Image Classification for Indoor vs Outdoor
Math 3333 Final Project

Chen Yutong
student no:216245474

April 23 2021

1 Abstract

Many research has been done for image classification problems[5] and with the new development of machine learning techniques, there is more and more remarkable development we can achieve in the solution to better classify images. Through analysing the reference paper from Columbia Photographic Images and Photorealistic Computer Graphics Dataset,[3] we get to know some background knowledge about this 800 images in this data set. Our aim is to predict whether or not the image is showing an outdoor scene or a non-outdoor scene. The first section of the report is focusing using the sample code to predict the results using logistic regression analysis, and then the second section is focusing on trying with other machine learning improved models such as support vector machine (SVM) and random forest to test the accuracy of this data set model predictions. At the end, we will access the various models by comparing and evaluating them based on accuracy rate, sensitivity, specificity, Receiver operating characteristic (ROC) curve and Area under the curve(AUC) values.

In this paper, I am trying to explain the detailed steps and codes to access these different techniques and compare with the benchmark code to weight them based on some determinants,and finding the best represented model for this data set and discuss limitation and future development of the model in greater application areas.

Keywords :image classification, logistic regression model, support vector machine (SVM), random forest

2 Introduction

Image clarifications means determine the class of the image and see which family it belongs to. Image clarification is a trendy research area since machine learning expand people's horizon.[4] There are increasing needs and applications following up for helping people without manually classifying images especially in massive amount of data sets we have today. This is because machine can

learn it and do it for human in a more effective way such as in the autonomous driving that we need a good image classification model in the driving car to identify pedestrians verse objects so that the car learns when to stop and when to drive on the roads in the real-world. Thus, many techniques and further application in this area waiting for us to open the mystery behind the scene of image clarifications to solve the real world problem.

In the previous work by researchers,machines can not interpret images as our eyes, so we need to actually compute numbers to represent these images to make the image tangible and readable by the machine. Thus, we have used color pixel intensity at a certain point to represent this image. The most well-know color coding is putting into three layers of red, blue and green (RGB)in order to analysis the color intensity of the pixel and we make a matrix to represent these numbers and then use this matrix to represent an image. [1]

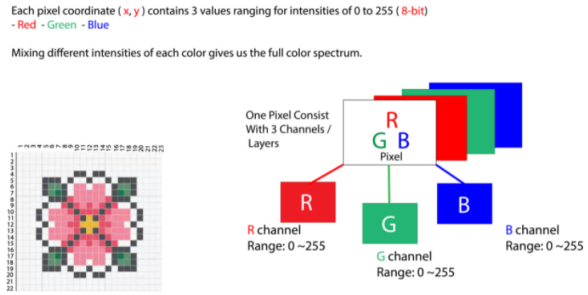

Figure 1: The concept of RGB

In my case, I choose to use SVM and random forest model to re-form the model analysis of this question because machine learning techniques are really aiming for efficiency to learn in artificial intelligence areas and known for increasing the accuracy of these digital numbers. Thus, I want to use it in this image classification problem to improve the model to justify which technique has the most improved model with highest accuracy.

3 Data

The data for these 800 hundreds pictures is extracted from the data set mentioned in Columbia Photographic Images and Photorealistic Computer Graphics Dataset[3].

These photos are collected under PIM images from the personal collections taken using professional camera lens of single-len-reflex (SLR) Canon 10D and Nikon D70 and taken by three photographers named as 'martin', 'jessie' and 'tian-tsong' in the data set. [3] There are total 8 categories for the photos taken including "Indoor-light", "Indoor-dark", "Outdoor-rain", "Outdoor-night", "Outdoor-day", "Outdoor-dusk", "Natural-obj" and "Artificial-obj".[3] However, for the simplicity of this project, we only taking into account for two

Figure 2: Example of outdoor image in the data set

Figure 3: Example of non-outdoor image in the data set

classes to do a binary image classification. The first class is classified as "outdoor" scenes and the second one is classified as "non-outdoor" scenes. Some examples of outdoor verse non-outdoor pictures are shown above.

Then we read the csv file in the data set that contains name of the photo, category it belongs to according to the 8 original categories, camera used to took these pictures, location of the photos taken and the name of the photographer who took the photo. And store this data set as named "pm" in the R studio environment. Then, we set the number of rows of this "pm" to be n which turns out to be 800 because we have 800 images and every row record the information regarding to that one image.

