regression model.

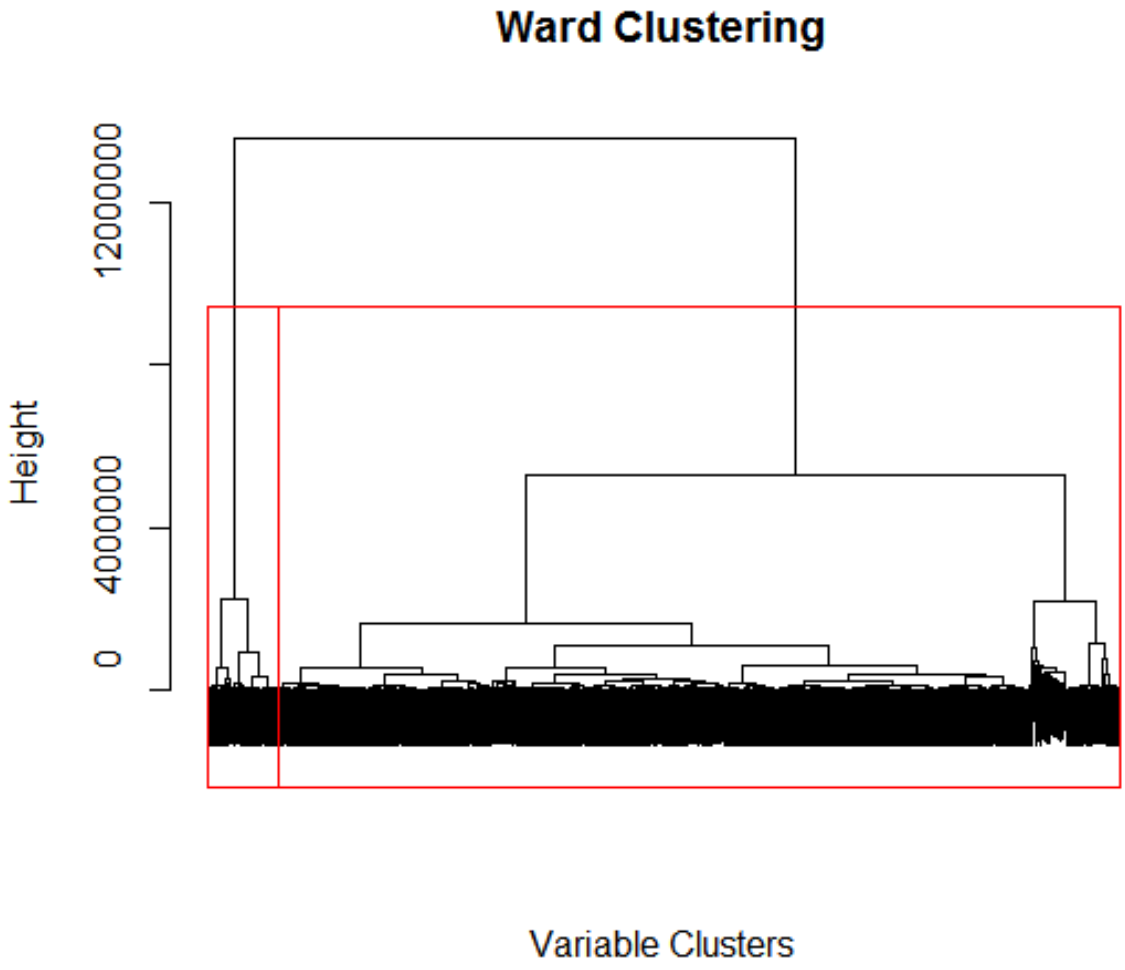

Figure 10: This plot is the dendrogram of the Ward's method of cluster analysis. The 2 red boxes indicate where we cut the dendrogram and created the 2 clusters.

Logistic Regression and SVM Comparison:

We now compare the two most successive models that we have performed thus far. We will compare all logistic model and SVM model results using ROC curves. We used this method for comparison because the goal of the analysis is to correctly categorize the two data sets of images. We provide the ROC plot in Figure 12. We present the logistic model's ROC curve in green. The SVM's ROC curve for the training and testing data sets are in red and purple

Table 2: The AUC for each of the models.

