Utilizing Data Analysis Methods for Photographic Classification

Howell Lu
York University

Author Note

For Mrs. Xin Gao's Math 3333 Class

Abstract

Abstract: The purpose of this project is to analyze the 800 image Columbia Dataset and to classify these images as taken outdoors during the day or not. The main objective of this specific project is to accurately identify images at a higher rate than the benchmark method. The benchmark method used a logistic regression and used the RGB of an image as the independent variables for the regression. That method when run 50 times, results in an average AOC of 81.4 percent. I attempted to do better. My methodology was rearranging the image dataset into a matrix with 800 columns for each picture to retrieve PCAs. I then proceeded to create and train a randomforest model using the first 40 PCAs and 400 images for training and then applied the random forest algorithm on the dataset. Results were excellent as an AUC of 95.99 was achieved.

Keywords: Data Analysis, Neural Networks, Principal Component Analysis, Support Vector Machines, Random Forests.

Utilizing Data Analysis Methods for Photographic Classification

Introduction

Image Classification has become more and more prevalent of an issue as humans spend more and more time interacting with technology. The question of Captcha and facial recognition speaks to the growing capacity for computers to recognize and classify images. However, there are still many limitations in terms of image classification with regards to predictive accuracy and computing power. The author of this report is limited by using simply RStudio, common R libraries and the Intel i7 5820K, a processor which was released in September 2014. He attempts to develop a method to identify images within the Columbia Dataset (800 images taken near the Columbia campus) and attempts to identify whether the image was taken outdoors and in daylight or not. The benchmark model used logistic regression in the images based on the RGB on each image, the model was trained on 400 images and tested on the remaining 400 and achieved an average AOC of 81.4%. [EXHIBIT A]

Data

Our dataset is overwhelmingly large, each picture is contains 722*480*3 features, some photos are 737*492, they average around 1039680 features which is around 8.32MB. The entire dataset is 6.962 GB. The data itself is comprised of differing images. Pictures taken outside during the day are generally composed of greenery and concrete buildings with a cloudy backdrop. However, there are noticeable exceptions where there are pictures of water, photos taken of flags and photos of humans, animals and billboards which are distinct from the average outdoor_day photo, comprised of gray buildings and leafy greenery. More importantly, some photos have been taken horizontally while others have been taken vertically.

A cursory glance at the pictures and its variability shows that many methods such as convolutionary neural networks simply would not work on such a varied dataset.

Methodology.

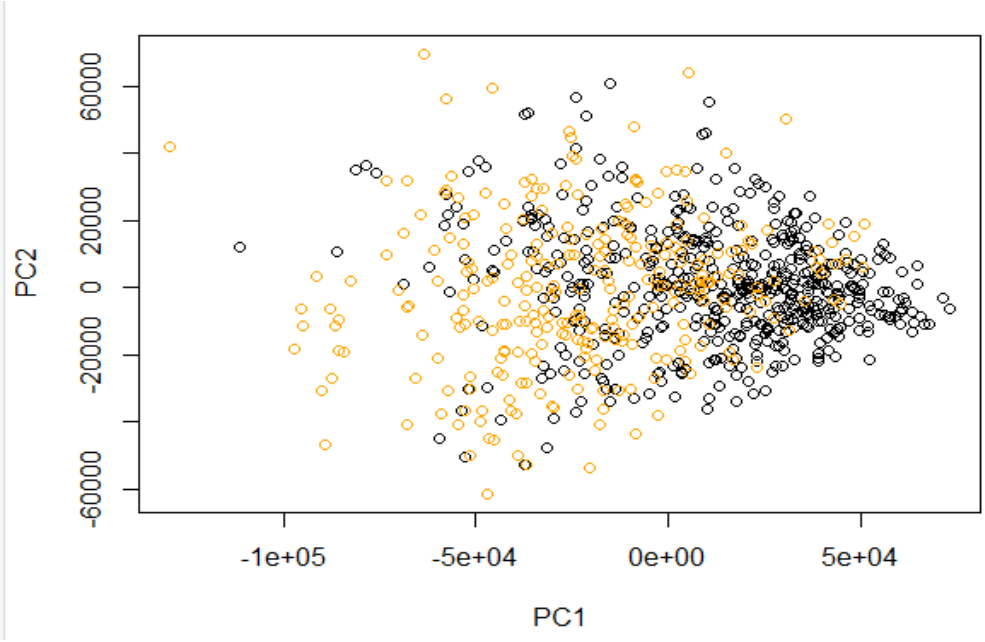
The idea begins with simplifying the large dataset into a reduced number of variables whilst also retaining as much of the variability as possible. PCA or principal component analysis is one of the main methods used to reduce complexity whilst preserving information. However, to create a matrix that PCA could be run on, we would need to reduce each photo from a three-variable matrix into a single column. The method that I used was to begin at [1,1] and to make the RGB of the first point into the first, second and third rows of the column respectively, then

we would go to [1,2] RGB to create the fourth, fifth and sixth row of a single column. This would continue until we finish the entire width of the first row and then we would go along the width of the second row until we transcribe the entire second row. This would go on until every row is complete. [EXHIBIT B]

There were some noticeable concerns that I had about the procedure I was using. Firstly, the dimensions of the photos are misaligned as some were rotated and others had different dimensions, this would make an impact as the covariance matrix would be impacted, and PCA works through the process of calculating the covariance matrix, and then calculating the eigenvectors to generate the principal components. However, rearranging pictures and cropping would take too long and would reduce the total amount of information. However, it should be acknowledged that there could have been more done for data preprocessing [CodeRTime].

I used another library called “magick” to read the images as it used 0-255 to represent the RGB values. I thought that simple integers could take less space than decimals. However, I soon realized that the values used under the exact same amount of space as fractions were used. I decided to keep the matrix as it took a long time to make. I had decided to merge column bind all 800 columns of images to get a matrix. This was very computationally intensive and took a several hours to complete. The next step is to run principal component analysis and used the command prcomp to retrieve the PCAs.

