MATH 3333: FINAL PROJECT

Justin Calcada

215473200

Prof. Xin Gao

Due Date: April 24, 2022

Table of Contents

Abstract

The goal of this paper is to develop a support vector machine classification method, or SVM for short, and compare it to how well it performs in comparison to a basic logistic regression method of classification in terms of whether is an image is an outdoor-day image or not using the images from the Personal Columbia Set of images [1]. In each method, a process was performed to randomly subset about 80% of the images into a training set and the other 20% of them into a testing set. From there, using the image category of "outdoor-day" as the response variable, and a predictor, or "x" matrix of the colour intensities in certain sections of the images was developed differently in both methods: using the median of the colour intensities in each of the 3 colour layers of the images in the basic logistic regression method—red, green, and blue and the mean colour intensities of specified blocks after cropping the images to equal sizes and orientations in the SVM method. After running the models and predictions of them in R, output of ROC curves, confusion matrices dealing with sensitivity, specificity, and accuracy of classification, and AUC values were developed as well. The main comparison measures between the two methods were the misclassification rates (29.77% for SVM and 35.66% for basic logistic regression) and the AUC values (about 83.3% for SVM and 81.8% for basic logistic regression). Overall, the SVM model and method faired better in classification of the images, or was slightly more accurate than that of the basic logistic regression model and method.

Introduction

In our society today, technology has become the driving force behind almost every industrial process, lifestyle choice, or simply how individuals unlock their phones. Many innovators have found ways to use algorithms and programming techniques to take this

technology to an even deeper level and use it for image identification in millions of ways, and the same will be done in this paper. This paper will establish the use of the programming language R in order to classify a set of photographic images from the *Personal Columbia Set* of images, as classified in figure 1, to test out specific classification techniques and how well they work for purposes such as these. By the use of Support Vector Machines—SVM for short—we will attempt to classify the 800 images in this dataset by being outdoor-day images or not, while using methods such as sensitivity and specificity along with ROC curves and AUC values to decide how well the SVM method works in comparison to a basic logistic regression method. The overall goal will be to minimize classification errors and improve our accuracy in image identification with this specific set of images.

Data

Image Category for the *Personal Columbia* Set	**Count**
indoor-light	74
indoor	68
outdoor-day	277
outdoor-night	34
outdoor-rain	63
outdoor-dawn-dusk	31
natural	111
artificial	142
Total	800

Figure 1: Image count of the images in the Personal Columbia Set from [1]

The table above describes the actual classification of the images that are derived from the Personal Columbia Dataset, but the file used for analysis in this report can be found in the Appendix.

