Analyzing the Columbia Dataset for Authentication of Passive-Blind Images

Ammar Mughal

MATH 3333 - Data Analytics

York University, Toronto

Abstract

Due to the rising program of Passive-blind image authentication in the world of verifying the legitimacy of an image, the need for an appropriate dataset has become crucial. In answer to this issue, the Dataset of Columbia Photorealistic Computer Graphics and Photographic Images is accessible to researchers in this domain. The data is made up of four sets of images: Recaptured Computer Graphics, Photorealistic Computer Graphics, Personal Photographic Images and Google Images. It is specifically designed for researchers focusing on distinguishing between photographic images and photorealistic computer graphics, which is an underlying problem behind the research for the authentication of passive-blind image. This report analyzes the dataset using two different machine-learning models. In the end, the report can help in determining what model can be best for minimizing errors in the analysis of the dataset.

Introduction

Digital watermarking and authentication signatures have both been important techniques for image authentication. Both methods embed a known signal into an image for authentication. However, a recent development in passive-blind image authentication eliminates the need for prior image knowledge, proving valuable in scenarios where active authentication is impractical, such as in image forensics. The performance of such techniques can vary significantly based on the dataset used. While various image datasets exist for different types of image processing research, reusing them for passive-blind image authentication research presents challenges due to the need for both authentic and fake images. Responding to this need, the Columbia Image Splicing Detection Evaluation Dataset was previously released. Another challenge in the authentication of passive-blind image is the organization of photorealistic computer graphics (PRCG) and photographic images (PIM). Addressing this requires datasets containing high-quality PRCG and diverse PIM from reliable sources along with various image acquirement factors. To fill this gap, the Columbia Dataset has been collected and offered to the community for research. In this report we use machine learning models in r-studio to break down the dataset into categories that help us selectively extract subsets for various research purposes. Thus we can eliminate part of the dataset not needed.

Data

In general CPIPCGD contains four sets: 800 Internet PRCG, 1200 Personal PIM, 800 Google PIM and eight hundred Photographed PRCG. The reasoning behind these different sets is to have diverse material in terms of factors such as lighting, cameras, sources etc. This is defined in the PIM sets as the Google set is known for its diversity, types of cameras, unique photographers etc. but lacks reliable sources; for which the Personal set makes up. For our analysis we will be focusing on the Personal PIM data-subset. Furthermore, the Personal PIM set can be broken into the following two: 800 Personal Columbia and 400 Personal Greenspun. This further distinction is due to the same reason mentioned earlier of diversity in terms of content. And so for our analysis we will be specifically using the 800 Personal PIM collection. Moreover, the Personal PIM collection is designed to have diversity in the following: Illumination/lighting, Object, Camera, Photographer. The Illumination source can be outdoor-dawn-dusk, outdoor-night, outdoor-day, outdoor-rain, indoor-dark and indoor-light. The object can be either natural or artificial. For the camera category the images can be taken from either Canon 10D or Nikon D70. Lastly, we have three photographers: Martin, Jessie, and Tian-Tsong.

Methodology

We will be using R-studio software to perform our analysis. In R, the two methods of analysis utilized are the basic logistic regression and the randomForest method. For the basic logistic regression we first import our dataset in the JPEG-version using the JPEG package in R. We also import the meta-information as an excel file to acquire data such as lighting category, camera model, file-name and photographer. Once the dataset is imported, we identify our variables and predictors- which are to be the three medians of the pixel intensities. Finally, we execute the code by acquiring the glm model, mean values and plotting the ROC curve. The outcome is to determine whether an image is outdoor. For our second method, we use the randomForest package which constructs basically a tree diagram in binary-like fashion to determine a similar outcome as our previous model. At each node in this tree diagram a distinct decision tree defines the split on the basis of a random selection of features (*Source: Lecture 21 Slides*). Ultimately, the class with the highest votes is returned as the result.