Then, besides classifying the classes, we also clear the data set to creates subsets and naming them according to our needs in order to running the model later on. We set the result for the prediction on outdoor or non-outdoor as our "y" in the data set. And because in the csv file, this category column is written in words, we need set "y" as numerical numbers with "1" indicates outdoor and "0" indicates non-outdoor.

On the other side, we also need to generate the "x" in this data set.

Firstly, we use uniform function to generate the training ID with a set of uniform random numbers and we need to "set.seed"(so we always generate the same training ID) for this data set because we want to compared these models later on based on same data set in order to make a fair comparison.

Then we want to generate a matrix X with 3 column and n rows so it can store the pixel number information from the image. Then, for every blank space in this matrix, based on the benchmark code method, we want to find the median for the number of the pixel among red, blue and green intensity to fill in every row about information of this image listed. This means in the first row of "X", the first column indicated the median of red pixel intensity of image one, then the second column indicated the median of green pixel intensity of image one, then the third column indicated the median of blue pixel intensity of image one. And the row number of the matrix "x" will corresponding to the image number and so on.

4 Methodology

In this whole project, we have make use of different model concept and some determinants to test the model. Here are the more detailed explanation here.

4.1 Logistic Regression Model

For the first one which based on the sample code, we are using logistic regression model to predict the outcome of whether the image predicted as outdoor or non-outdoor.

The response variable which is the "y" in logistic regression is binary interpreted. This means it can only take "1" as success and "0" as failure. In our case, "1" represents outdoor and "0" represent as non-outdoor. Interpreting the coefficients of the logistic regression model is also very important because it closely involved in the calculation of probabilities which links to variable "x" in the model formula. so to sum up, the relationship of the probability is as follows: $probability, p = f(\alpha + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_k \cdot x_k) = \frac{e^{\alpha+\beta_1 \cdot x_1+\beta_2 \cdot x_2+\ldots+\beta_k \cdot x_k}}{1+e^{\alpha+\beta_1 \cdot x_1+\beta_2 \cdot x_2+\ldots+\beta_k \cdot x_k}}$ and when we simplify this equation we can see

$$\log(\frac{p}{1-p}) = \alpha + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_k \cdot x_k \tag{1}$$

4.2 Support Vector Machines (SVM)

The second model we use is Support vector machines(SVM). SVM is a way that we can use to classify both linear and non-linear data and classify them into two categories by separating training set and testing set in both the variable "x" and response "y".

For example, for a general data set with linear data, we can plot all the points from the data in the graph to indicate the position (x,y) and we can then clearly see which data point belongs to which class. Thus, same as shown in the picture, SVM is trying to find this line of separation, the separate hyper-plane that can successfully classify the two classes. And the supported vector will be along the parallel lines in between the line of separation and have the data points of each class that have the shortest distance to the line of separation. We also define margin as the distance between the two parallel lines. The way we can access the model is to see how big is this margin, if this distance is big, this means this model is good that can successfully classify the data into two classes.

Just like when two people standing together is hard to separate them, but if we shine a light and form a shadow of the two people, we can easily distinguish the two person apart. As for non-linear data set, SVM is also excellent in separating the classes just like the purpose of the shadow by transforming the data into a higher dimension such that we can see the separation from another aspect of angle and the way we analyse the data for linear model can be apply to access this non-linear data set then. A neat example is shown below.

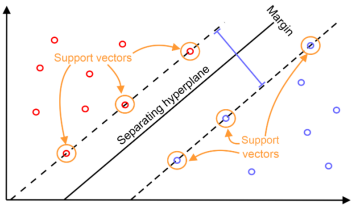

Figure 4: The concept of SVM for linear case

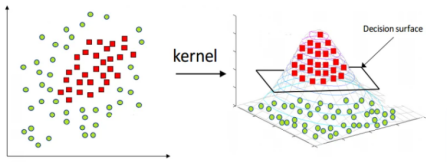

Figure 5: The concept of SVM for non-linear case

4.3 Random Forest

The third model we use is random forest. This is also a well-known machines learning algorithm that actually act as an improved version of bagged decision trees. Random forest has corrected the over-fitting problems of the decision trees due to less in similarity in the structures(sub-trees) and less co-relation rate between variables from predictions. In the random try model, m represents the split point, there is a need to specify m in this model because we need to know the number of features can be put in the split point. We can change this number together with cross validation methods.