Methodology

Method Used and What It Is

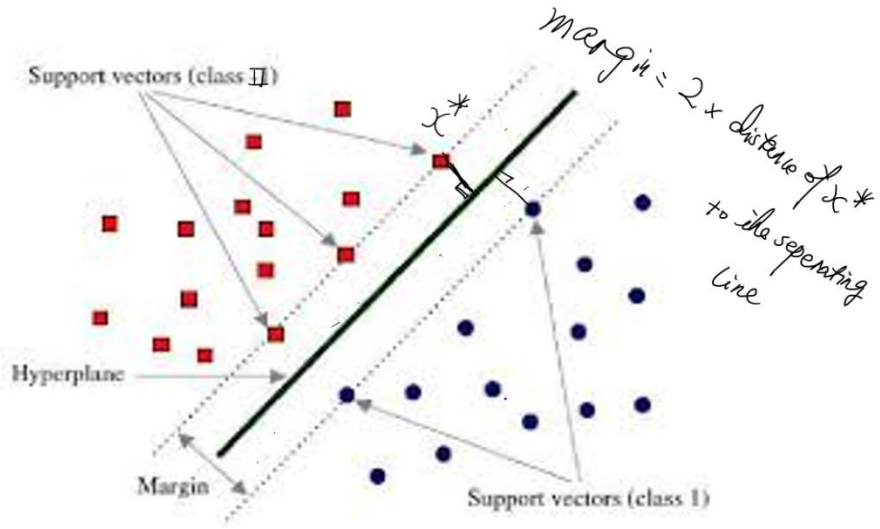


Figure 2: SVM example

Support Vector Machines are supervised machine learning algorithms that are mainly used for the purposes of classification and regression in many areas of mathematics and science [2]. They are made up of three main parts: support vectors, a separating plane with hyperplanes, and a margin [2]. Referring to figure 2, each of these components work in harmony to separate datapoints into specific groups, or classes, based on feature vectors, or predictors, from the dataset, with the goal of maximizing the distance between them for the clearest distinction possible [4]. To establish this result, a separating line is developed that aims to divide the datapoints into distinct classes—two separate groups in this case: outdoor-day images and other images [3]. Given this line, the closest point(s) from each class to it are called "support vectors", which set up hyperplanes for each class of points on either side of the separating line that are parallel to it and act as the boundaries between the classes of points [3]. These hyperplanes then

develop the concept of the margin, which is the gap between those lines and allows us to realize our main goal through this method: maximizing the margin [3]. Although, to note, this method can be extended to higher dimensions—referring to figure 3—and beyond linear classification—referring to figure 4—through the use of kernel functions, which would allow us to separate classes of datapoints that are not easily separable using a simple linear function, as shown in figure 3, but the ideas hold nonetheless [4] [5]. Through these concepts, the SVM method will allow us to understand how well we can analyze and separate the images from this dataset.

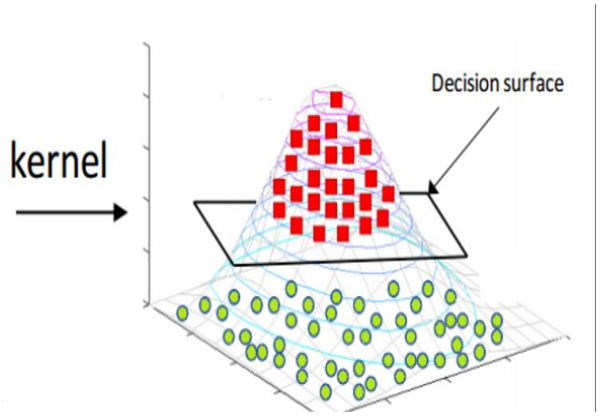


Figure 3: Use of SVM on higher dimensional data [5]

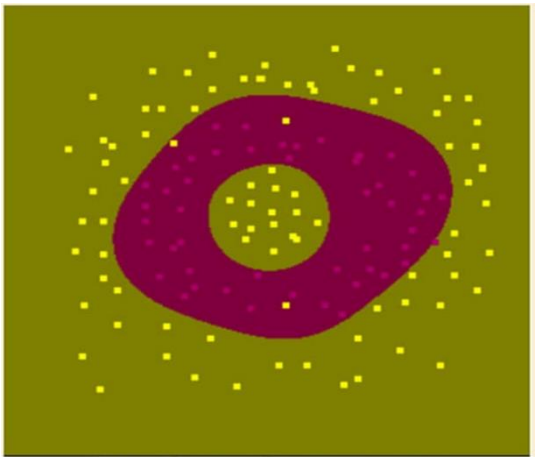


Figure 4: Use of SVM on non-linearly separable data [5]

Mathematics behind the model (from [5])

To be able to maximize the margin, we'll have to develop the equation for the separating line, which is given by:

$$g(x) = w0 + w1 * x1 + w2 * x2 = 0$$

Also known as an equation of a straight line. Given values for w0, w1, and w2 we can develop the equation for a desired line of choice based on a certain data set. Using points from a data set, we can calculate their distances to this separating line with the following distance formula:

$$Distance = \frac{|g(x)|}{\sqrt{(w1)^2 + (w2)^2}}, with\ g(x) = w0 + w1 * x1 + w2 * x2 = 0$$

Given these formulas, we take the closest point to the separating line from each class, labelled by the vector "x*" and derive the value for g(x*) from it, which scales it to be equal to 1, in absolute value, and develops the support vectors in this regard as shown by figures 5 and 6.