Model	AUC
Logistic	0.8255
SVM	>0.500

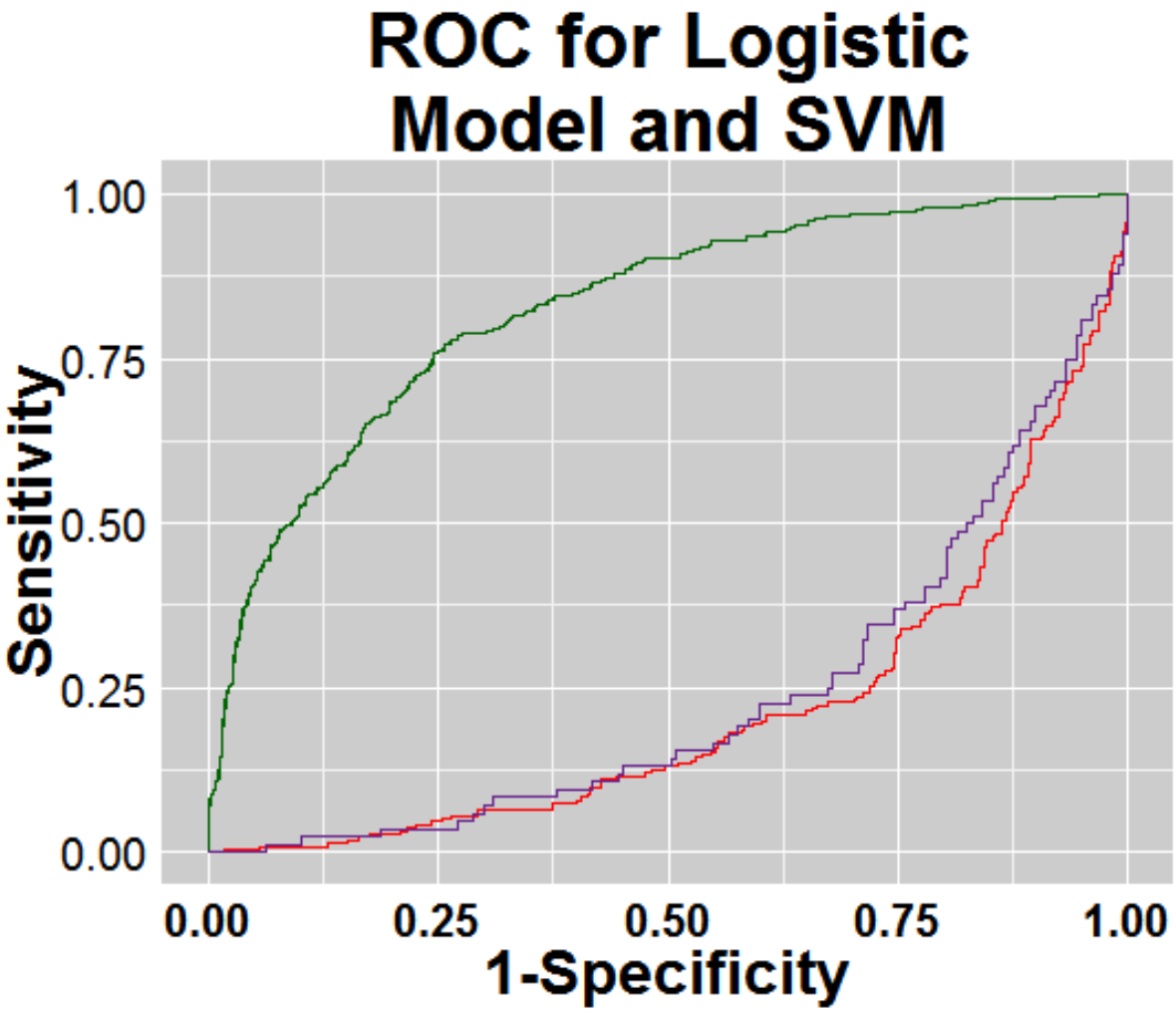

Figure 12: This plot compares all the ROC curves of the logistic and SVM models. The logistic model's curve is in green. The SVM's curves are in red and purple for the training and test data respectively.

respectively. It is quickly apparent without comparing the AUC's that the logistic model performs the best. It helps to discriminate the data well. As stated previously, the AUC for the logistic model was 0.8255. The AUC for the SVM model was less than 0.500 for the training and testing data. These results are summarized in Table 2.

Discussion:

In regards to the data analyzed, we need to correct confirm that each image was 8 bits. Since ImageJ assumes that the image is 8 bits, this assumption could easily be violated in social

media images due to the great variation in smart phone camera quality. Additionally, this study could be further improved by means of getting the distributions for each of the colors, green, red, and blue. This would give us more information about each image. When we convert to the gray scales, we lose a lot of information about each image. Having each distribution on color would allow us to make comparisons for differences in colors between these data sets. For instance, we believe that images with orange would be more prevalent than those in winter images. Having three variables to describe a single color would be more helpful than simply a gray color. However, this method would require a larger data set for using multiple logistic regression. Additionally, it would introduce correlated variables, another complication.

Additionally, after the analyses were completed, additionally resources suggested that standardizing the data would help to improve the performance of SVM. Thus, we would like to consider this approach at a future time. We would also like to see if attempts of using a Bayesian logistic model would provide an improved model. An additional variable that would desire to measure are order statistics. The order in which the colors or intensities appear on the image can make a big difference for the type of image. For example, if we were trying to distinguish images of a forest that had a lake and did not, we hypothesize that the most important distinguishing feature is the absence of the lake itself. In general, we might believe that the lake would have blueish hue with perhaps a mixture of other colors from reflections. If the images all had the sky on the top half or third of the image, we would generally believe that the lake would be in the middle to the bottom of the photograph. Thus, the order statistics for when specific colors arrive matters. Obviously, this method might still have a hard time distinguishing images when a swamp is involved as the swamp can also be green and brown, a common color for trees. However, we do believe that this variable will help to further distinguish sets of images.

However, we have a significant amount of interest if this approach is generalizable. We hypothesize that if we remove those variables with high VIF, then the remaining first order model will do fairly well at categorizing the images. However, we recognize the complexity in this kind of hypothesis as we only performed this analysis on social media data. For instance, we did not use this approach on fMRI data. We do not know how this approach will work. However, we would like to explore if this approach does well. Nonetheless, we also recognize the nature of the image data that we have. Social media data can be vastly different from image data ascertained via satellites from NASA. Thus, we hope to find statistical methods that can generalize from different types of images.