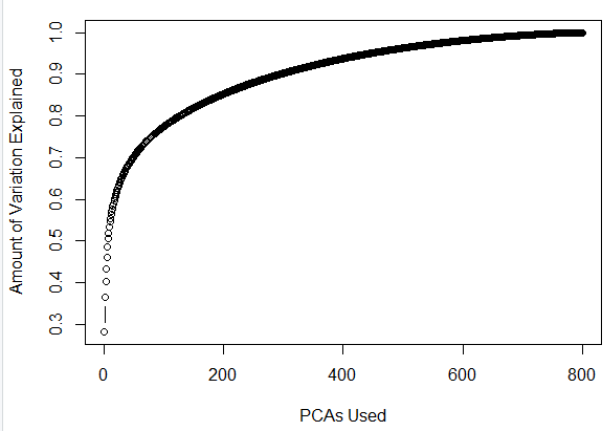
Each PCA is an arbitrary metric that explains some of the variation in the data set. The first PCA explains most of the variation and second the second most and so forth. (Gao) Specifically, each PCA is essentially an 800-dimensional line of best fit through our 800-dimensional set of data, with over a million observations. Each pixel being an observation and each image being a dimension. Within each PCA line, we can find close values for some images which means that images have some similarities. For example, in a matrix comparing Height, Weight and Number of hair follicles, the PCAs values of height and weight would be very closely related, while the PCA of number of hair follicles and weight would not be. Likewise, certain pictures can be very closely related to each other. The orange dots are the values classified as outdoor_day, Outdoor images are correlated with lower values of PC1.[EXHIBIT C]



For example, let us look at instances where PCA1 is less than -60000 and PCA 2 is less than -20000. There are six pictures, (Images 2, 570,573,596,599,710) and this is the cluster to our bottom left. Let us see if we can find any similarities between these pictures. [EXHIBIT D]

Although these photos do not share distinctive similarities, we can notice that there are usually high amounts of white to blue shades in the second and third images and the bottom 3 pictures are extremely similar, with bright blue skies and blueness in the form of a lake or a large building. It's impossible to know what arbitrary metric PCA1 and PCA2 are attempting to define, but there are clear visual similarities between pictures that are closely lumped together.

The next choice was the exact amount of PCAs to retain in the model, and we would select by plotting the total amount of variance explained. [EXHIBIT D]



PCAs Used	Amount of Variance Explained
50	70.28
100	77.45
200	85.29
400	93.76
600	98.11

Afterwards, I had proceeded to use our principal components as dependent variables and had proceeded to test them using four methods. Logistic regression, Neural Networks, Random Trees and Support Vector Machines. I have listed the results from each method.

One method that I used to check for overfitting and outliers was by creating our model 50 times with different images and running our model and calculating AOC 50 times, taking the average of AOC of the attempts, this would remove outliers like how k-fold cross validation would. These were my results with each method. I also set the amount of training data at 400 images, despite 80:20 being a more conventional practice. I wanted our model to be directly comparable to the benchmark and I feared using 640 would lead to overfitting. [EXHIBIT F]

200 PCAs

	AOC	Runtime
Logistic Regression	0.8209	20.33 seconds
Random Forest	0.9546	5.548 minutes
SVM	0.9357	15.02 seconds
Neural Network	0.8955	1.178 min

Analysis.

There are a multitude of questions I thought were pertinent

1.) Does the way we aligned the images into a column make a difference?

2.) Which model works the best and does that change with the number of principal components?

3.) How different is the runtime for each specific method and each amount of variables.

4.) What was my justification for the settings that I had used for each model?

One experiment that I made was to see if there was a large difference between setting up the matrix while separating the pixels, having all the reds as the first 1/3 of the matrix column, greens as the second 1/3 and blues as the third 1/3 rather than adding each pixel's RGB sequentially. Then I proceeded to take out the first 200 variables and run the exact same models to see if results were substantially different. Did the first PCA's store a higher amount of variance? Were our models more accurate?

There was practically no difference between the whether we plotted out each pixel's RBG or decided to list all the Red, Greens and Blues first. The amount of variation captured by the first N PCAs were nearly identical, and the performance of each algorithm was also nearly identical. [EXHIBIT I]

There was little possibility of customization of the logistic regression and the svm method due to the nature of the method or limitations with the library. However, there were substantial questions regarding the settings used for neural networks.

I ran the randomForest algorithm because I worried that normal decision trees would lead to the problem of overfitting. The project wanted to classify the variables so I used 14 features for each tree and I had over 200 features and the common practice was to use the sqrt(features) as the value of mtry. I also decided to use 1000 random trees because I had a surfeit of computing power. (Gao)

The question of how many hidden layers is needed in a neural network is complicated. Too few hidden layers lead to imprecise predictions, while too many leads to overfitting. However, it was stated that two hidden layers is enough to represent an arbitrary decision boundary to arbitrary accuracy and can approximate any smooth mapping. Likewise, too many neurons create overfitting and too few leads to underfitting. Common practice is that the number of the hidden neurons should be about 2/3rd the size of the input layer plus the size of the output layer and the number of hidden layers should be less than twice of the input layer. (Bhattiprolu). Therefore, I decided to keep the number of hidden layers at 85 with 60 in the first layer and 25 in the second. There was no true consensus regarding what an ideal learning rate was, so I tried multiple values and found that our AUC did not change. However, lower learning rates usually led to a higher AUC. But the difference in AUC between a learning rate of 0.1 and a learning rate of 0.001 was a fraction of a percent.

The last thing that I wanted to try is to see if our predictive accuracy changes substantially when we include a greater number of PCAs. It's important to note that I only used 397PCAs for logistic regression as we only had 400 training observations.