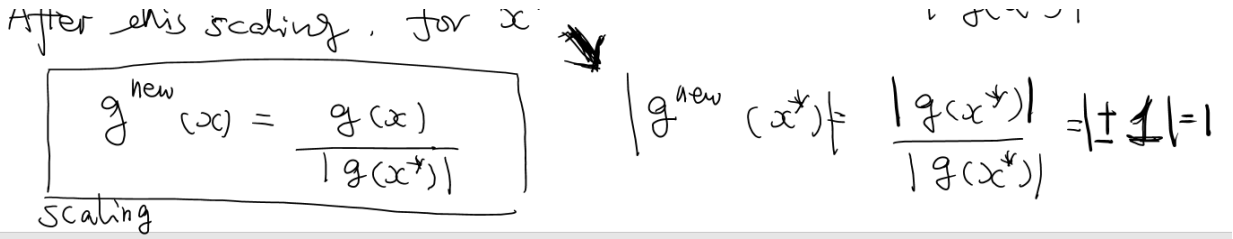


Figure 5: Scaling the g(x) equations by using the closest points to the separating line

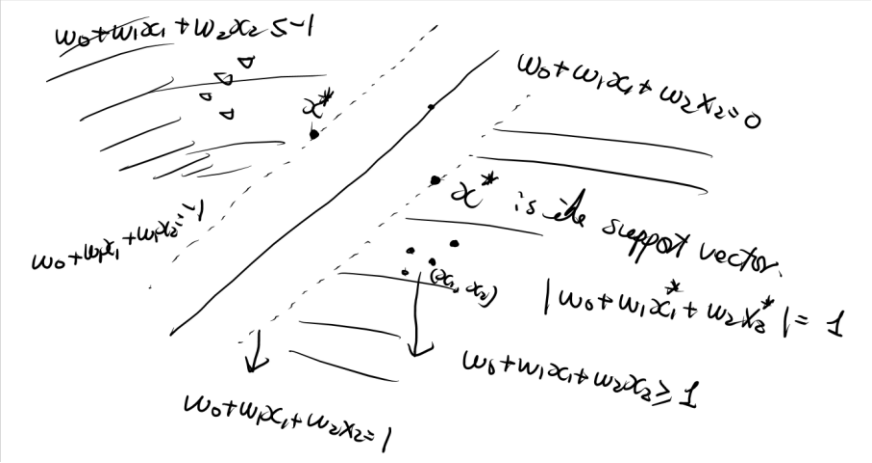


Figure 6: Separating line, its parallel hyperplanes and the classification boundaries

These lines make up the parallel hyperplanes to the separating lines and the distance between those lines are called the "margin". As a result, the margin is two times the distance of a support vector from the separating line, and we aim to maximize that margin to get the optimal separation between classes of the response variables in order to effectively classify them. Given this we have the following result:

If g(x) > 1, then that point is farther away from the separating line beyond that of the g(x*) = 1 equation and is labelled as "class 1", and if g(x) < -1, we can say that that point is farther away from the separating line beyond that of the g(x*) = -1 line and is labelled as "class 2".

These are arbitrary classification labels, but they display what SVM is attempting to do and how it is done. Moreover, from this result, we can calculate that margin discussed earlier as being twice the distance of g(x*) to the separating line, which is shown as:

$$margin = 2 * \frac{|g(x*)|}{||w*||}, where\ ||w*|| = \sqrt{(w1*)^2 + (w2*)^2}$$

The w* represents the parameters that are used to map out those hyperplanes and scale them as a result, or scale the distances from them to the separating plane as a result.

Overall, this is the preliminary mathematics that explain how SVM performs the classification, but we'll be using R and special packages to use SVM for image classification, with this being the underlying mathematics behind that.