Appendix

```
#cleaning data

library('reshape')

fall<-cast(fall, Label~Bin)

saveRDS(fall, 'fall.rds')
save(fall, file="fall_ver2.RData")
write.csv(fall, file = "fall_reshape.csv")


winter<-cast(winter, Label~Bin)

saveRDS(winter, 'winter.rds')
save(winter, file="winter_ver2.RData")
write.csv(winter, file = "winter_reshape.csv")

#--------------------------------#

one<-c(rep.int(1,299))
zero<-c(rep.int(0,570))

winter<-data.frame(winter)
fall<-data.frame(fall)

fall<-cbind(fall, one)
winter<-cbind(winter, zero)

colnames(fall)[257]<- 'group'
colnames(winter)[257]<- 'group'

finaldata<-rbind(fall, winter)

#logistic regression ------------------------------------------------------
-----------------------------
library("zoo")
library("mvtnorm")

library('LogisticDx')
library('ggplot2')
library('arm')

load('finaldata.RData')
labels<-finaldata$Label
finaldata$Label<-NULL


groups<-finaldata[,256]


g1 <- glm(group ~. , family =binomial,  data =finaldata)
```

```
library('car')
vif1<-vif(g1) #remove X47
NAME<-names(vif1)
Value<-unname(vif1)


vif1<-cbind(Value,NAME)
vif1

classes <- paste("G",51:1,sep="")
classes
reps<-rep.int(5,51)
vif1<-as.data.frame(vif1)

vif1$Grp<-factor(rep(classes,reps),level=classes)

VAL1<-as.numeric(as.character(vif1$vif1))

#way too many that they don't fit...lets just do a simple plot with a
horizontal line instead
p <-
(ggplot(vif1, aes(x=Value,
    y=NAME,group=Grp))
  + labs(x="VIF Value", y="Variable",
      title="VIF Values by Varibale")
  + geom_point(shape=21,size=I(2.8))
  + facet_grid(Grp ~ ., scale="free_y", space="free" )
  + theme(axis.text.y=element_text(size=rel(.78)))
)
p

max(vif(g1))
which.max(vif(g1))


mytheme<-theme(

      plot.title = element_text(size=35, face="bold"),
      axis.text.x  = element_text(size=20, face="bold"),
      axis.text.y=element_blank(),
      axis.title.x=element_text(size=28, face='bold'),
      axis.title.y=element_text(size=28, face='bold'),
      panel.background=element_rect(fill="gray80"),
      axis.ticks= element_blank(),
      panel.grid.major=element_blank(),
      panel.grid.minor=element_blank(),
      axis.text=element_text(colour="black")

      )
vif1<-vif(g1) #remove X47
NAME1<-names(vif1)
Value1<-unname(vif1)
```

```
vif1<-cbind(Value1,NAME3)
vif1
vif1<-as.data.frame(vif1)
VAL1<-as.numeric(as.character(Value1))
v<-
(ggplot(NULL, aes(x=VAL1,
    y=NAME1))
  + labs(x="VIF Value", y="Variable",
      title="VIF Values by Variable\nBefore Variables Removed")
  + geom_point(size=3)
  +mytheme
   +geom_hline(yintercept = 10, colour = "white", size=.7) + #change these
as you can't tell what the months are
                      geom_hline(yintercept = 20, colour = "white",
size=.7) +
                      geom_hline(yintercept = 30, colour = "white",
size=.7) +
                      geom_hline(yintercept = 40, colour = "white",
size=.7) +
                      geom_hline(yintercept = 50, colour = "white",
size=.7) +
                      geom_hline(yintercept = 60, colour = "white",
size=.7) +
                      geom_hline(yintercept = 70, colour = "white",
size=.7) +
                      geom_hline(yintercept = 80, colour = "white",
size=.7) +
                      geom_hline(yintercept = 90, colour = "white",
size=.7) +
                      geom_hline(yintercept = 100, colour = "white",
size=.7) +
                      geom_hline(yintercept = 110, colour = "white",
size=.7) +
                      geom_hline(yintercept = 120, colour = "white",
size=.7) +
                    geom_hline(yintercept = 130, colour = "white",
size=.7) +
                    geom_hline(yintercept = 140, colour = "white",
size=.7) +
                    geom_hline(yintercept = 150, colour = "white",
size=.7) +
                    geom_hline(yintercept = 160, colour = "white",
size=.7) +
                    geom_hline(yintercept = 170, colour = "white",
size=.7) +
                    geom_hline(yintercept = 180, colour = "white",
size=.7) +
                    geom_hline(yintercept = 190, colour = "white",
size=.7) +
                    geom_hline(yintercept = 200, colour = "white",
size=.7) +
                    geom_hline(yintercept = 210, colour = "white",
size=.7) +
```

```
                      geom_hline(yintercept = 220, colour = "white",
size=.7) +
                      geom_hline(yintercept = 230, colour = "white",
size=.7) +
                      geom_hline(yintercept = 240, colour = "white",
size=.7) +
                      geom_hline(yintercept = 250, colour = "white",
size=.7)+

                      geom_vline(xintercept = 300, colour = "white",