Analysis and Model

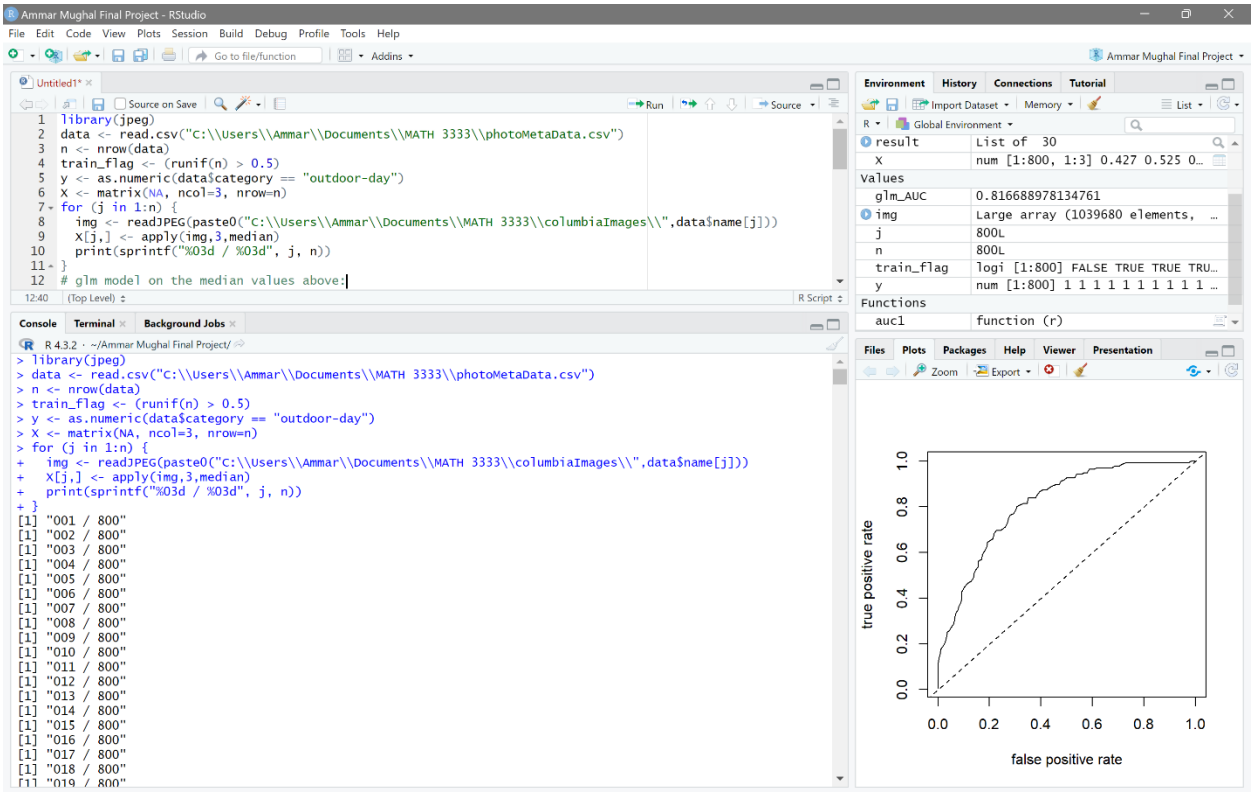
Method 1-Basic Logistic Regression

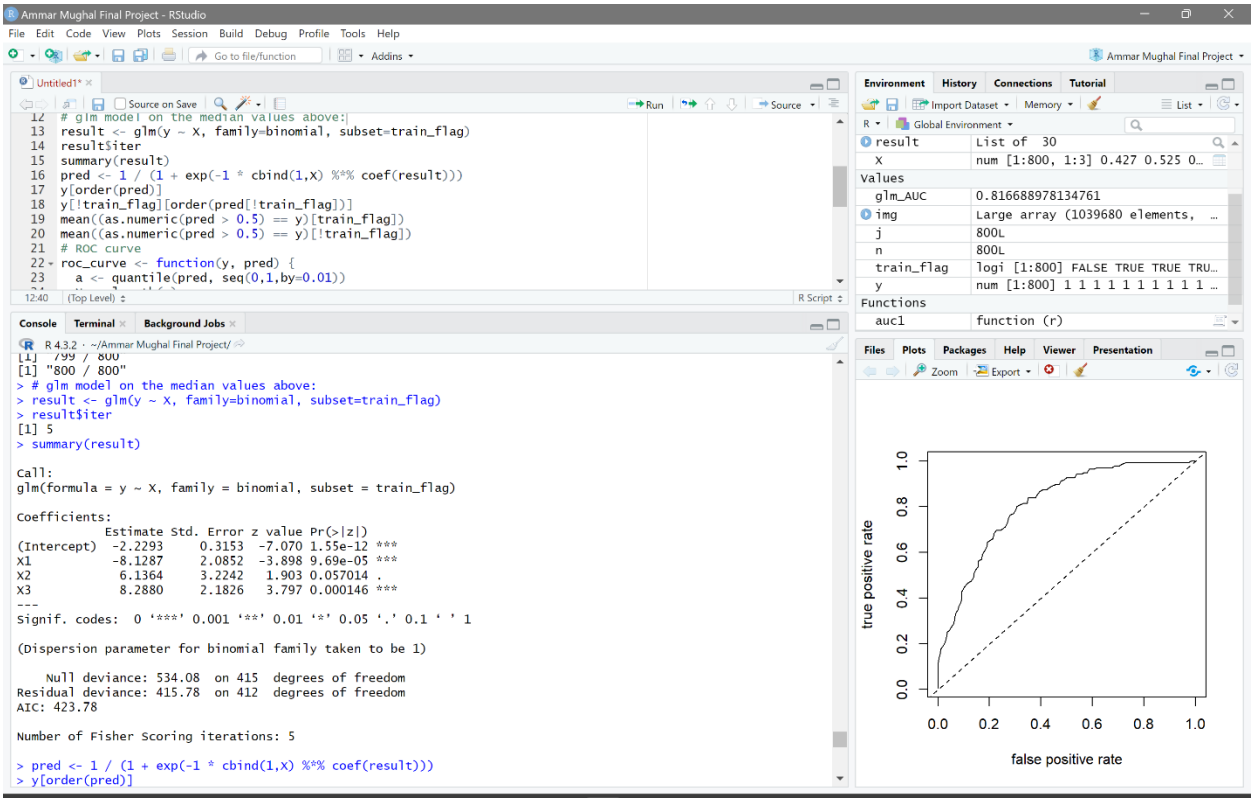
```
library(jpeg)
data <- read.csv("C:\\Users\\Ammar\\Documents\\MATH
3333\\photoMetaData.csv")
n <- nrow(data)
train_flag <- (runif(n) > 0.5)
y <- as.numeric(data$category == "outdoor-day")
X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n) {
  img <- readJPEG(paste0("C:\\Users\\Ammar\\Documents\\MATH
3333\\columbiaImages\\",data$name[j]))
  X[j,] <- apply(img,3,median)
  print(sprintf("%03d / %03d", j, n))
}
# glm model on the median values above:
result <- glm(y ~ X, family=binomial, subset=train_flag)
result$iter
summary(result)
pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(result)))
y[order(pred)]
y[!train_flag][order(pred[!train_flag])]
mean((as.numeric(pred > 0.5) == y)[train_flag])
mean((as.numeric(pred > 0.5) == y)[!train_flag])
# ROC curve
roc_curve <- function(y, pred) {
  a <- quantile(pred, seq(0,1,by=0.01))
  N <- length(a)
  sens <- rep(NA,N)
  spec <- rep(NA,N)
  for (i in 1:N) {
    predClass <- as.numeric(pred >= a[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))
}
```