Model and Analysis

Model Development for SVM Analysis

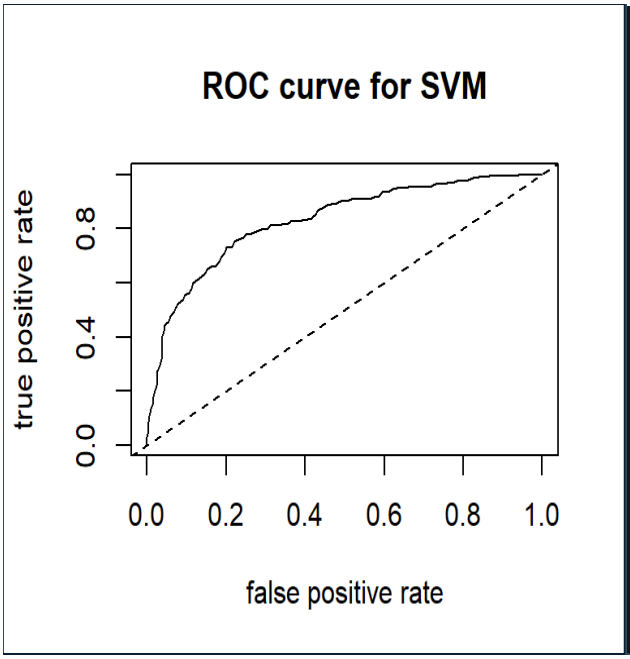
Based on the mathematics above, I developed R-code—with assistance from Professor Xin Gao—to model this SVM analysis, which can be found in the Appendix of this report. Essentially in the code, the images used in this method are cropped to dimensions of 400x700x3 pixels, with the 3 representing the layers of red, green, and blue—with some images transposed in order to make the images the same size and orientation for classification purposes. After cropping the images, the code extracts features—based on colour intensities—from the image by splitting the images into 100x100x1 pixel blocks and takes the mean of those sections, to develop 4x7x3 feature matrices of the mean colour intensities in those blocks using the necessary for loops. Once these features are extracted, they are vectorized and after running the whole process, we derive a predictor, or "x" matrix that is made up of 800 rows and 84 columns—representing the 84 mean colour intensities or "features" of the 800 images in the dataset (post-cropping). Furthermore, I randomly select images to develop a training set and a testing set of the predictor matrix and the response variable—80/20 percentage split in favour of the training set—which in this case was whether the image is an "outdoor-day" image or not. Finally, I used an SVM function found in the R-package "e1071", to perform the SVM analysis, predictions from it, and other statistical analyses that will be illustrated in the next section.

Analysis

The SVM method detailed above is used to compare to the basic logistic regression method developed by Professor Xin Gao, where the codes for that method can be found in the Appendix below. Essentially, what that basic logistic regression method code does is use the same training and testing sets from the data as was used in the SVM analysis, but here, the X matrix is derived from the median of the colour intensities of each layer—red, blue, and green—instead of from the mean of the 100x100x1 blocks above, so there are only 3 features per image rather than 84. Therefore, the SVM method refined and increased the number of features used to more accurately describe the images, or data, used in its analysis. By having both types of analysis, a comparison is done through the use of ROC curves, confusion matrices that show us statistics on accuracy of classification, sensitivity and specificity measures, and AUC values that allow us to measure the "Area Under the Curve" of the ROC plot which represents how well our method has classified the data.

Results

From the r-code described above, the following output for the SVM analysis method that I developed is illustrated below—keeping in mind that this output is based on a given randomization of the training and testing sets, so running the codes under a different randomization but similar conditions may give slightly different results than what is shown:



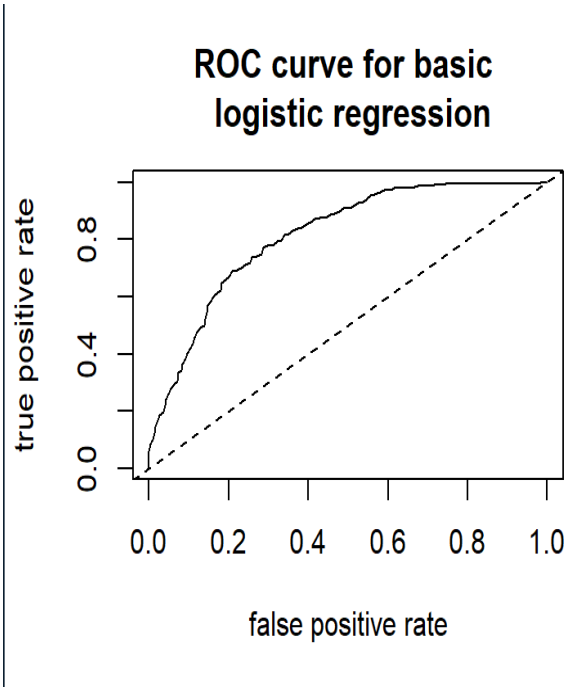

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 331  96
         1  96 122

               Accuracy : 0.7023
                 95% CI : (0.6654, 0.7374)
    No Information Rate : 0.662
    P-Value [Acc > NIR] : 0.01618