4.4 Receiver Operating Characteristic Curve(ROC)

Besides the model, we also make use of some determinants to determine which model gives us the best accuracy rate.

A receiver operating characteristic (ROC) curve is a graph representation for displaying different confusion matrix information with different threshold set. The y axies of the ROC curve shows the true positive rate, which is also called, sensitivity, thus we have the equation for

$$TruePositiveRate = Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (2)$$

Where true positives are classified corrected as "1"s, when they are indeed outdoor images in our data. And false negative are predicted to be "0" but actual is outdoor image in our data. This true positive rate tells us the proportion of outdoor images that are correctly classified.Thus, the higher the sensitivity, the better the model.

The x-axies of the ROC shows the false positive rate, which is also same as the $1 - specificity$, thus we have the following equation to calculate the false positive rate which

$$FalsePositiveRate = (1 - Specificity) = \frac{FalsePositives}{FalsePositive + TrueNegatives} \quad (3)$$

Where the false positives are predicted to be "1" but are actually non-outdoor images, and true negatives are predicted to be "0" and indeed are

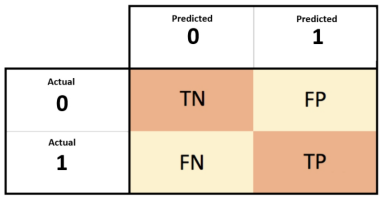

Figure 6: Example of a confusion matrix

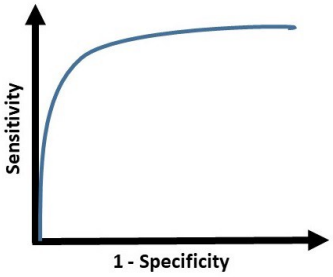

Figure 7: Example of a ROC curve

non-outdoor images. This false positives rate tells us the proportion of non-outdoor images that are incorrectly classified. If we lower the threshold, we will then increase the sensitivity, but at the same time we will also decrease the specificity, thus, it is really important that we know which threshold we can bear to evaluate the model.

4.5 Area Under The Curve(AUC)

Area under the curve is what we called it as AUC curve, which literally means the area under the curve of ROC, which gives us a easy direct way to compare the ROC curves. Thus, the bigger the value of AUC means the bigger the area under the ROC curve, the better the model used to classify this data set.

5 Analysis and Model

Based on the above concepts, we want to then build our model to test and compared our accuracy to see which model gives us the best results of predictions.

Because this is a problem with regarding to image analysis, the first thing we need to do is to install JEPG package in r studio. JPEG is used for read and write images files stored in the JPEG format to make the image readable by the machine to do data analysis.

5.1 Model with Median of RGB Color Intensity

5.1.1 Logistic Regression

The R command for "glm" is used to fit logistic regression models in analysing the data set. "glm" stands for generalized linear models, so it can used with specifying a formula used and listing the linear predictors with a specific link function. For logistic regression, binomial is the family of the error distribution. This family name also needs to specify in the "glm" model.

For the sample code, the first thing we do is to set "y" to be "1" when indicates "outdoor" and "0" when indicates "non-outdoor". Then secondly,we set "x" to be taken the median of the three color pixel intensity as explained above. Thirdly, we set training set as uniform random number more than 0.5 to be the set for training and the rest will be in the testing set.

For the logistics regression model, we use training flag as the subset and it follows the formula of $y = x$ with binomial classification as family. The results after running this model show all the coefficients of the intercept, red, blue and green color intensity in the formula and also able to see the iteration times from the data.

After that, we use equation (1) in logistic model for calculating the predictions. Then, we see the predicted results for the value in training data set and testing data set. We then calculate the mean value of the prediction in the training data set and testing data set when the predict value is more than 0.5 then we classify that set in y.

Last but not the least we use ROC curve and AUC value to see the sensitivity and specificity of the model. Record down this value so we can compare it with other models later on to see which model has the largest AUC will give the best model for fitting this data set.

5.1.2 Support Vector Machine (SVM)

Moreover, "e1071" package is needed to set up for the SVM model. We need to form some subset in order to run this model, so firstly, we set a training set for "x" and testing set for "x", training set for "y" and testing set for "y". Because here "y" need to be classes as "0" or "1", we also need to make it as factor when we set the training and testing set for "y".