I noticed that higher numbers of regressors didn't automatically mean higher accuracy, so I decided to test all these methods with differing amount of PCA predictors to find an optimal value. For the sake of brevity, I didn't add the images of each experiment in the appendix, just run the code and you can find the results yourself. I have also scaled the RandomForest's number of features and the number of hidden neurons respectively. Each iteration ran 50 times to take the average AOC. I have added the runtime in parentheses

N	SVM	RandomForest	Neural Nets	Logistic Regression
800	0.90754	0.9519	0.71854	More Regressors than Variables
400	0.9222 (RT25.15 sec)	0.9519 (RT 7.75 min)	0.8993 (RT 3.05 min)	0.6762 RT(52.71 sec)
150	0.9352	0.9571	0.8860	0.8716
100	0.9385	0.9561	0.8702	0.8556
80	0.9387 (RT 8.99 sec)	0.9570	0.8635	0.8780
50	0.9359	0.9582	0.8318	0.8796
40	0.9366	0.9599 (2.2 min RT)	0.8221	0.8758
30	0.9331	0.9573	----------	0.8813
20	0.9185	0.9586	----------	0.8825
10	0.9040	0.9544	----------	0.8843
5	0.8768	0.9443	----------	0.8772
3	0.8065	0.9072	----------	0.7888

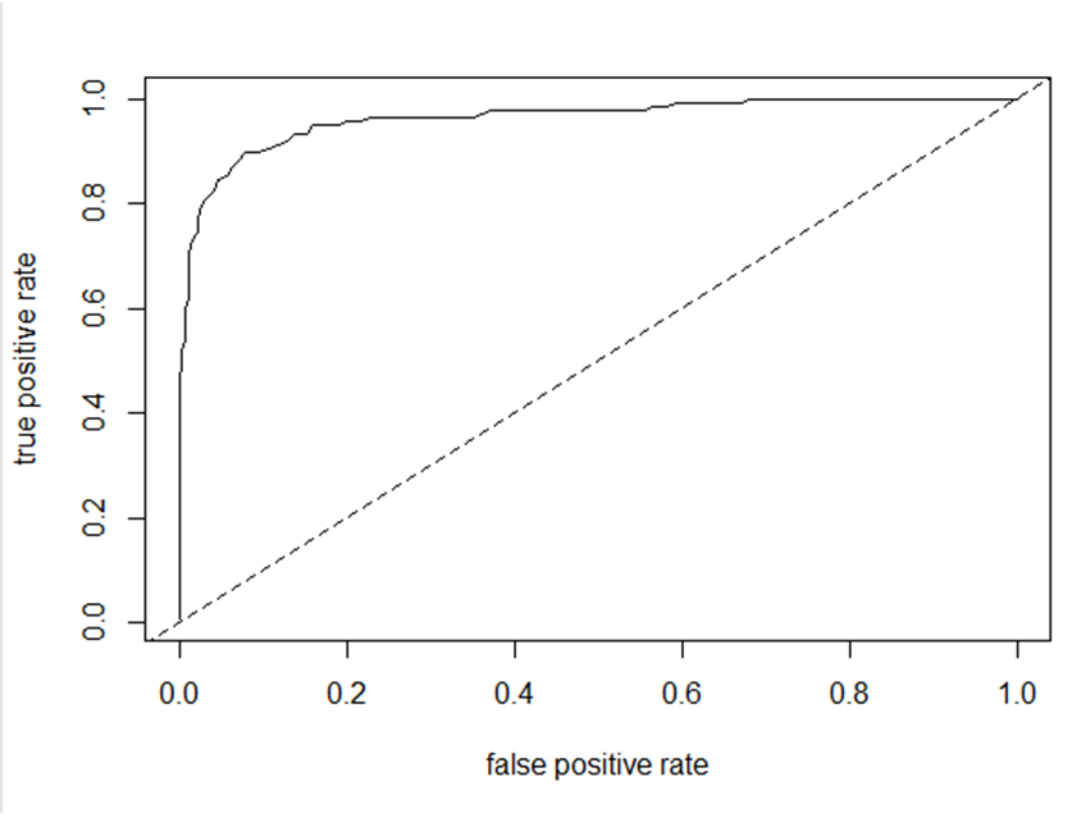
Worth noting is the fact that Neural Networks fail to converge unless the learning rate is made larger than 0.01 when we have 30 or less regressors. However, since neural networks are not the ideal method of image prediction, I decided not to finish the method off.

Model.

Our optimal model is using the Random Forest method using PCA#1 to PCA#40 as our 40 regressors. We would generate 1000 random trees and use 7 estimators for generation

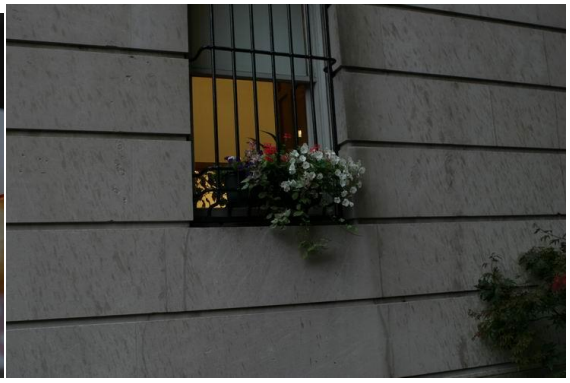
purposes. This is an ensemble method and samples the data multiple times (over 1000), generating new models and take the majority vote. This method is rather computationally intensive, and the method of Support Vector Machines (with 80 PCAs) could be used to obtain similar levels of accuracy whilst saving on computing power. However, one could also lower the number of trees generated to save on computing time, as the same model with 40PCAs and only 100 trees would take 24 seconds to compute, resulting in an accuracy of 0.9536.