                  Kappa : 0.3348

 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 0.7752
            Specificity : 0.5596
         Pos Pred Value : 0.7752
         Neg Pred Value : 0.5596
             Prevalence : 0.6620
         Detection Rate : 0.5132
   Detection Prevalence : 0.6620
      Balanced Accuracy : 0.6674

       'Positive' Class : 0
```

```
> svmAuc
[1] 0.8332939
```

Looking at the output above we can see a few important statistics that help us decide on the performance of this analysis method. Firstly, we take a look at the ROC curve for SVM and see that it has a relatively proper shape nearly creeping up to the top left corner of the graph which is the ideal point to be at, representing a low false positive rate and a high true positive rate. The confusion matrix adjacent to it gives us the measures of accuracy, sensitivity, and specificity. The accuracy measure allows us to calculate the misclassification rate, which in this case is: 1 – Accuracy = 1 – 0.7023 = 0.2977 (29.77%). Also, we see that sensitivity, the measure of observed positive values classified correctly (1 – false negative rate)—correctly classifying an outdoor-day image—is 77.52%, and specificity, the measure of observed negative values classified correctly (1 – false positive rate)—correctly classifying an image that is not an outdoor-day image—is 55.96%. Overall, the value of AUC, shown below the ROC curve for SVM and its confusion matrix, provides a value of 83.3% which is relatively high and helps to

determine that this SVM method was fairly accurate in its classification of whether an image was an outdoor-day image or not.

Just below, a similar output under similar conditions, but with the use of the basic logistic regression method, is displayed.




```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 312 115
         1 115 103

               Accuracy : 0.6434
                 95% CI : (0.6051, 0.6804)
    No Information Rate : 0.662
    P-Value [Acc > NIR] : 0.8509

                  Kappa : 0.2032

 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.7307
            Specificity : 0.4725
         Pos Pred Value : 0.7307
         Neg Pred Value : 0.4725
             Prevalence : 0.6620
         Detection Rate : 0.4837
   Detection Prevalence : 0.6620
      Balanced Accuracy : 0.6016