size=.7) +
                      geom_vline(xintercept = 600, colour = "white",
size=.7) +
                      geom_vline(xintercept = 900, colour = "white",
size=.7) +

                      geom_vline(xintercept = 10, colour = "red", size=1)

  )

v<- v+ geom_point(data=NULL, aes(x=VAL1, y=NAME1), size=3)


#UPDATING MODEL BY REMOVING THOSE WITH VIF HIGHER THAN 10

g2<- glm(group ~. -X47-X145-X59-X42-X49-X62-X137-X124-X148-X169-X182-X141-
X51-X55-X53
                        -X152-X64-X184-X176-X150-X163-X121-X161-X133-
X174-X159-X157-X135
                        -X76-X45-X30-X129-X167-X57-X32-X155-X198-X127-
X114-X146-X106-X130
                        -X171-X139-X78-X40-X112-X70-X143-X165-X125-X108-
X119-X118-X61-X101
                        -X74-X66-X72-X97-X84-X104-X48-X110-X189-X86-X79-
X89-X94-X186-X209
                        -X153-X98-X69-X196-X123-X178-X81-X181-X102-X173-
X87-X205-X100-X194
                        -X37-X92-X207-X201-X132-X105-X95-X191-X24-X7-
X140-X77-X166-X54-X117
                        -X136-X83-X44-X149-X187-X93-X160-X156-X28-X217-
X41-X193-X111-X199-X230
                        -X19-X206-X225-X58-X90-X122-X85-X185-X142-X50-
X71-X223-X115-X36-X170
                        -X128-X177-X214-X4-X5-X216-X202-X239-X107-X147-
X131-X27-X164-X15-X99
                        -X56-X210-X80-X180-X158-X73-X113-X190-X22-X172-
X227-X222-X197-X204-X38
                        -X103-X138-X240-X65-X68-X13-X228-X46-X120-X195-
X88-X31-X25-X151-X10-X224
                        -X213-X219-X63-X20-X96-X75-X14-X203-X245
, family =binomial,  data =finaldata) #simpilying by just removing and
updating this model

max(vif(g2))
```

```
which.max(vif(g2))

#now ensuring well below 5


g3<- glm(group ~. -X47-X145-X59-X42-X49-X62-X137-X124-X148-X169-X182-X141-
X51-X55-X53
                          -X152-X64-X184-X176-X150-X163-X121-X161-X133-
X174-X159-X157-X135
                          -X76-X45-X30-X129-X167-X57-X32-X155-X198-X127-
X114-X146-X106-X130
                          -X171-X139-X78-X40-X112-X70-X143-X165-X125-X108-
X119-X118-X61-X101
                          -X74-X66-X72-X97-X84-X104-X48-X110-X189-X86-X79-
X89-X94-X186-X209
                          -X153-X98-X69-X196-X123-X178-X81-X181-X102-X173-
X87-X205-X100-X194
                          -X37-X92-X207-X201-X132-X105-X95-X191-X24-X7-
X140-X77-X166-X54-X117
                          -X136-X83-X44-X149-X187-X93-X160-X156-X28-X217-
X41-X193-X111-X199-X230
                          -X19-X206-X225-X58-X90-X122-X85-X185-X142-X50-
X71-X223-X115-X36-X170
                          -X128-X177-X214-X4-X5-X216-X202-X239-X107-X147-
X131-X27-X164-X15-X99
                          -X56-X210-X80-X180-X158-X73-X113-X190-X22-X172-
X227-X222-X197-X204-X38
                          -X103-X138-X240-X65-X68-X13-X228-X46-X120-X195-
X88-X31-X25-X151-X10-X224
                          -X213-X219-X63-X20-X96-X75-X14-X203-X245-X231-
X34-X8-X3-X188-X248-X168
                          -X126-X144-X29-X229-X18-X52-X208-X211-X39-X243-
X238-X162
, family =binomial,  data =finaldata) #simpilying by just removing and
updating this model

max(vif(g3))
which.max(vif(g3))


vif3<-vif(g3) #remove X47
NAME3<-names(vif3)
Value3<-unname(vif3)


vif3<-cbind(Value3,NAME3)
vif3
vif3<-as.data.frame(vif3)
VAL3<-as.numeric(as.character(vif3$Value3))
v3<-
(ggplot(vif3, aes(x=VAL3,
    y=NAME3))
  + labs(x="VIF Value", y="Variable",
      title="VIF Values by Variable\nAfter Variables Removed")
```

```
  + geom_point(size=3)
  +mytheme
  +geom_hline(yintercept = 10, colour = "white", size=.7) + #change these
as you can't tell what the months are
                      geom_hline(yintercept = 20, colour = "white",
size=.7) +
                      geom_hline(yintercept = 30, colour = "white",
size=.7) +
                      geom_hline(yintercept = 40, colour = "white",
size=.7) +
                      geom_hline(yintercept = 50, colour = "white",
size=.7) +


                      geom_vline(xintercept = 1, colour = "white",
size=.7) +
                      geom_vline(xintercept = 2, colour = "white",
size=.7) +
                      geom_vline(xintercept = 3, colour = "white",
size=.7) +
                      geom_vline(xintercept = 4, colour = "white",
size=.7) +
                      geom_vline(xintercept = 5, colour = "white",
size=.7))



#validate model

logLik(g3)

#chi-squared pvalue
with(g3, pchisq(null.deviance - deviance, df.null - df.residual,
lower.tail = FALSE))




#ROC CURVE
library('pROC')

#obtaining fitted values

prob <- predict(g3, type = "response")