Result.



The result was an AOC of 95.99% when run 50 times and taking the average of the runs. The model had a sensitivity rate of 82.04%, a specificity of 94.1% and a misrate of 9.98% when we use 0.5 as the cutoff. [EXHIBIT G] I decided to look at our testing data in depth and to study the extreme false positives and the false negatives for any possible details. There are 19 false negatives in our test sample but 14 of them were estimated between 0.4-0.5. So, I decided to look at the most egregious cases where our algorithm predicted value of 0.08, 0.175, 0.35 and 0.37 for images 365, 104, 12 and 595. [EXHIBIT H]

	Over 0.9 away from true value	0.8-0.9 away from true value	0.7-0.8 away from true value	0.6-0.7 away from true value	0.5-0.6 away from true value	Total Amount
False Negative	1	1	0	3	14	19
False Positive	0	0	1	5	8	14

It seems from these observations that this algorithm usually fails to recognize pictures that do not have an obvious outdoor backdrop to them. The first picture is mainly centered around the food itself and lacks any distinguishing characteristics. To be frank, it's extremely hard for me to tell whether this was taken indoors or outdoors. The second picture is mostly concrete with a tinge of indoor lighting and doesn't have any indicators of it being taken outdoors except that it is pointed at a wall, something only a human can discern. The third picture is a concrete building and difficult for a computer to ascertain whether that would be indoors or not. The one last case that I found interesting was image 595 which is clearly outdoors but it seemed like the algorithm was confused with the colors of the balloon.

In the case of false positives, we had a total of 14 false positives and the most egregious ones were pictures 743, 754, 681. They are predictions of 0.76, 0.69 and 0.67. These are the

photos: [EXHIBIT H]



The first one is deemed as “natural” for some reason. I think that might be a mislabel. The second picture is deemed as artificial, and the last picture is deemed as outdoor-dawn-dusk. The clear problem with the labeling on these images is the fact that all these categories are clearly overlapping with “outdoor_day”. If you were to simply ask a person on the street whether any of these photo was taken outdoors during the day, they would clearly say yes. Furthermore, these pictures often are indistinguable from outdoor_day. Picture 672 (first image) was labeled as outdoor_Day but also was hued. Likewise, image 696 (second image) is listed as outdoor_day despite also being an artificial building like our monument.


Likewise, image 507(left) is deemed as natural while image 551(right) is deemed as outdoor_day despite having a similar number of non-natural materials.


This model tends to have great results. However, it tends to have problems with rather

ambiguous images such as extreme close ups on items, or very nuanced and arbitrary labels, but humans would likely have problems with predicting these images too.

Conclusion.

To conclude, many different methods have been used to predict images. However, for this particular set of pictures it appears that the RandomForest created the best results. Many things have been gleaned from this experience. Firstly, compiling image data is an extremely computationally intensive act. It took several hours to simply compile our image matrix and to run PCA analysis. PCAs are correlated with each other and can be used as valid regressors or features of images so one can derive important information from large swathes of unrelated data.

However, there is an optimal amount of PCAs required for each method as adding too many PCAs simply brings unnecessary noise, increases computational complexity, and lowers predictive accuracy. The optimal amount of PCAs is different for each method.

Our optimal method was the process of using RandomForest with 40PCAs as predictors and using 7 features in the random forest while also having 1000 trees and we ended up with an AOC of 0.9599 (average of 50 times). The model had a sensitivity rate of 82.04% and a specificity of 94.1% when we use 0.5 as the cutoff. However, the true accuracy of this model would be higher if image labeling were more precise.

References

Bhattiprolu, Sreenivas. “155 - How Many Hidden Layers and Neurons Do You Need in Your Artificial Neural Network?” *YouTube*, YouTube, 2 Sept. 2020, www.youtube.com/watch?v=bqBRET7tbiQ.

Gao, Xin. “R-On-Decision-Trees.” Random Forests, York University, 2021, pp. Slide 45-Slide 47.

Gao, Xin. “R-On-PCA.” PCA, York University, 2021, pp. Slide 2.

Gao, Xin. “Xin Gao's Code.” York University, 2021.

CodeRTime. “R Image Processing and Image Clustering: Simple Computer Vision in R.” *YouTube*, YouTube, 30 Nov. 2020, www.youtube.com/watch?v=nkX_gQKsFzQ.

Exhibits

EXHIBIT A:

```
> storeme<-0
> for (i in 1:50) {
+
+     trainFlag <- (runif(n) > 0.5)
+     y <- as.numeric(pm$category == "outdoor-day")
+
+     # build a glm model on these median values
+     out <- glm(y ~ X, family=binomial, subset=trainFlag)
+     out$iter
+     summary(out)
+
+     # How well did we do? We only get the regression from the first bit
+     pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(out)))
+
+     +(as.numeric(pred > 0.5) == y)
+     +mean((as.numeric(pred > 0.5) == y))
+     mean((as.numeric(pred > 0.5) == y)[trainFlag])
+     mean((as.numeric(pred > 0.5) == y)[!trainFlag])
+
+     ## ROC curve (see lecture 12)
+     roc <- function(y, pred) {
+         alpha <- quantile(pred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(pred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[!trainFlag], pred[!trainFlag])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
>
> storeme/50
[1] 0.8145965
>
```

EXHIBIT B:

```
for (j in 1:n) {
    img <- readJPEG(paste0("C:\\Users\\howel\\OneDrive\\Desktop\\Math 3333 Project\\columbiaImages\\",pm$name[j]))
    current_image <- image_read(img)
    current_image_data <- as.numeric(current_image[[1]][,,])
    FatMatrix <- rbind(FatMatrix,current_image_data)
    print(j)
}

compressedpca<-prcomp(FatMatrix)
```

```
Choosing How many PCAs we need in our X

PCASTORE<-compressedpca$x[,1:800]
x<-PCASTORE[,1:200]
y <- as.numeric(pm$category == "outdoor-day")
NeededPCA<-cbind(y,x)


plot(NeededPCA[,2:3])
typea<-grep(NeededPCA[,1], pattern = 1)
points(NeededPCA[typea,2:3],col="orange")
```