In the SVM model, we use both training set for x and y in it to build the model and allow probability to be true so that we can view the probability later. We will then predict the y value in the svm model by using the training set of x in the model. We can then tabulate the result of observed y and predicted y to see how many (non-outdoor)"0" there are and how many we predict correctly to be "0", and how many (outdoor)"1" there are and we predict correctly to be "1".

we can then calculate the accuracy by summing all the diagonals of the table because it indicates we predict correctly and then divide by the total number of data in this testing set. We can also use confusion matrix function to see all the results in one step and also double checking our predicted accuracy rate by using this function. Moreover, in order to launch this function, we need to first also open the packages called "caret", and in order to open "caret", we also need to install "lattice" and "ggplot2".

Last but not the least, we can use the ROC curve and AUC value again to see the sensitivity and specificity of the SVM model. Record down this value so we can compare it with other model later on to see which model has the largest AUC will give the best model for fitting this data set.

5.1.3 Random Forest

"RandomForest" package is needed for running random forest analysis. This package makes use of Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) to classify data set.[2]

In the random forest model, same as SVM, we use both training set for x and y in it to build the model, and setting to choose 100 trees in this model and m, the split point to be 2. We can also use the "varImpPlot" function to make use of a dot-chart to access on the importance of all the variables in the random forest model.

We can then use the predict function based on the model and training set of x to predict the y , and tabulate the result of observed y and predicted y to see how many (non-outdoor)"0" there are and how many we predict correctly to be "0", and how many (outdoor)"1" there are and we predict correctly to be "1" in to a confusion matrix table.

We can use the table above to calculate the accuracy by summing all the diagonals of the table because it indicates the cases we predict correctly and then divide by the total number of data points in this testing set. We can also use confusion matrix function to see all the results in one step and also double checking our predicted accuracy rate.

We can then predict the probability of each value in the training set x in the random forest model to generate the ROC and AUC curve later on to see the sensitivity, specificity, AUC values and compare with other model to see which model has the largest AUC will give the best model for fitting this data set.

5.2 Model with Mean of RGB Color Intensity

We usually think about mean of the data in order to get a more unbiased data set, thus we want to try using mean intead of mediaN in calculating the RGB color intensity for all the model used before and see if we can further improve the model with mean function between RGB pixels.

5.2.1 Logistic Regression

We are practicing the same steps as previous model in the sample code with the change from calculating the median to mean and keep everything else the same.

5.2.2 Support Vector Machine(SVM)

We are practicing the same steps as previous model in the sample code with the change from calculating the median to mean and keep everything else the same.

5.2.3 Random Forest

We are practicing the same steps as previous model in the sample code with the change from calculating the median to mean and keep everything else the same.

Model with Median			
	Logistic regression	SVM	Random Forest
Accuracy	0.7372263	0.7712082	0.6915167
AUC	0.8120599	0.8022664	0.7589206
Model with Mean			
	Logistic regression	SVM	Random Forest
Accuracy	0.7980535	0.8071979	0.7866324
AUC	0.8633943	0.8483638	0.8301379

Figure 8: Table for comparing the models

Model with Median		
	SVM	Random Forest
Sensitivity	0.8992	0.7752
Specificity	0.5191	0.5267
Model with Mean		
	SVM	Random Forest
Sensitivity	0.8837	0.8605
Specificity	0.6565	0.6412