       'Positive' Class : 0
```

```
> glmAuc
[1] 0.8176095
```

Looking at the ROC curve for basic logistic regression, we see that it is not as far to the top left corner as we see in that of the ROC curve for SVM. Also, the misclassification rate here (1 – Accuracy = 1 – 0.6434 = 0.3566 (35.66%)), which is much higher than the misclassification rate that we see using the SVM method. Additionally, when we take a look at the sensitivity from this method (73.07%) and the specificity (47.25%), we see that both of those are lower than what we see with the SVM output. Finally, when we look at the AUC measure for the basic

logistic regression method of classification—which is found below the ROC curve and confusion matrix for the basic logistic regression method—we see a value of about 81.8%, which is lower than that of the AUC from the SVM method.

Conclusion

Overall, after looking at the results from the outputs of each method, we see that the SVM method classifies the images more accurately not only visually from the ROC curves, but from the sensitivity, specificity, misclassification rate, and most importantly, the AUC values. As a result, the SVM method in this regard, with its further developed and refined extraction of features, turns out to be a better classifier than that of the basic logistic regression for image classification purposes using the Personal Columbia Set of images.

References

1. Tian-Tsong Ng, Shih-Fu Chang, Jessie Hsu, Martin Pepeljugoski∗.“Columbia Photographic Images and Photorealistic Computer Graphics Dataset”.

(Found from Eclass: https://eclass.yorku.ca/pluginfile.php/3514517/mod_folder/content/0/finalprojectreferencepaper.pdf?forcedownload=1)

2. “Support Vector Machine (SVM).” *Www.tutorialspoint.com*, www.tutorialspoint.com/machine_learning_with_python/classification_algorithms_support_vector_machine.htm#:~:text=%20Support%20Vector%20Machine%20%28SVM%29%20%201%20Introduction. Accessed 20 Apr. 2022.

3. Srivastava, Durgesh, and Lekha Bhambhu. *DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE*. www.jatit.org/volumes/research-papers/Vol12No1/1Vol12No1.pdf?msclkid=36437d05b39911ec9a13e85cd42324dc. Accessed 20 Apr. 2022.

4. Huang, Shujun, et al. “Applications of Support Vector Machine (SVM) Learning in Cancer Genomics.” *Cancer Genomics & Proteomics*, vol. 15, no. 1, Jan. 2018, https://doi.org/10.21873/cgp.20063.

5. Course notes and slides

For sensitivity, specificity, and ROC curve information:

https://eclass.yorku.ca/pluginfile.php/3456627/mod_folder/content/0/Lecture-on-R-logistic-regression-1.pptx?forcedownload=1

For SVM information:

https://eclass.yorku.ca/pluginfile.php/3536658/mod_folder/content/0/R-on-SVM-analysis-2.pptx?forcedownload=1

Link to code used for basic logistic regression:

https://eclass.yorku.ca/pluginfile.php/3514517/mod_folder/content/0/math3333projectsamplecode.txt?forcedownload=1

6. "Confusion Matrix in R | a Complete Guide." *JournalDev*, 10 Dec. 2020, www.journaldev.com/46732/confusion-matrix-in-r#:~:text=A%20confusion%20matrix%20in%20R%20is%20a%20table.

Appendix

Data File Used for This Project

https://eclass.yorku.ca/pluginfile.php/3514517/mod_folder/content/0/photoMetaData.csv?forcedownload=1

R Code Used for This Project

```
##' MATH 3333 FINAL PROJECT ##

#'

### Photograph examples


## Read in the library and metadata

library(jpeg)

library(caret)

pm <- read.csv("C:/Users/Justin A. Calcada/Downloads/photoMetaData.csv")

n <- nrow(pm)

set.seed(1)

trainFlag <- (runif(n) > 0.8)
```

```
y <- as.numeric(pm$category == "outdoor-day")



X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n){
  img <- readJPEG(paste0("C:\\Users\\Justin A. Calcada\\Downloads\\columbiaImages\\columbiaImages\\",pm$name[j]))
  X[j,] <- apply(img,3,median)
  print(sprintf("%03d / %03d", j, n))
}
X


# 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?
pred_log <- 1 / (1 + exp(-1 * cbind(1,X) %*% coef(out)))
y[order(pred_log)]
y[!trainFlag][order(pred_log[!trainFlag])]


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


CF_log <- confusionMatrix(factor(y[!trainFlag][order(pred_log[!trainFlag])]),factor(y[!trainFlag]))
#'
#' The misclassification rate here is given by the "1 - Accuracy", where the value
#' for Accuracy comes from the output of the Confusion Matrix labelled by CF_log.
```

```
#' Therefore, the misclassification rate here is:
#'
#' misclassification rate = 1 - Accuracy = 1 - 0.6107 = 0.3893
#'
# Using a confusion matrix to get specificity and sensitivity analysis


## ROC curve
roc_log <- function(y, pred_log) {
 alpha <- quantile(pred_log, 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_log >= 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_log <- roc_log(y[!trainFlag], pred_log[!trainFlag])
plot(r_log$fpr, r_log$tpr, xlab="false positive rate", ylab="true positive rate", type="l", main = "ROC
curve for basic \n logistic regression")
abline(0,1,lty="dashed")