#finaldata$prob=prob

g <- roc(group ~ prob, data = finaldata)
plot(g)


#performing Lasso

library(glemnet)
```

```
x = model.matrix(group ~. -X47-X145-X59-X42-X49-X62-X137-X124-X148-X169-
X182-X141-X51-X55-X53
                         -X152-X64-X184-X176-X150-X163-X121-X161-X133-
X174-X159-X157-X135
                         -X76-X45-X30-X129-X167-X57-X32-X155-X198-X127-
X114-X146-X106-X130
                         -X171-X139-X78-X40-X112-X70-X143-X165-X125-X108-
X119-X118-X61-X101
                         -X74-X66-X72-X97-X84-X104-X48-X110-X189-X86-X79-
X89-X94-X186-X209
                         -X153-X98-X69-X196-X123-X178-X81-X181-X102-X173-
X87-X205-X100-X194
                         -X37-X92-X207-X201-X132-X105-X95-X191-X24-X7-
X140-X77-X166-X54-X117
                         -X136-X83-X44-X149-X187-X93-X160-X156-X28-X217-
X41-X193-X111-X199-X230
                         -X19-X206-X225-X58-X90-X122-X85-X185-X142-X50-
X71-X223-X115-X36-X170
                         -X128-X177-X214-X4-X5-X216-X202-X239-X107-X147-
X131-X27-X164-X15-X99
                         -X56-X210-X80-X180-X158-X73-X113-X190-X22-X172-
X227-X222-X197-X204-X38
                         -X103-X138-X240-X65-X68-X13-X228-X46-X120-X195-
X88-X31-X25-X151-X10-X224
                         -X213-X219-X63-X20-X96-X75-X14-X203-X245-X231-
X34-X8-X3-X188-X248-X168
                         -X126-X144-X29-X229-X18-X52-X208-X211-X39-X243-
X238-X162, family =binomial,  data =finaldata)
y = finaldata$group


lasso.mod=glmnet(x, y,alpha=1)
plot(lasso.mod) #but lambdas so small that it isn't worth taking any out


final.model.data<-finaldata
final.model.data<-final.model.data[,-
c(47,145,59,42,49,62,137,124,148,169,182,141,51,55,53

,152,64,184,176,150,163,121,161,133,174,159,157,135

,76,45,30,129,167,57,32,155,198,127,114,146,106,130

,171,139,78,40,112,70,143,165,125,108,119,118,61,101

,74,66,72,97,84,104,48,110,189,86,79,89,94,186,209

,153,98,69,196,123,178,81,181,102,173,87,205,100,194

,37,92,207,201,132,105,95,191,24,7,140,77,166,54,117

,136,83,44,149,187,93,160,156,28,217,41,193,111,199,230

,19,206,225,58,90,122,85,185,142,50,71,223,115,36,170
```

```
,128,177,214,4,5,216,202,239,107,147,131,27,164,15,99

,56,210,80,180,158,73,113,190,22,172,227,222,197,204,38

,103,138,240,65,68,13,228,46,120,195,88,31,25,151,10,224

,213,219,63,20,96,75,14,203,245,231,34,8,3,188,248,168
                          ,126,144,29,229,18,52,208,211,39,243,238,162)]



#renaming g3 using simplier dataframe to improve speed
g3<- glm(group ~., family =binomial,  data =final.model.data)

summary(g3)

#creating plot

library(ggplot2)

mytheme.ci<-theme(

      plot.title = element_text(size=35, face="bold"),
      axis.text.x  = element_text(size=20, face="bold"),
      axis.text.y=element_text(size=20),
      axis.title.x=element_text(size=28, face='bold'),
      axis.title.y=element_text(size=28, face='bold'),
      strip.background=element_rect(fill="gray80"),
      panel.background=element_rect(fill="gray80"),
      axis.ticks= element_blank(),
      axis.ticks.margin=unit(-0.05,"cm"),
      axis.text=element_text(colour="black"),
        panel.grid.minor.y = element_blank(),
  panel.grid.major.y = element_blank()


      )


coeff.model.1<-coef(g3)
coeff.names<-names(coeff.model.1)

#remove intercept
coeff.model<-coeff.model.1[-1]

coeff.model<-as.data.frame(coeff.model)

classes <- paste("G",11:1,sep="")
classes
reps <- c(5,5,5,5,5,1,5,5,5,5,5)
coeff.model$Grp <- factor(rep(classes,reps),level=classes)

# Create a factor for rows within groups
```

```
top <- rep(1:5,5)
top
subs <- c(top,1,top)
subs
labs <- c('1', '2',
          '3', '4', '5')
tmp <- labs[subs]
coeff.model$Row <- factor(tmp,levels=labs)

coeff.model$Index<- c(1:51)
coeff.model$c.names<-coeff.names[-1]

g3.CI<-confint(g3)
g3.CI<-as.data.frame(g3.CI)
g3.CI<- g3.CI[-1,]

nine<-g3.CI[,2]
two<-g3.CI[,1]

coeff.model$two<-two
coeff.model$nine<-nine

#coeff.model$stand.error<- - coeff.model$coeff.model
#coeff.model$st.er.95<-stand.error*1.646609


library(grid)
library(scales)

c <-ggplot(coeff.model, aes(x=coeff.model, y=Index ,fill=Row, group=Grp))
+geom_point()+
               xlab("Coefficient Values")+
               ylab("Variable")+
               ggtitle("Variable Coefficient\nValues and 95% CI")+
               facet_grid(Grp ~ ., scale="free_y", space="free" )+
               theme(axis.text.y=element_text(size=rel(.78)))+
               mytheme.ci+
               theme(legend.position="none")+
               theme(axis.text.y=element_text(size=rel(.9)))+
               scale_y_continuous(trans = "reverse", breaks =