Figure 9: Table for comparing the models

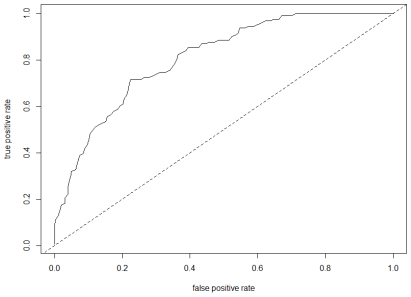

Figure 10: ROC curve for logistic regression with median model

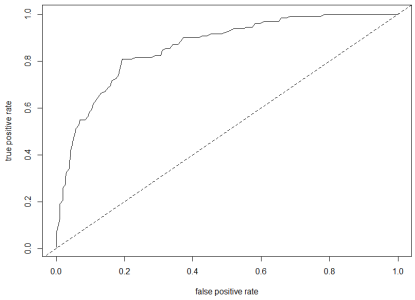

Figure 11: ROC curve for logistic regression with mean model

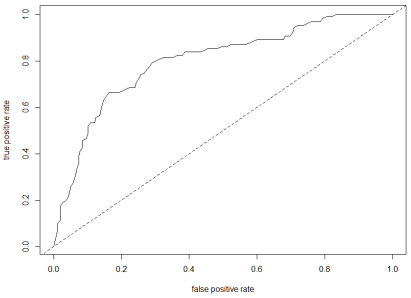

Figure 12: ROC curve for SVM with median model

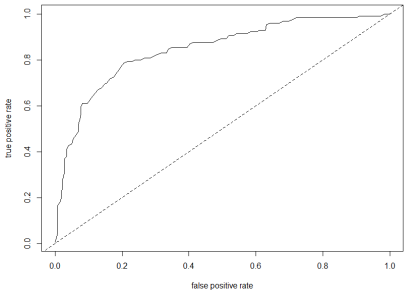

Figure 13: ROC curve for SVM with mean model

6 Result

By running these models, we have summarized the accuracy, sensitivity, specificity, the ROC and AUC in a table and graphs shown for comparison.

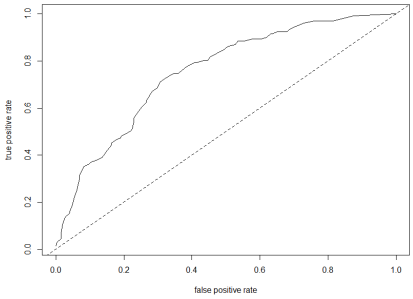

Figure 14: ROC curve for Random Forest with median model

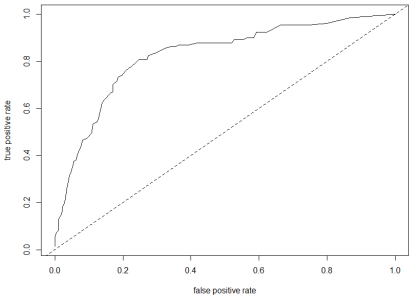

Figure 15: ROC curve for Random Forest with mean model

7 Conclusion

From above graph and tables, we can see firstly, models that are constructed by using mean of the three color intensity all have better accuracy and AUC values compared to their median models. Then, we can see although the accuracy rate for SVM is the highest but the AUC value is a little bit smaller than logistic model. This is because the accuracy rate maybe is just happened that nicely we pick a good threshold cutoff value but the AUC value represents the whole possible cutoff values. That is why it is still possible that we get this result. SVM is more famous to sort out kernel data and unstructured data set thus that is maybe the reason why it has a good accuracy rate for this image classification analysis. Generally speaking, by doing these models, we find that for this data set, setting seed to be 420, logistics model is overall better compared to using SVM model and random forest model.

However, we do still have some limitations, because currently, these 800 is still a manageable size for analysing using logistic regression, SVM or random forest, but as we will face more complex data set and it will take more time for running these data using the logistic regression and SVM codes, so there maybe an need for partition the image before we can run the model. Otherwise, r may crush and not able to go through the data and classify them.

Moreover, some consideration for future work maybe using other machine learning models such as convolutional neural network(CNN)[1],K-Nearest Neighbour(KNN) [5]as they may have even better accuracy results over a bigger size of data points. We may also consider multi-classification methods to begin with for not only just consider "0" as non-outdoor and "1" as outdoor but specify in details of the categories to make multiple classes so we have a more diverse classes that we can further classified more details in the image and in the data set.

8 Appendix

8.1 sample code

Here are the codes for logistic regression model.

```
## Read in the library and metadata
library(jpeg)
pm <- read.csv("C:\\Filesforthefinal\\photoMetaData.csv")
n <- nrow(pm)
set.seed(420)
trainFlag <- (runif(n) > 0.5) #generate uniform random numbers
y <- as.numeric(pm$category == "outdoor-day") #Convert Y Data to Numeric

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


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

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

Figure 16: Sample code for Logistic Regression model part 1

```
## ROC curve
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
```

Figure 17: Sample code for Logistic Regression model part 2

8.2 Improved code for SVM

Here are the codes for svm model.