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

```
}
glmAuc <- auc(r_log)
glmAuc

#########################
#########################
# SVM METHOD
library("e1071")
library(jpeg)
pm <- read.csv("C:/Users/Justin A. Calcada/Downloads/photoMetaData.csv")
n <- nrow(pm)

x<-array(dim=c(800,4*7*3))
newimage<-array(dim=c(400,700,3))
set.seed(1)
for (m in 1:800){
  print(m)
  img <- readJPEG(paste0("C:\\Users\\Justin A.
Calcada\\Downloads\\columbiaImages\\columbiaImages\\",pm$name[m]))
  if(nrow(img[,,1])>=700){
    newimage[,,1]<-t(img[,,1])[1:400,1:700]
    newimage[,,2]<-t(img[,,2])[1:400,1:700]
    newimage[,,3]<-t(img[,,3])[1:400,1:700]
  } else {

    newimage[,,1]<-(img[,,1])[1:400,1:700]
    newimage[,,2]<-(img[,,2])[1:400,1:700]
    newimage[,,3]<-(img[,,3])[1:400,1:700]
  }
```

```
 features<-array(dim=c(4,7,3))

 for (k in 1:3){
  for (i in 1:4){
   for (j in 1:7){
    # (i,j)th block

    features[i,j,k]<-mean(newimage[(100*(i - 1) + 1):(100*i),(100*(j - 1) + 1):(100*j),k])
    #print(c(i,j,k))

   }
  }
 }
 x[m,] <- as.vector(features)
 }
head(x)
y <- as.numeric(pm$category == "outdoor-day")
new_images <- cbind(y, x)

svm_model1 <- svm(y~., data = new_images[trainFlag,])
summary(svm_model1)

pred_svm <- predict(svm_model1,x[!trainFlag,])
system.time(pred_svm)
table(pred_svm,y[!trainFlag])
CF_svm <- confusionMatrix(factor(y[!trainFlag][order(pred_svm[!trainFlag])]),factor(y[!trainFlag]))
#'
#' The misclassification rate here is given by the "1 - Accuracy", where the value
#' for Accuracy comes from the output of the Confusion Matrix labelled by CF_svm.
```

```
#' Therefore, the misclassification rate here is:
#'
#' misclassification rate = 1 - Accuracy = 1 - 0.6837 = 0.3163
#'
roc_svm <- function(y, pred_svm) {
  alpha <- quantile(pred_svm, 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_svm >= 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_svm <- roc_svm(y[!trainFlag], pred_svm)
plot(r_svm$fpr, r_svm$tpr, xlab="false positive rate", ylab="true positive rate", type="l", main = "ROC curve for SVM")
abline(0,1,lty="dashed")

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

#'

#' After developing the code for SVM analysis, we see that the sensitivity and specificity

#' when using SVM are higher than when using the basic logistic regression method,

#' so is the AUC, along with the misclassification rate being lower, which establishes

#' the idea that SVM classifies the images more efficiently and accurately.

CF_log

Confusion Matrix and Statistics

#

Reference

Prediction 0 1

0 312 115

1 115 103

#

Accuracy : 0.6434

95% CI : (0.6051, 0.6804)

No Information Rate : 0.662

P-Value [Acc > NIR] : 0.8509

#

Kappa : 0.2032

#

Mcnemar's Test P-Value : 1.0000

#

Sensitivity : 0.7307

Specificity : 0.4725

Pos Pred Value : 0.7307

Neg Pred Value : 0.4725

Prevalence : 0.6620

Detection Rate : 0.4837

Detection Prevalence : 0.6620

```
#       Balanced Accuracy : 0.6016
#
#        'Positive' Class : 0
CF_svm
#
# Confusion Matrix and Statistics
#
#            Reference
# Prediction   0   1
#          0 331  96
#          1  96 122
#
# Accuracy : 0.7023
# 95% CI : (0.6654, 0.7374)
# No Information Rate : 0.662
# P-Value [Acc > NIR] : 0.01618
#
# Kappa : 0.3348
#
# Mcnemar's Test P-Value : 1.00000
#
#            Sensitivity : 0.7752
#            Specificity : 0.5596
#         Pos Pred Value : 0.7752
#         Neg Pred Value : 0.5596
#             Prevalence : 0.6620
#         Detection Rate : 0.5132
#    Detection Prevalence : 0.6620
#       Balanced Accuracy : 0.6674
```

#

'Positive' Class : 0

glmAuc

> glmAuc

[1] 0.8176095

svmAuc

> svmAuc

[1] 0.8332939