unique(coeff.model$Index), labels=rownames(coeff.model))


c<-c+ geom_errorbarh(aes(xmax = nine, xmin = two, height=.1))

#performing residual anlaysis of each variable

model.res<-resid(g3)

plot(finaldata$X1, model.res)


res.data<-final.model.data
res.data$res<-model.res
```

```
res.data$group<-NULL

obs<-c(1:869)

#reforming data

library(reshape2)

melting<-melt(res.data, id.vars="res", value.name="count")

head(melting)

#Creating plots
library(ggplot2)
library(grid)

p<-qplot(data=melting, y=res, x=count)

p<-p+facet_wrap(~variable)

p<-p+ theme(panel.margin = unit(5, "cm"))



#diagonostics plots and getting covariates

dig<-dx(g3)

library(epiR)

mf.g3 <- model.frame(g3)
cp.g3 <- epi.cp(mf.g3[-1])
cov.num<-length(cp.g3$cov.pattern[,2])


#reforming data

library(reshape2)

melting<-melt(res.data, id.vars="res", value.name="count")

head(melting)

#Creating plots
library(ggplot2)
library(grid)



#diagonostics plots and getting covariates

dig<-dx(g3)
```

```
#-------------------------------------------------------------------------
-plots-------------


mytheme.roc<-theme(

      plot.title = element_text(size=35, face="bold"),
      axis.text.x  = element_text(size=20, face="bold"),
      axis.text.y=element_text(size=20),
      axis.title.x=element_text(size=28, face='bold'),
      axis.title.y=element_text(size=28, face='bold'),
      strip.background=element_rect(fill="gray80"),
      panel.background=element_rect(fill="gray80"),
      axis.ticks= element_blank(),
      axis.text=element_text(colour="black")

      #plot.margin = unit(c(27,19,27,19), 'mm')

      )


#h vs est. prob


lev<-ggplot(data=dig, aes(x=P, y=h))+ geom_point()+mytheme.roc+
               xlab("Estimated Probability")+
               ylab("Leverage")+
               ggtitle("Leverage vs. \n Estimated Probability")
lev

h.nums<-which(dig$h>.25)
h2<-dig$h
h2[h.nums]<-.25
p2<-dig$P
hf<-c(rep(0,868))
hf[h.nums]<-1
v1.2<-dig$P
v1.2[-h.nums]<-NA
h2.2<-h2
h2.2[-h.nums]<-NA


h.mat<-as.data.frame(cbind(dig$P, h2))
h.mat$hf.f<-factor(hf)

lev2<-ggplot(data=h.mat, aes(x=V1, y=h2, shape=hf.f))+geom_point(size=4)+
scale_shape_manual(values=c(16, 17))+ mytheme.roc+
               xlab("Estimated Probability")+
               ylab("Leverage")+
               ggtitle("Leverage vs. \n Estimated Probability")+
               theme(legend.position="none")

lev2 <- lev2+ geom_point(data=NULL, aes(x=v1.2, y=h2.2), shape=17,
colour="white", fill='white', size=2)
```

```
# change in chi2 vs est prob
plot(x=dig$P, y=dig$dChisq) #looks good for the most part

ch<-ggplot(data=dig, aes(x=P, y=dChisq))+ geom_point()+mytheme.roc+
              xlab("Estimated Probability")+
              ylab("Change in Chi-Sq")+
              ggtitle("Change in Chi-Sq \n vs. Estimated Probability")
ch

sum(dig$dChisq >7) #about 2 percent of the data falls away from the norm


#change in deviance vs est prob
plot(x=dig$P, y=dig$dDev) # again looks okay

div<-ggplot(data=dig, aes(x=P, y=dDev))+ geom_point()+mytheme.roc+
              xlab("Estimated Probability")+
              ylab("Change in Deviance")+
              ggtitle("Change in Deviance \n vs. Estimated Probability")
div


#cook's distance vs est prob

plot(x=dig$P, y=dig$dBhat)
text(x=dig$P,y=dig$h, labels=as.character(obs), cex=0.5) #covariate
pattern 868 is bad; must

cook<-ggplot(data=dig, aes(x=P, y=dBhat))+geom_point()+mytheme.roc+
              xlab("Estimated Probability")+
              ylab("Cook's Distance")+
              ggtitle("Cook's Distance \n vs. Estimated Probability")

cook


cooks.nums<-which(dig$dBhat>.4)
cook2<-dig$dBhat
cook2[cooks.nums]<-.4
p2<-dig$P
cookf<-c(rep(0,868))
cookf[cooks.nums]<-1
v1.3<-dig$P
v1.3[-cooks.nums]<-NA
cooks2.2<-cook2
cooks2.2[-cooks.nums]<-NA


cooks.mat<-as.data.frame(cbind(dig$P, cook2))
cooks.mat$cooksf.f<-factor(cookf)
```

```
cook_2<-ggplot(data=cooks.mat, aes(x=V1, y=cook2,
shape=cooksf.f))+geom_point(size=4)+ scale_shape_manual(values=c(16, 17))+
mytheme.roc+
              xlab("Estimated Probability")+
              ylab("Cook's Distance")+
              ggtitle("Cook's Distance\n vs. Estimated Probability")+
              theme(legend.position="none")

cook_2 <- cook_2+ geom_point(data=NULL, aes(x=v1.3, y=cooks2.2), shape=17,
colour="white", fill='white', size=2)



#roc plot-----------------------------------------------------------


library('pROC')

prob=predict(g3,type=c("response"))
final.model.data$prob=prob
g <- roc(group ~ prob, data = final.model.data)

auc(g)

x.log<-g$specificities
y.log<-g$sensitivities