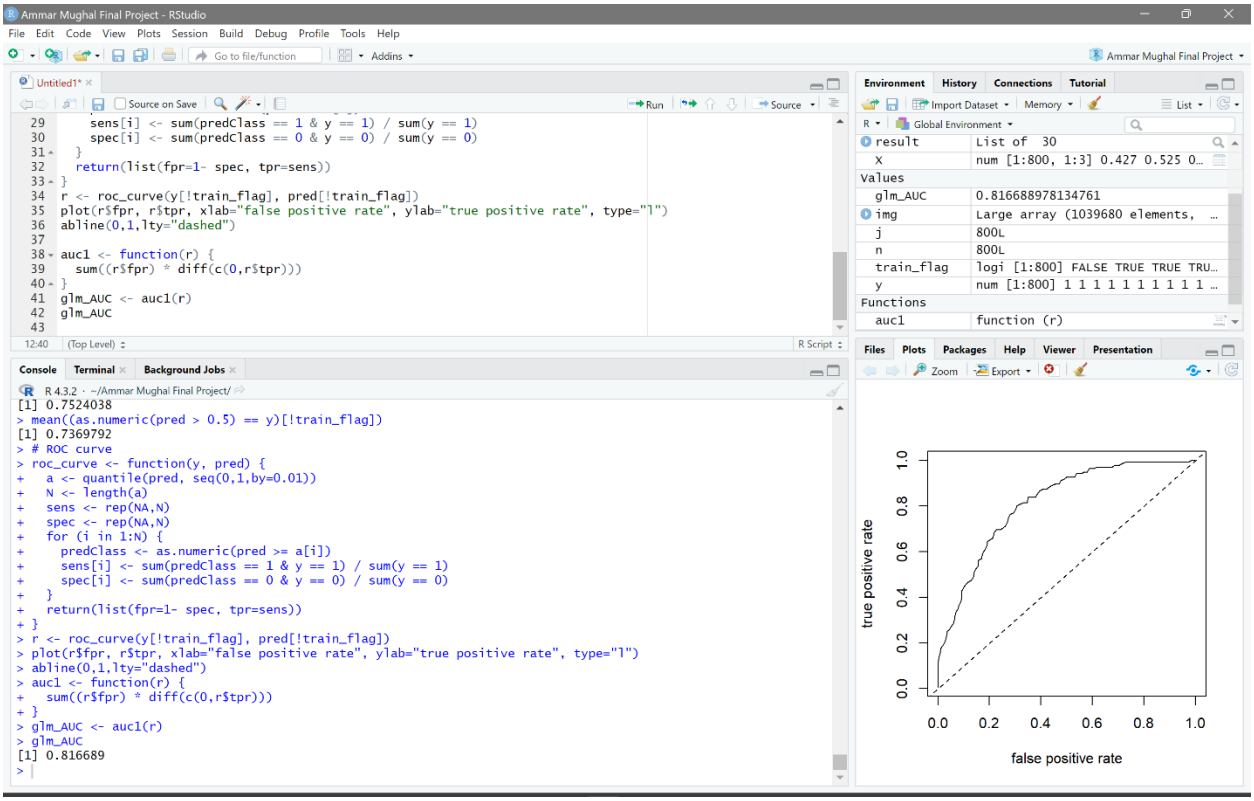
```
r <- roc_curve(y[!train_flag], pred[!train_flag])
plot(r$fpr, r$tpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")

auc1 <- function(r) {
  sum((r$fpr) * diff(c(0,r$tpr)))
}
glm_AUC <- auc1(r)
glm_AUC

> mean(sens)
[1] 0.7019695
> mean(spec)
[1] 0.6069515
```