EXHIBIT C:

EXHIBIT D:

Image Number	PCA 1	PCA 2
Image 2	-90016	-30368
Image 570	-89126	-46582
Image 573	-68000	-40655
Image 596	-60053	-20973
Image 599	-87528	-26949
Image 710	-65780	-26956

EXHIBIT E:

```
Plotting


plot(cumsum(compressedpca$sdev^2 / sum(compressedpca$sdev^2)), type="b", xlab="PCAs Used", ylab="Amount of Variation Explained")
cumsum(compressedpca$sdev^2 / sum(compressedpca$sdev^2))
```

EXHIBIT F:

```
> start_time <- Sys.time()
> framex <- as.data.frame(x[,1:200])
> framey<-as.data.frame(y)
> frame<-cbind(framex,framey)
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     RegModel<-glm(y~.,data = frame,family=binomial, subset = trainingID,maxit=200)
+     framex<-as.matrix(framex)
+     pred <- 1 / (1 + exp(-1 * cbind(1,framex) %*% coef(RegModel)))
+     mean((as.numeric(pred > 0.5) == y)[testID])
+
+
+     roc <- function(y, pred) {
+         alpha <- quantile(pred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(pred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], pred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> storeme/50
[1] 0.820905
> end_time <- Sys.time()
> end_time - start_time
Time difference of 20.32781 secs
>
```

```
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     nn <- neuralnet(y ~ ., data=NeededPCA[trainingID,], hidden=c(60,25), linear.output=FALSE, threshold=0.1)
+     nn.results <- compute(nn, x)
+     results <- data.frame(actual = y, prediction = nn.results$net.result)
+     ynn<-nn.results$net.result
+     mean((as.numeric(ynn > 0.5) == y)[testID])
+
+
+     roc <- function(y, ynn) {
+         alpha <- quantile(ynn, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(ynn >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], ynn[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
> storeme/50
[1] 0.8955316
> end_time <- Sys.time()
> end_time - start_time
Time difference of 1.178422 mins
```

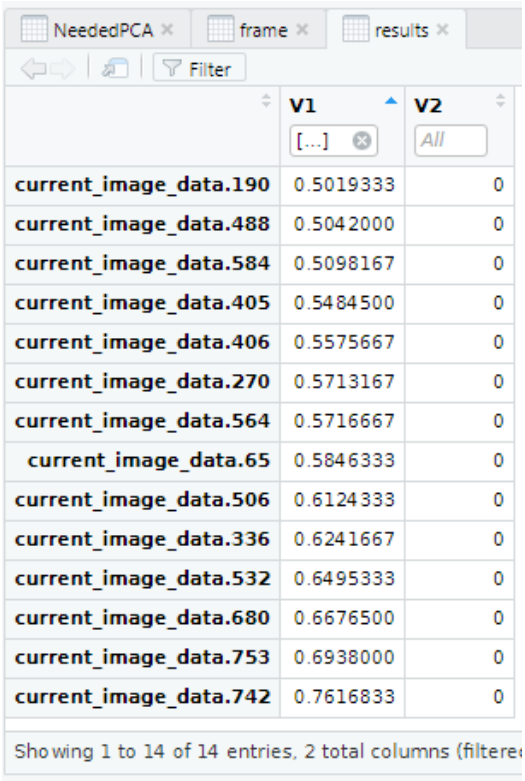
```
> x<-PCASTORE[,1:400]
> y <- as.numeric(pm$category == "outdoor-day")
> NeededPCA<-cbind(y,x)
> library(e1071)
> library(neuralnet)
> library(nnet, lib.loc = "C:/Program Files/R/R-4.0.5/library")
> library(tree)
> ## ROC curve (see lecture 12)
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     svmmodel<-svm(x[trainingID,],y[trainingID])
+     pred<-predict(svmmodel,x)
+     roc <- function(y, pred) {
+         alpha <- quantile(pred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(pred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], pred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+ }
> storeme/50
[1] 0.9222336
> end_time <- Sys.time()
> end_time - start_time
Time difference of 25.15675 secs
```

```
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     rf_classifier = randomForest(y ~ ., data=NeededPCA[trainingID,], ntree=1000, mtry=14, importance=TRUE)
+     treepred<-predict(rf_classifier,x)
+     mean((as.numeric(treepred > 0.5) == y)[testID])
+
+     roc <- function(y, treepred) {
+         alpha <- quantile(treepred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(treepred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], treepred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> storeme/50
[1] 0.9545914
> end_time <- Sys.time()
> end_time - start_time
Time difference of 5.548285 mins
```

EXHIBIT G:

```
> ttp<-0
> ttn<-0
> tfp<-0
> tfn<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     rf_classifier = randomForest(y ~ ., data=NeededPCA[trainingID,], ntree=1000, mtry=7, importance=TRUE)
+     treepred<-predict(rf_classifier,x)
+     tp<-sum((round(treepred)==1)*(y==1))
+     tn<-sum((round(treepred)==0)*(y==0))
+     fp<-sum((round(treepred)==1)*(y==0))
+     fn<-sum((round(treepred)==0)*(y==1))
+
+     roc <- function(y, treepred) {
+         alpha <- quantile(treepred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(treepred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], treepred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+     ttp<-ttp+tp
+     ttn<-ttn+tn
+     tfp<-tfp+fp
+     tfn<-tfn+fn
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> storeme/50
[1] 0.9598753
> end_time <- Sys.time()
> end_time - start_time
Time difference of 2.2065 mins
> misrate<-(tfp+tfn)/40000
> sensitivity<-ttp/(ttp+tfn)
> specificity<-ttn/(ttn+tfp)
> misrate
[1] 0.099825
> sensitivity
[1] 0.820361
> specificity
[1] 0.9424474
```

Exhibit H: (The index that I'm using started from 0 so I had to manually add a number).