x.points<-1-x.log
y.points<-y.log

x.points[1]<-.25
y.points[1]<-.76


for(i in 1:869){

      if(x.points[i]==.25){

      x.points[i]=x.points[i]

      }

      else{

      x.points[i]=NA

      }

}



for(i in 1:869){
```

```
      if(y.points[i]==.76){

      y.points[i]=y.points[i]

      }

      else{

      y.points[i]=NA

      }

}



plot.roc<-ggplot(data=NULL, aes(x=1-x.log,
y=y.log))+geom_line()+mytheme.roc+
                        xlab("1-Specificity") +
                        ylab("Sensitivity")+
                        ggtitle("ROC for Logistic Model")+
                        geom_vline(xintercept = .25, colour = "red",
size=.7)+
                        geom_hline(yintercept = .76, colour = "red",
size=.7)+
                        geom_point(data=NULL, aes(x=x.points, y=y.points),
shape= 21, colour="white", fill='white', size=7.5)+
                        geom_point(data=NULL, aes(x=x.points, y=y.points),
shape= 13, colour="blue", fill='white', size=6)





#table

table(g3$y,fitted(g3)>0.364)


#svm-----------------------------------------------------------------------
------------------------------
#trying SVM on finaldata

load('finaldata.RData')
finaldata$Label<-NULL

data1<-finaldata

y<-data1[,256]

data1$group<-NULL
```

```
y=as.factor(y)

data1<-cbind(data1, y)

data1=data.frame(data1)

library (e1071)

set.seed (9347053)

#training and testing data;
train=sample(869 ,608)
svmfit =svm(y~., data=data1[train ,], kernel ='polynomial', degree =.5,
cost =.1)

summary (svmfit)

#tuning the model

set.seed (9347053)

tune.out=tune(svm, y~., data=data1[train ,], kernel ='sigmoid',
ranges =list(cost=c( 0.1 , 5, 10, 50),
coef0=c(0.0005, .005,0.05, .5) ))
summary (tune.out)


#table of the best model using testing data
table(true=data1[-train,'y'], pred=predict(tune.out$best.model
,newdata=data1[-train,]))


#ROC Curves

library(ROCR)

#obtaining fitted values
svmfit.opt=svm(y~., data=data1[train ,], kernel ='sigmoid', coef0 =.5,
cost=.1, decision.values =T) #sigmoid is the best (out of linear and
raidal)
fitted=attributes(predict(svmfit.opt ,data1[train ,], decision.values
=TRUE))$decision.values


#produce ROC plot

par(mfrow =c(1,2))
rocplot(fitted,data1[train,'y'], main="Training Data")

predob=prediction(fitted, data1[train,'y'])
perf=performance(predob, 'tpr', 'fpr')

#convert s4 object to data frame
```

```
library('raster')

df.fp<-as.data.frame(perf@x.values)

df.spec<- (1-df.fp)

df.spec<-as.data.frame(df.spec)

colnames(df.spec)<-c('Specificity')

df.sens<-as.data.frame(perf@y.values)

colnames(df.sens)<- c('Sensitivity')

df.sens<-as.data.frame(df.sens)


svm.roc<-cbind(df.sens, df.spec)

#convert other s4 object -- test data, just as bad

fitted.test=attributes(predict(svmfit.opt ,data1[-train ,],
decision.values =T))$decision.values

predob.test=prediction(fitted.test, data1[-train,'y'])
perf.test=performance(predob.test, 'tpr', 'fpr')

df.fp.test<-as.data.frame(perf.test@x.values)

df.spec.test<- (1-df.fp.test)

df.spec.test<-as.data.frame(df.spec.test)

colnames(df.spec.test)<-c('Specificity')

df.sens.test<-as.data.frame(perf.test@y.values)

colnames(df.sens.test)<- c('Sensitivity')

df.sens.test<-as.data.frame(df.sens.test)

svm.roc.test<-cbind(df.sens.test, df.spec.test)

#roc plot to comapre svm and logistic regression model ---NOTE only run
after logistic model code was run

svm.roc<-as.data.frame(svm.roc)
svm.roc.test<-as.data.frame(svm.roc.test)

svm.roc.1<-matrix(nrow=869, ncol=2)

svm.roc.1[,1]<-c(svm.roc[,1], 4:264)
svm.roc.1[,2]<-c(svm.roc[,2],4:264)
```

```
svm.roc.2<-matrix(nrow=869, ncol=2)
svm.roc.2[,1]<-c(svm.roc.test[,1], 4:610)
svm.roc.2[,2]<-c(svm.roc.test[,2], 4:610)


for(i in 1:869){

      if(svm.roc.1[i,1]>2){

      svm.roc.1[i,1]=NA

      }

      else{

      svm.roc.1[i,1]=svm.roc.1[i,1]

      }

}


for(i in 1:869){

      if(svm.roc.1[i,2]>2){

      svm.roc.1[i,2]=NA

      }

      else{

      svm.roc.1[i,2]=svm.roc.1[i,2]

      }

}




for(i in 1:869){

      if(svm.roc.2[i,1]>2){

      svm.roc.2[i,1]=NA

      }

      else{

      svm.roc.2[i,1]=svm.roc.2[i,1]
```

```
      }
}


for(i in 1:869){
      if(svm.roc.2[i,2]>2){
      svm.roc.2[i,2]=NA
      }
      else{
      svm.roc.2[i,2]=svm.roc.2[i,2]
      }
}



svm.roc.1<-as.data.frame(svm.roc.1)
svm.roc.2<-as.data.frame(svm.roc.2)



plot.roc.compare<-ggplot(data=svm.roc.1, aes(x=(1-V2),
y=(V1)))+geom_line(colour='red')+mytheme.roc+
                        xlab("1-Specificity") +
                        ylab("Sensitivity")+
                        ggtitle("ROC for SVM\nTraining and Test Data")