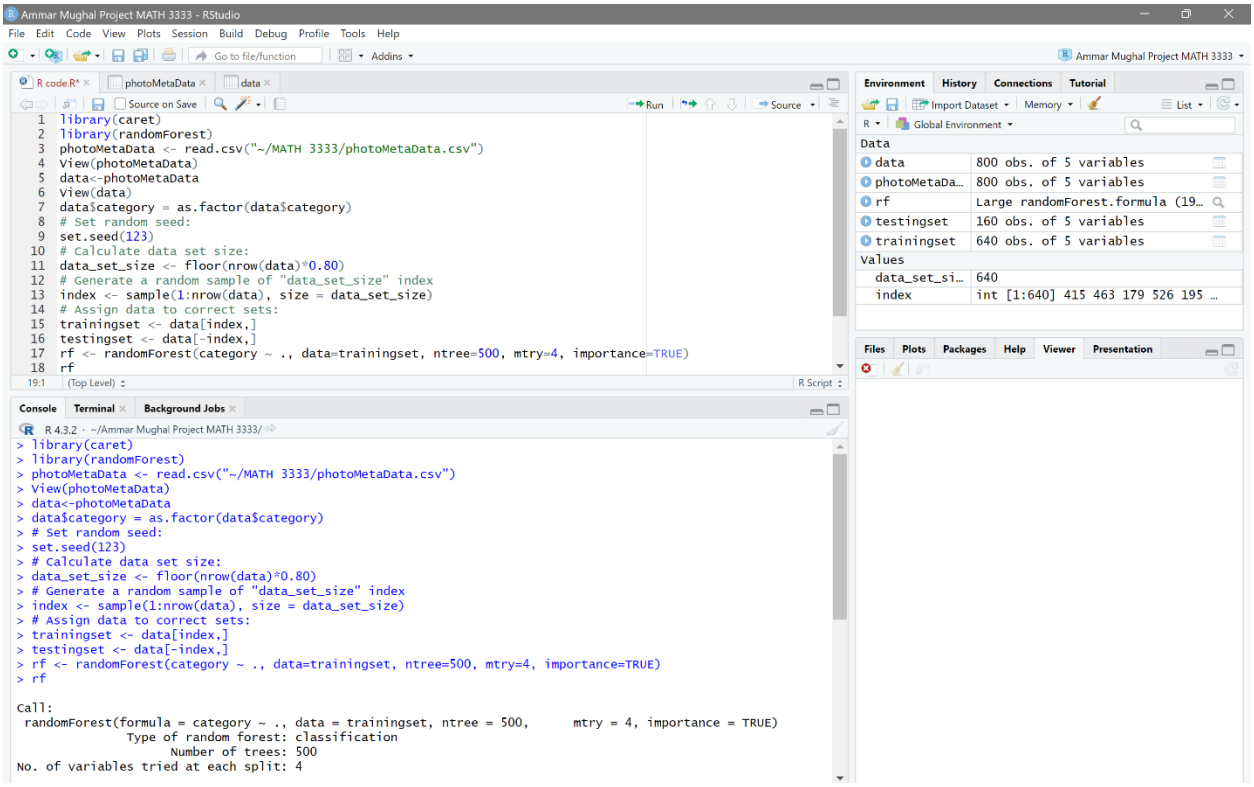


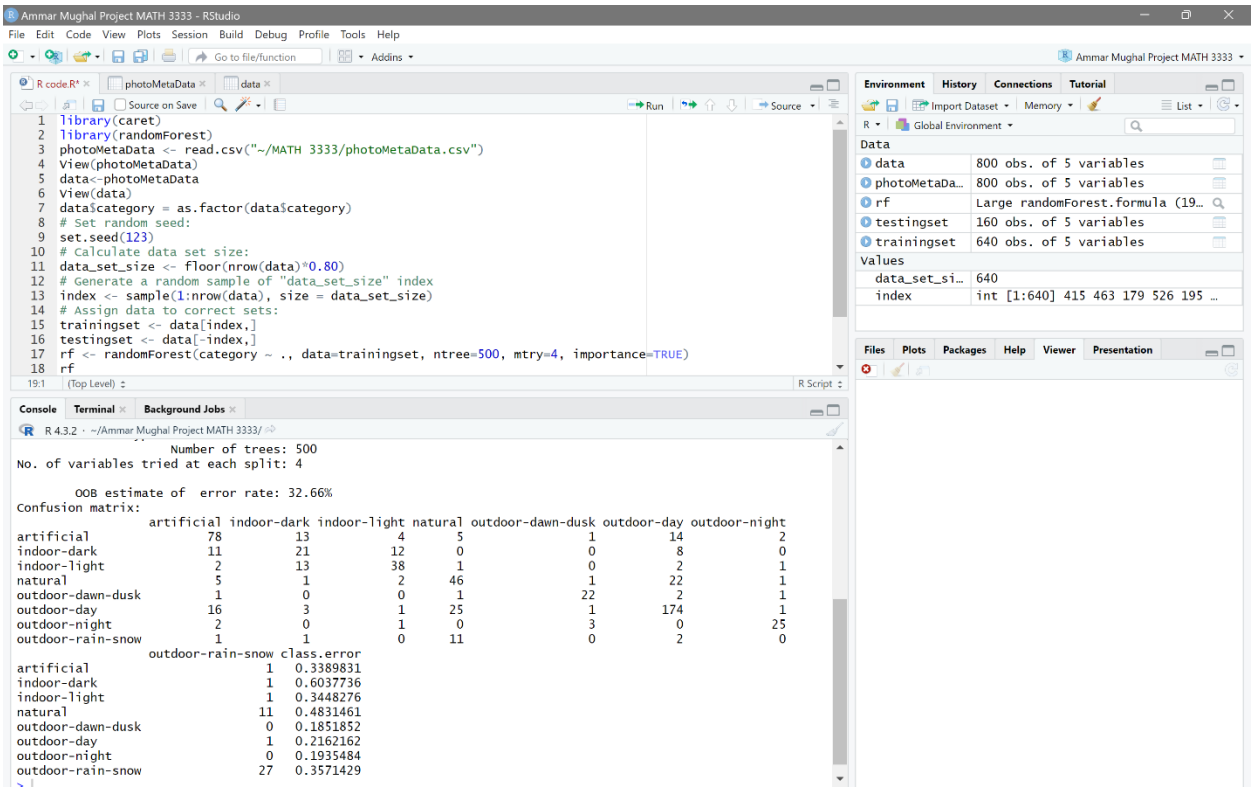

Ammar Mughal Final Project - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
# glm model on the median values above:
result <- glm(y ~ X, family=binomial, subset=train_flag)
result$iter
summary(result)
pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(result)))
y[order(pred)]
y[!train_flag][order(pred[!train_flag])]
mean((as.numeric(pred > 0.5) == y)[train_flag])
mean((as.numeric(pred > 0.5) == y)[!train_flag])
# ROC curve
roc_curve <- function(y, pred) {
a <- quantile(pred, seq(0,1,by=0.01))
Console Terminal Background Jobs
R 4.3.2 · ~/Ammar Mughal Final Project/
[1] "800 / 800"
> # glm model on the median values above:
> result <- glm(y ~ X, family=binomial, subset=train_flag)
> result$iter
[1] 5
> summary(result)
Call:
glm(formula = y ~ X, family = binomial, subset = train_flag)
Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.2293 0.3153 -7.070 1.55e-12 ***
X1 -8.1287 2.0852 -3.898 9.69e-05 ***
X2 6.1364 3.2242 1.903 0.057014 .
X3 8.2880 2.1826 3.797 0.000146 ***
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 534.08 on 415 degrees of freedom
Residual deviance: 415.78 on 412 degrees of freedom
AIC: 423.78
Number of Fisher Scoring iterations: 5
> pred <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(result)))
> y[order(pred)]
Environment History Connections Tutorial
Import Dataset Memory List
Global Environment
result List of 30
X num [1:800, 1:3] 0.427 0.525 0...
Values
glm_AUC 0.816688978134761
img Large array (1039680 elements, ...
j 800L
n 800L
train_flag logi [1:800] FALSE TRUE TRUE TRU...
y num [1:800] 1 1 1 1 1 1 1 1 1 1 ...
Functions
auc1 function (r)
Files Plots Packages Help Viewer Presentation
Zoom Export
true positive rate
false positive rate
0.0 0.2 0.4 0.6 0.8 1.0