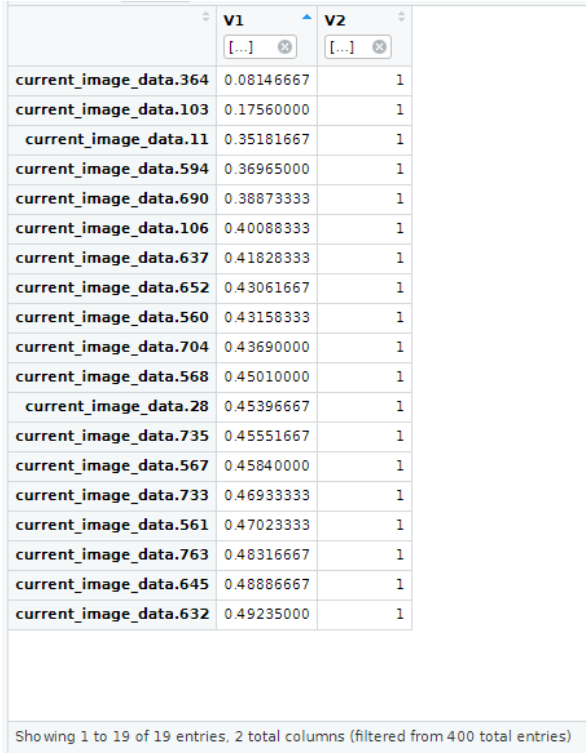


NeededPCA × | frame × | results ×

Filter

	V1 [...]	V2 All
current_image_data.190	0.5019333	0
current_image_data.488	0.5042000	0
current_image_data.584	0.5098167	0
current_image_data.405	0.5484500	0
current_image_data.406	0.5575667	0
current_image_data.270	0.5713167	0
current_image_data.564	0.5716667	0
current_image_data.65	0.5846333	0
current_image_data.506	0.6124333	0
current_image_data.336	0.6241667	0
current_image_data.532	0.6495333	0
current_image_data.680	0.6676500	0
current_image_data.753	0.6938000	0
current_image_data.742	0.7616833	0

Showing 1 to 14 of 14 entries, 2 total columns (filtere



	V1 [...]	V2 [...]
current_image_data.364	0.08146667	1
current_image_data.103	0.17560000	1
current_image_data.11	0.35181667	1
current_image_data.594	0.36965000	1
current_image_data.690	0.38873333	1
current_image_data.106	0.40088333	1
current_image_data.637	0.41828333	1
current_image_data.652	0.43061667	1
current_image_data.560	0.43158333	1
current_image_data.704	0.43690000	1
current_image_data.568	0.45010000	1
current_image_data.28	0.45396667	1
current_image_data.735	0.45551667	1
current_image_data.567	0.45840000	1
current_image_data.733	0.46933333	1
current_image_data.561	0.47023333	1
current_image_data.763	0.48316667	1
current_image_data.645	0.48886667	1
current_image_data.632	0.49235000	1

Showing 1 to 19 of 19 entries, 2 total columns (filtered from 400 total entries)