plot.roc.compare<-plot.roc.compare + geom_line(data=svm.roc.2, aes(x=(1-
V2), y=V1), colour='darkorchid4')



plot.roc.compare<-ggplot(data=svm.roc.1, aes(x=(1-V2),
y=(V1)))+geom_line(colour='red')+mytheme.roc+
                        xlab("1-Specificity") +
                        ylab("Sensitivity")+
                        ggtitle("ROC for Logistic\nModel and SVM")

plot.roc.compare<- plot.roc.compare + geom_line(data=NULL, aes(x=1-x.log,
y=y.log), colour='darkgreen')

plot.roc.compare<-plot.roc.compare + geom_line(data=svm.roc.2, aes(x=(1-
V2), y=V1), colour='darkorchid4')
```

```
#k means-------------------------------------------------------------------
------------------------------


wssplot <- function(data, nc=15, seed=584981){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of Clusters",
                     ylab="Within groups sum of squares")}


load('finaldata.RData')

mydata<-finaldata[,-1]
mydata$groups<-NULL

wssplot(mydata) #says two clusters best, but maybe 8

library(NbClust)

set.seed(584981)
fit <- NbClust(mydata, min.nc=2, max.nc=8, method="kmeans")

table(fit$Best.n[1,])

barplot(table(mydata$Best.n[1,]),
          xlab="Numer of Clusters", ylab="Number of Criteria",
          main="Number of Clusters Chosen by 26 Criteria")


 set.seed(584981)
 fit.km <- kmeans(mydata, 2, nstart=25) #i have the object saved, so I can
just load it now                          #3

 fit.km$size
 fit.km$centers

aggregate(finaldata[-1], by=list(cluster=fit.km$cluster), mean)

ct.km <- table(finaldata$group, fit.km$cluster)

ct.km

library(flexclust)
randIndex(ct.km)


#other clustering-----------------------------------------------------------
------------------------------
```

```
#clustering the image data
load('finaldata.RData')
finaldata.clustering<-finaldata

finaldata.clustering$Label<-NULL


finaldata.clustering$group<-NULL


augMat <- t(apply(finaldata.clustering[,1:255],1,diff))
colnames(augMat) <- paste("Dif_",colnames(finaldata.clustering)[-
1],sep='')
picMat <- cbind(finaldata.clustering[,1:255],augMat)

eucDist <- dist(picMat)

clus3 <- hclust(eucDist, method= "ward")

finaldata$clusternum<-cutree(clus3, 2)

plot(clus3)

marks <- c(0,4000000,8000000,12000000)

plot(clus3, labels=FALSE, main='Ward Clustering',ylab='Height',
xlab='Variable Clusters', yaxt='n', cex=.6)
axis(2,at=marks,labels=format(marks,scientific=FALSE))
rect.hclust(clus3, k=2, border="red")

numright<-c()

for (i in 1:299){

      if(finaldata[i,257]==finaldata[i,258]){

      numright[i]=1

      }

      else{

      numright[i]=0

      }

}

for (i in 300:869){

if(finaldata[i,257]+2==finaldata[i,258]){

      numright[i]=1
```

```
      }

      else{

      numright[i]=0

      }

}

s<-sum(numright)

l<-length(numright)

s/l



#roc curve ------------------------------------------------------------------
--------------------------------

#logistic roc
mytheme.roc<-theme(

      plot.title = element_text(lineheight=1.5, size=35, face="bold"),
      axis.text.x=element_text(size=23),
      axis.text.y=element_text(size=23),
      axis.title.x=element_text(size=28, face='bold'),
      axis.title.y=element_text(size=28, face='bold'),
      strip.background=element_rect(fill="gray80"),
      panel.background=element_rect(fill="gray80"),
      axis.ticks= element_blank(),
      axis.text=element_text(colour="black"),
      plot.margin = unit(c(27,13,27,13), 'mm')

      )


library('pROC')

prob=predict(g3,type=c("response"))
finaldata$prob=prob
g <- roc(group ~ prob, data = finaldata)

x.log<-g$specificities
y.log<-g$sensitivities


plot.roc<-ggplot(data=NULL, aes(x=1-x.log,
y=y.log))+geom_line()+mytheme.roc+
                        xlab("1-Specificity") +
                        ylab("Sensitivity")+
                        ggtitle("ROC Plot")
```

```
#svm roc                                          JUST USE LOGISTIC REGRESSION
FOR NOW
predicted.svm<-fitted(svmfit.opt)


#sensitivity function; calculate true positive rate
sen<-function(one,zero, t){

      TP=sum(one>t)
      FP=sum(zero>t)
      final<-TP/(TP+FP)
      print(final)

}

#specificity function; calculates true negative rate
spec<-function(one,zero,t){

            FN<-sum(one<t)
            TN<-sum(zero<t)
            final<-TN/(TN+FN)
            print(final)
}

#creating ROC curve

tees<-seq(from=0, to=1, by=(1/608))

sora<-length(tees)

sen_roc<-c()

for (i in 1:sora){

sen_roc[i]<-sen(P_fall, P_winter, tees[i])

}
spec_roc<-c()

for (i in 1:sora){

spec_roc[i]<-spec(P_fall, P_winter, tees[i])

}


plot(x=1-spec_roc, y=sen_roc)
```