```
library(jpeg)
pm <- read.csv("C:\\Filesforthefinal\\photoMetaData.csv")
n <- nrow(pm)
set.seed(420)
trainFlag <- (runif(n) > 0.5) #generate uniform random numbers
y <- as.numeric(pm$category == "outdoor-day")
X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n) {
  img <- readJPEG(paste0("C:\\Filesforthefinal\\columbiaImages\\columbiaImages\\",pm$name[j]))
  X[j,] <- apply(img,3,median )
  print(sprintf("%03d / %03d", j, n))
}
library(lattice)
library(ggplot2)
library(caret)
library("e1071")
  set.seed(420)
  trainFlag <- (runif(n) > 0.5)
  xtrain<-X[trainFlag,]
  xtest<-X[!trainFlag,]
  ytrain<-as.factor(y[trainFlag])
  ytest<-as.factor(y[!trainFlag])

svm_model1 <- svm(xtrain,ytrain,probability = TRUE)
summary(svm_model1)
pred <- predict(svm_model1,xtest,probability = TRUE)
pred

tsvm<-table(pred,ytest)
tsvm
accuracy <- (sum(diag(tsvm))/sum(tsvm))
accuracy
confusionMatrix(pred,ytest)
```

Figure 18: Sample code for SVM model part 1

```
## ROC curve
p<-attr(pred,"probabilities")
prob<-p[,1]

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(ytest,prob)
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
```

Figure 19: Sample code for SVM model part 2

8.3 Improved code for random forest

Here are the codes for random forest model.

```
library(jpeg)
pm <- read.csv("C:\\Filesforthefinal\\photoMetaData.csv")
n <- nrow(pm)
set.seed(420)
trainFlag <- (runif(n) > 0.5) #generate uniform randum numbers
y <- as.numeric(pm$category == "outdoor-day") #Convert H2O Data to Numeric
X <- matrix(NA, ncol=3, nrow=n)
for (j in 1:n) {
  img <- readJPEG(paste0("C:\\Filesforthefinal\\columbiaImages\\columbiaImages\\",pm$name[j]))
  X[j,] <- apply(img,3,median )
  print(sprintf("%03d / %03d", j, n))
}

library(lattice)
library(ggplot2)
library(caret)
library(randomForest)
set.seed(420)
trainFlag <- (runif(n) > 0.5)
xtrain<-X[trainFlag,]
xtest<-X[!trainFlag,]

ytrain<-as.factor(y[trainFlag])
ytest<-as.factor(y[!trainFlag])

rf_classifier = randomForest(ytrain~ ., data=xtrain, ntree=100, mtry=2, importance=TRUE)
summary(rf_classifier)
varImpPlot(rf_classifier)
pred1<-predict(rf_classifier, xtest)
rft<-table(pred1,ytest)
rft
accuracy <- (sum(diag(rft))/sum(rft))
accuracy
confusionMatrix(pred1,ytest)
```

Figure 20: Sample code for Random Forest model part 1

```
p<-predict(rf_classifier, xtest,type="prob")
p
prob<-p[,2]
prob

roc <- function(y, pred1) {
  alpha <- quantile(pred1, 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(pred1 >= 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], prob)
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
```

Figure 21: Sample code for Random Forest model part 2

References

[1] IEEE Yuliya Tarabalka Guillaume Charpiat Emmanuel Maggiori, Student Member and Pierre Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE transactions on geoscience and remote sensing*, 55(2):645–657, 2017.

[2] Fortran original by Leo Breiman, R port by Andy Liaw Adele Cutler, and Matthew Wiener. *Breiman and Cutler's Random Forests for Classification and Regression*. Package 'randomForest', 3 2018.

[3] Jessie Hsu Martin Pepeljugoski Tian-Tsong Ng, Shih-Fu Chang. Columbia photographic images and photorealistic computer graphics dataset. *Department of Electrical Engineering Columbia University*, 2(205-2004-5):1–23, 2005.

[4] IEEE Kui Jia Shenghua Gao Jiwen Lu Zinan Zeng Tsung-Han Chan, Member and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE transactions on image processing*, 24(12):5017–5032, 2015.

[5] Bingzheng Yan Zhehang Tong, Dianxi Shi and Jing Wei. A review of indoor-outdoor scene classification. *Advances in Intelligent Systems Research*, 134(410073):469–474, 2017.