EXHIBIT I:

```
for (j in 1:n) {
  img <- readJPEG(paste0("C:\\Users\\howel\\OneDrive\\Desktop\\Math 3333 Project\\columbiaImages\\",pm$name[j]))
  image_data <- as.numeric(img[,,])
  FatMatrix <- rbind(FatMatrix,image_data)
  print(j)
}
compressedpca<-prcomp(FatMatrix)
```

```
> library(e1071)
> ## ROC curve (see lecture 12)
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     svmmodel<-svm(x[trainingID,],y[trainingID])
+     pred<-predict(svmmodel,x)
+     roc <- function(y, pred) {
+         alpha <- quantile(pred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(pred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], pred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+ }
> storeme/50
[1] 0.9370175
> end_time <- Sys.time()
> end_time - start_time
Time difference of 14.72569 secs
```

```
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     rf_classifier = randomForest(y ~ ., data=NeededPCA[trainingID,], ntree=1000, mtry=20, importance=TRUE)
+     treepred<-predict(rf_classifier,x)
+     mean((as.numeric(treepred > 0.5) == y)[testID])
+
+     roc <- function(y, treepred) {
+         alpha <- quantile(treepred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(treepred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], treepred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> storeme/50
[1] 0.9518806
> end_time <- Sys.time()
> end_time - start_time
Time difference of 7.750706 mins
```

```
> PCASTORE<-compressedpca$x[,1:800]
> X<-PCASTORE[,1:200]
> y <- as.numeric(pm$category == "outdoor-day")
> NeededPCA<-cbind(y,x)
>
> start_time <- Sys.time()
> framex <- as.data.frame(x[,1:200])
> framey<-as.data.frame(y)
> frame<-cbind(framex,framey)
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     RegModel<-glm(y~.,data = frame,family=binomial, subset = trainingID,maxit=200)
+     framex<-as.matrix(framex)
+     pred <- 1 / (1 + exp(-1 * cbind(1,framex) %*% coef(RegModel)))
+     mean((as.numeric(pred > 0.5) == y)[testID])
+
+
+     roc <- function(y, pred) {
+         alpha <- quantile(pred, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(pred >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], pred[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> storeme/50
[1] 0.8203208
> end_time <- Sys.time()
> end_time - start_time
Time difference of 20.76233 secs
```

```
> start_time <- Sys.time()
> storeme<-0
> for (i in 1:50) {
+     trainingID<-sample(1:800,400)
+     testID<-sample(1:800)[-trainingID]
+     nn <- neuralnet(y ~ ., data=NeededPCA[trainingID,], hidden=c(120,50), linear.output=FALSE, threshold=0.01)
+     nn.results <- compute(nn, x)
+     results <- data.frame(actual = y, prediction = nn.results$net.result)
+     ynn<-nn.results$net.result
+     mean((as.numeric(ynn > 0.5) == y)[testID])
+
+
+     roc <- function(y, ynn) {
+         alpha <- quantile(ynn, seq(0,1,by=0.01))
+         N <- length(alpha)
+
+         sens <- rep(NA,N)
+         spec <- rep(NA,N)
+         for (i in 1:N) {
+             predClass <- as.numeric(ynn >= alpha[i])
+             sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
+             spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
+         }
+         return(list(fpr=1- spec, tpr=sens))
+     }
+
+     r <- roc(y[testID], ynn[testID])
+     plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
+     abline(0,1,lty="dashed")
+
+     # auc
+     auc <- function(r) {
+         sum((r$fpr) * diff(c(0,r$tpr)))
+     }
+     glmAuc <- auc(r)
+     glmAuc
+     storeme<-storeme+glmAuc
+
+ }
> storeme/50
[1] 0.8992933
> end_time <- Sys.time()
> end_time - start_time
Time difference of 3.053905 mins
```

```
### Photograph examples

## Read in the library and metadata
library(jpeg)
pm <- read.csv("C:\\Users\\howel\\OneDrive\\Desktop\\Math 3333
Project\\photoMetaData.csv")
n <- nrow(pm)

trainFlag <- (runif(n) > 0.5)
y <- as.numeric(pm$category == "outdoor-day")

X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n) {
  img <- readJPEG(paste0("C:\\Users\\howel\\OneDrive\\Desktop\\Math 3333
  Project\\columbiaImages\\",pm$name[j]))
  X[j,] <- apply(img,3,median)
  print(sprintf("%03d / %03d", j, n))
}

storeme<-0
for (i in 1:50) {

trainFlag <- (runif(n) > 0.5)
y <- as.numeric(pm$category == "outdoor-day")

# build a glm model on these median values
out <- glm(y ~ X, family=binomial, subset=trainFlag)
out$iter
summary(out)

# How well did we do? We only get the regression from the first bit
pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(out)))

+(as.numeric(pred > 0.5) == y)
+mean((as.numeric(pred > 0.5) == y))
mean((as.numeric(pred > 0.5) == y)[trainFlag])
mean((as.numeric(pred > 0.5) == y)[!trainFlag])

## ROC curve (see lecture 12)
roc <- function(y, pred) {
  alpha <- quantile(pred, seq(0,1,by=0.01))
  N <- length(alpha)

  sens <- rep(NA,N)
  spec <- rep(NA,N)
  for (i in 1:N) {
    predClass <- as.numeric(pred >= alpha[i])
    sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
    spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
  }
  return(list(fpr=1- spec, tpr=sens))
}

r <- roc(y[!trainFlag], pred[!trainFlag])
plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")

# auc
auc <- function(r) {
  sum((r$fpr) * diff(c(0,r$tpr)))
}
glmAuc <- auc(r)
glmAuc
storeme<-storeme+glmAuc

}


_____________________________________________________________________________
Generate PCAs
```

```
68
69
70
71
72
73  for (j in 1:n) {
74      img <- readJPEG(paste0("C:\\Users\\howel\\OneDrive\\Desktop\\Math 3333
        Project\\columbiaImages\\",pm$name[j]))
75      current_image <- image_read(img)
76      current_image_data <- as.numeric(current_image[[1]][,,])
77      FatMatrix <- rbind(FatMatrix,current_image_data)
78      print(j)
79  }
80
81  compressedpca<-prcomp(FatMatrix)
82
83
84  ____________________________________________________________________________________
    ____________________________________
85  PLOTTING AND SVM
86
87
88
89
90  Choosing How many PCAs we need in our X
91
92  PCASTORE<-compressedpca$x[,1:800]
93  x<-PCASTORE[,1:200]
94  y <- as.numeric(pm$category == "outdoor-day")
95  NeededPCA<-cbind(y,x)
96
97
98  plot(NeededPCA[,2:3])
99  typea<-grep(NeededPCA[,1], pattern = 1)
100 points(NeededPCA[typea,2:3],col="orange")
101
102
103 OneThree<-NeededPCA[,1:4]
104 OneThree<-OneThree[,-3]
105 plot(OneThree[,2:3])
106 typea<-grep(NeededPCA[,1], pattern = 1)
107 points(OneThree[typea,2:3],col="orange")
108
109 plot(NeededPCA[,3:4])
110 typea<-grep(NeededPCA[,1], pattern = 1)
111 points(NeededPCA[typea,3:4],col="orange")
112
113
114
115 trainingID<-sample(1:800,400)
116 testID<-sample(1:800)[-trainingID]
117 svmmodel<-svm(x[trainingID,],y[trainingID])
118 pred<-predict(svmmodel,x)
119 mean((as.numeric(pred > 0.5) == y))
120 mean((as.numeric(pred > 0.5) == y)[trainingID])
121 mean((as.numeric(pred > 0.5) == y)[testID])
122
123
124 ## ROC curve (see lecture 12)
125 start_time <- Sys.time()
126 storeme<-0
127 for (i in 1:50) {
128 trainingID<-sample(1:800,400)
129 testID<-sample(1:800)[-trainingID]
130 svmmodel<-svm(x[trainingID,],y[trainingID])
131 pred<-predict(svmmodel,x)
132 roc <- function(y, pred) {
133   alpha <- quantile(pred, seq(0,1,by=0.01))
134   N <- length(alpha)
```

```
  sens <- rep(NA,N)
  spec <- rep(NA,N)
  for (i in 1:N) {
    predClass <- as.numeric(pred >= alpha[i])
    sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
    spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
  }
  return(list(fpr=1- spec, tpr=sens))
}

r <- roc(y[testID], pred[testID])
plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")

# auc
auc <- function(r) {
  sum((r$fpr) * diff(c(0,r$tpr)))
}
glmAuc <- auc(r)
glmAuc
storeme<-storeme+glmAuc
}
storeme/50
end_time <- Sys.time()
end_time - start_time
```