Ammar Mughal Final Project - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
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_curve(y[!train_flag], pred[!train_flag])
plot(rfpr, rtpr, xlab="false positive rate", ylab="true positive rate", type="l")
abline(0,1,lty="dashed")
auc1 <- function(r) {
sum((r$fpr) * diff(c(0,r$tpr)))
}
glm_AUC <- auc1(r)
glm_AUC
Console Terminal Background Jobs
R 4.3.2 · ~/Ammar Mughal Final Project/
[1] 0.7524038
> mean((as.numeric(pred > 0.5) == y)[!train_flag])
[1] 0.7369792
> # ROC curve
> roc_curve <- function(y, pred) {
+ a <- quantile(pred, seq(0,1,by=0.01))
+ N <- length(a)
+ sens <- rep(NA,N)
+ spec <- rep(NA,N)
+ for (i in 1:N) {
+ predclass <- as.numeric(pred >= a[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_curve(y[!train_flag], pred[!train_flag])
> plot(rfpr, rtpr, xlab="false positive rate", ylab="true positive rate", type="l")
> abline(0,1,lty="dashed")
> auc1 <- function(r) {
+ sum((r$fpr) * diff(c(0,r$tpr)))
+ }
> glm_AUC <- auc1(r)
> glm_AUC
[1] 0.816689
Environment History Connections Tutorial
Import Dataset Memory List
Global Environment
result List of 30
X num [1:800, 1:3] 0.427 0.525 0...
Values
glm_AUC 0.816688978134761
img Large array (1039680 elements, ...
j 800L
n 800L
train_flag logi [1:800] FALSE TRUE TRUE TRU...
y num [1:800] 1 1 1 1 1 1 1 1 1 1 ...
Functions
auc1 function (r)
Files Plots Packages Help Viewer Presentation
Zoom Export
true positive rate
false positive rate
0.0 0.2 0.4 0.6 0.8 1.0

Method 2-Random Forest:

```
library(caret)
library(randomForest)
photoMetaData <- read.csv("~/MATH 3333/photoMetaData.csv")
View(photoMetaData)
data<-photoMetaData
View(data)
data$category = as.factor(data$category)
# Set random seed:
set.seed(123)
# Calculate data set size:
data_set_size <- floor(nrow(data)*0.80)
# Generate a random sample of "data_set_size" index
index <- sample(1:nrow(data), size = data_set_size)
# Assign data to correct sets:
trainingset <- data[index,]
testingset <- data[-index,]
rf <- randomForest(category ~ ., data=trainingset, ntree=500, mtry=4,
importance=TRUE)
rf
```





Result

The result for the basic logistic regression gave us a glm(auc) value of approximately 0.777. Which means that the function of Area Under Curve (AUC) over the ROC curve plot (r) is equal to 0.777. Here the r-plot is a graph that plots a model's true-positive rate (tpr = Sensitivity) against its false positive rate (fpr =1-Specificity). The mean of our specificity is 0.607. The mean for our sensitivity is 0.702. Going back to our glm(auc) value of 0.817=81.7% – this is the accuracy for this model; which seems to do just all right as the error rate is 1-0.817=0.183=18.3%. For our second method, random forest model, we were able tot obtain an Out-of-Bag (OOB) error estimate of 32.66%.

This means that 100-32.66 = 67.34% is our accuracy for this model (67.34% of the OOB samples were correctly classified by the random forest method). This was based on a training set of 640 observations (80% of 800). However, the class error for outdoor-day, which is the outcome we seek from both models, is 0.216. Which gives us the success rate of 1-0.216 = 0.784 = 78.4%. Ultimately, comparing the two models we see that basic logistic regression performs slightly better in determining the desired outcome of outdoor-day with a success rate of 81.7% compared to 78.4% of the random forest method.

Conclusion

Both of the machine learning algorithms had a decent success rate of achieving the desired outcome of determining whether an image is an outdoor-day or not. Both had almost similar results with the basic logistic regression method performing slightly better. However, there is still room for improvement as the error rate in both models is almost 20%. This translates to the issue of analyzing the dataset and being able to successfully select the required subset for research purposes; which using these two methods provides us with about 80% accuracy. With some fine tuning of the dataset, such as breaking it down into further smaller subsets that are more specific, adding more diversity, increasing the dataset size etc., and repeating the process to make more efficient predictions, these methods can give us results that are much more accurate with lower error rates. Nevertheless, so far the results do in fact seem respectable to work with and draw conclusions for research in the field of passive-blind image authentication.

References

Gao, Xin (2024). Data Analytics: A Hands-On Approach (Winter 2024) [Power Point slides] – Lectures 12, 21

Oliveira, S. P. de, posts, V. all, & Oliveira, S. P. de. (2017a, April 10). Oxford Protein Informatics Group. https://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/

Final Project Reference Paper PDF on E-class

Bhalla, D. (n.d.). How to calculate AUC (area under curve) in R. ListenData. https://www.listendata.com/2023/08/calculate-auc-in-r.html