RandomForest Code

```
start_time <- Sys.time()
storeme<-0
ttp<-0
ttn<-0
tfp<-0
tfn<-0
for (i in 1:50) {
    trainingID<-sample(1:800,400)
    testID<-sample(1:800)[-trainingID]
    rf_classifier = randomForest(y ~ ., data=NeededPCA[trainingID,], ntree=1000,
    mtry=14, importance=TRUE)
    treepred<-predict(rf_classifier,x)
    tp<-sum((round(treepred)==1)*(y==1))
    tn<-sum((round(treepred)==0)*(y==0))
    fp<-sum((round(treepred)==1)*(y==0))
    fn<-sum((round(treepred)==0)*(y==1))

    roc <- function(y, treepred) {
        alpha <- quantile(treepred, seq(0,1,by=0.01))
        N <- length(alpha)

        sens <- rep(NA,N)
        spec <- rep(NA,N)
        for (i in 1:N) {
            predClass <- as.numeric(treepred >= alpha[i])
            sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
            spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
        }
        return(list(fpr=1- spec, tpr=sens))
    }

    r <- roc(y[testID], treepred[testID])
    plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
    abline(0,1,lty="dashed")

    # auc
    auc <- function(r) {
        sum((r$fpr) * diff(c(0,r$tpr)))
    }
```

```
    glmAuc <- auc(r)
    glmAuc
    storeme<-storeme+glmAuc
    ttp<-ttp+tp
    ttn<-ttn+tn
    tfp<-tfp+fp
    tfn<-tfn+fn
}
storeme/50
end_time <- Sys.time()
end_time - start_time
misrate<-(tfp+tfn)/40000
sensitivity<-ttp/(ttp+tfn)
specificity<-ttn/(ttn+tfp)
```

Neural Net Code

```
start_time <- Sys.time()
storeme<-0
for (i in 1:50) {
    trainingID<-sample(1:800,400)
    testID<-sample(1:800)[-trainingID]
    nn <- neuralnet(y ~ ., data=NeededPCA[trainingID,], hidden=c(60,25),
    linear.output=FALSE, threshold=0.01)
    nn.results <- compute(nn, x)
    results <- data.frame(actual = y, prediction = nn.results$net.result)
    ynn<-nn.results$net.result
    mean((as.numeric(ynn > 0.5) == y)[testID])


    roc <- function(y, ynn) {
        alpha <- quantile(ynn, seq(0,1,by=0.01))
        N <- length(alpha)

        sens <- rep(NA,N)
        spec <- rep(NA,N)
        for (i in 1:N) {
            predClass <- as.numeric(ynn >= alpha[i])
            sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
            spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
        }
        return(list(fpr=1- spec, tpr=sens))
    }

    r <- roc(y[testID], ynn[testID])
    plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
    abline(0,1,lty="dashed")

    # auc
    auc <- function(r) {
        sum((r$fpr) * diff(c(0,r$tpr)))
    }
    glmAuc <- auc(r)
    glmAuc
    storeme<-storeme+glmAuc

}
storeme/50
end_time <- Sys.time()
end_time - start_time
```

Logistic Regression Code

```
modelcompare<-glm(y~., data = frame,family=binomial(link=logit),subset =
trainingID,maxit=200)
predb<-predict(modelcompare, framex,type="response")


start_time <- Sys.time()
framex <- as.data.frame(x[,1:396])
framey<-as.data.frame(y)
frame<-cbind(framex,framey)
start_time <- Sys.time()
storeme<-0
for (i in 1:50) {
    trainingID<-sample(1:800,400)
    testID<-sample(1:800)[-trainingID]
    RegModel<-glm(y~.,data = frame,family=binomial, subset = trainingID,maxit=200)
    framex<-as.matrix(framex)
    pred <- 1 / (1 + exp(-1 * cbind(1,framex) %*% coef(RegModel)))
    mean((as.numeric(pred > 0.5) == y)[testID])


    roc <- function(y, pred) {
        alpha <- quantile(pred, seq(0,1,by=0.01))
        N <- length(alpha)

        sens <- rep(NA,N)
        spec <- rep(NA,N)
        for (i in 1:N) {
            predClass <- as.numeric(pred >= alpha[i])
            sens[i] <- sum(predClass == 1 & y == 1) / sum(y == 1)
            spec[i] <- sum(predClass == 0 & y == 0) / sum(y == 0)
        }
        return(list(fpr=1- spec, tpr=sens))
    }

    r <- roc(y[testID], pred[testID])
    plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
    abline(0,1,lty="dashed")

    # auc
    auc <- function(r) {
        sum((r$fpr) * diff(c(0,r$tpr)))
    }
    glmAuc <- auc(r)
    glmAuc
    storeme<-storeme+glmAuc

}
storeme/50
end_time <- Sys.time()
end_time - start_time
```

Plotting

```
plot(cumsum(compressedpca$sdev^2 / sum(compressedpca$sdev^2)), type="b", xlab="PCAs
Used", ylab="Amount of Variation Explained")
cumsum(compressedpca$sdev^2 / sum(compressedpca$sdev^2))
```

Image comparision

```
numbers<-(1:800)
results<-cbind(treepred[testID],y[testID])
```