Predicting skin and non skin from image data using machine learning techniques

Aurelius Ferdinand Almeida

1/9/2020

Introduction

This analysis is done for a capstone project submission as part of the Harvard Data science Professional Certification. The dataset used is the 'Skin segmentation dataset' created by Bhatt et al and obtained from the UCI Machine learning repository in the following Link. The authors created this dataset by randomly sampling B,G,R values from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database. Total learning sample size is 245057; out of which 50859 are skin samples and 194198 are non-skin samples

- ### Objectives

  The key objective of this project is to successfully train a machine learning model using this dataset and then predict if samples are skin or non skin based on B,G,R values

- ### Summary

  This report describes the review of this dataset and the initial assumptions that are made. It then explains the models used and the results that were obtained. The report concludes with notes on the project, its limitations and possible future work.

Methods and Analysis

- ### Getting Started

  Any analysis requires the right tools to be made available. In R these are the additional libraries and packages, the code chunk below will install if needed and load the required libraries

```
#Installing if necessary and loading the required packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
if(!require(mda)) install.packages("mda", repos = "http://cran.us.r-project.org")
if(!require(foreach)) install.packages("foreach", repos = "http://cran.us.r-project.org")
if(!require(MASS)) install.packages("MASS", repos = "http://cran.us.r-project.org")
```

- **Overview of the dataset**

The first step of this analysis is to obtain the data from the UCI repository. Post that it will be read and assigned to an object 'sknnon' with the correct column names. The below code will perform these steps

```
#Create an object to hold the URL address
URL <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00229/Skin_NonSkin.txt"

# Read file into object and insert column names
sknnon <- read_tsv(URL, col_names = c("B","G","R","Y"))
```

```
## Parsed with column specification:
## cols(
##   B = col_double(),
##   G = col_double(),
##   R = col_double(),
##   Y = col_double()
## )
```

Once the sknnon dataset is available the characteristics of the dataset can be viewed by using the code below

```
# View structure of the dataset
str(sknnon)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 245057 obs. of  4 variables:
##  $ B: num  74 73 72 70 70 69 70 70 76 76 ...
##  $ G: num  85 84 83 81 81 80 81 81 87 87 ...
##  $ R: num  123 122 121 119 119 118 119 119 125 125 ...
##  $ Y: num  1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   B = col_double(),
##   ..   G = col_double(),
##   ..   R = col_double(),
##   ..   Y = col_double()
##   .. )
```

The summary function will provide additional useful information regarding the dataset. The first 5 rows of the dataset can be viewed using the below code

```
#View summary of the dataset
summary(sknnon)
```

```
##        B               G               R               Y
##  Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   Min.   :1.000
##  1st Qu.: 68.0   1st Qu.: 87.0   1st Qu.: 70.0   1st Qu.:2.000
##  Median :139.0   Median :153.0   Median :128.0   Median :2.000
##  Mean   :125.1   Mean   :132.5   Mean   :123.2   Mean   :1.792
##  3rd Qu.:176.0   3rd Qu.:177.0   3rd Qu.:164.0   3rd Qu.:2.000
##  Max.   :255.0   Max.   :255.0   Max.   :255.0   Max.   :2.000
```

```
# View first 5 rows of the dataset

head(sknnon,5)
```

```
## # A tibble: 5 x 4
##       B     G     R     Y
##   <dbl> <dbl> <dbl> <dbl>
## 1    74    85   123     1
## 2    73    84   122     1
## 3    72    83   121     1
## 4    70    81   119     1
## 5    70    81   119     1
```

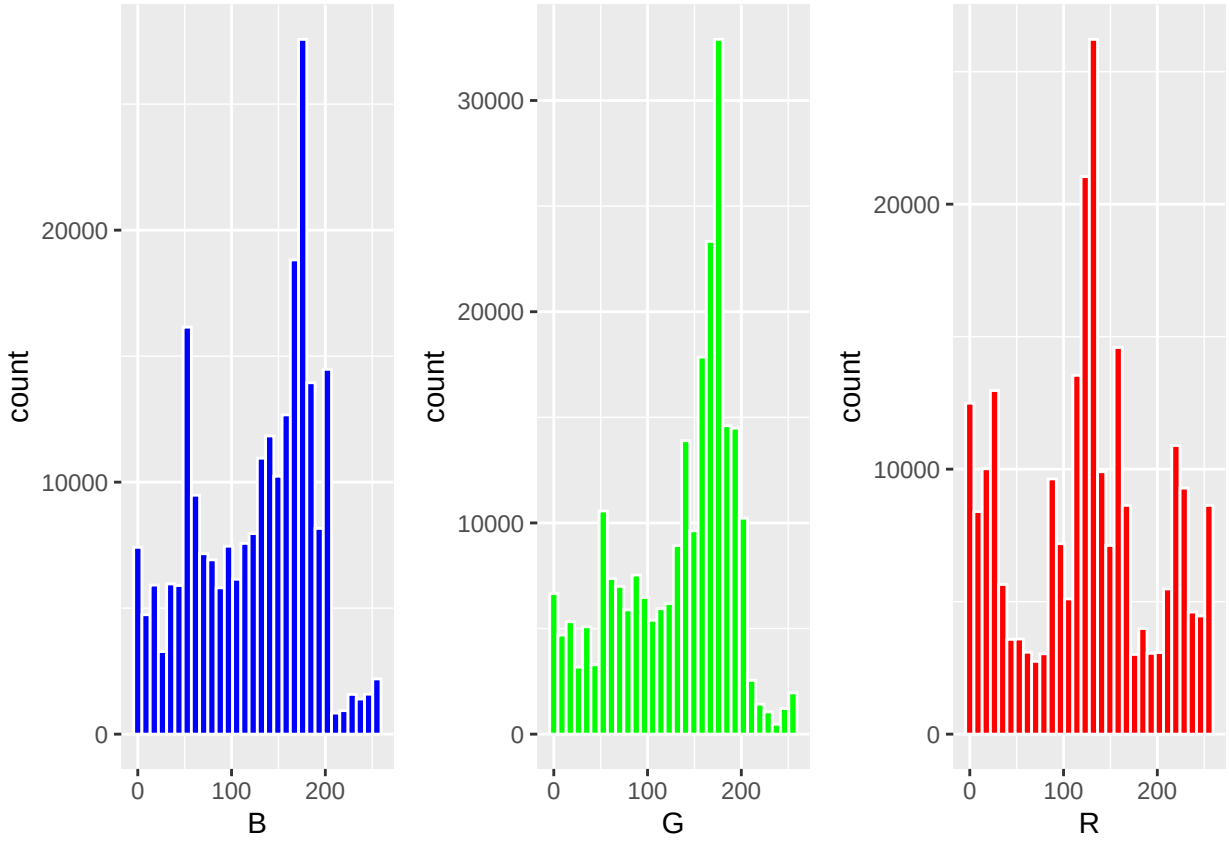
The dataset has 245057 observations with 4 variables. The outcome variable is Y and the other 3 variables B,G and R are predictors. To aid classification we will convert the column Y to a factor and also check proportions using the below code

```
# Convert column Y into factor for classification
sknnon$Y <- as_factor(sknnon$Y)

# check proportions
prop.table(table(sknnon$Y))
```

```
##
##         1         2
## 0.2075395 0.7924605
```

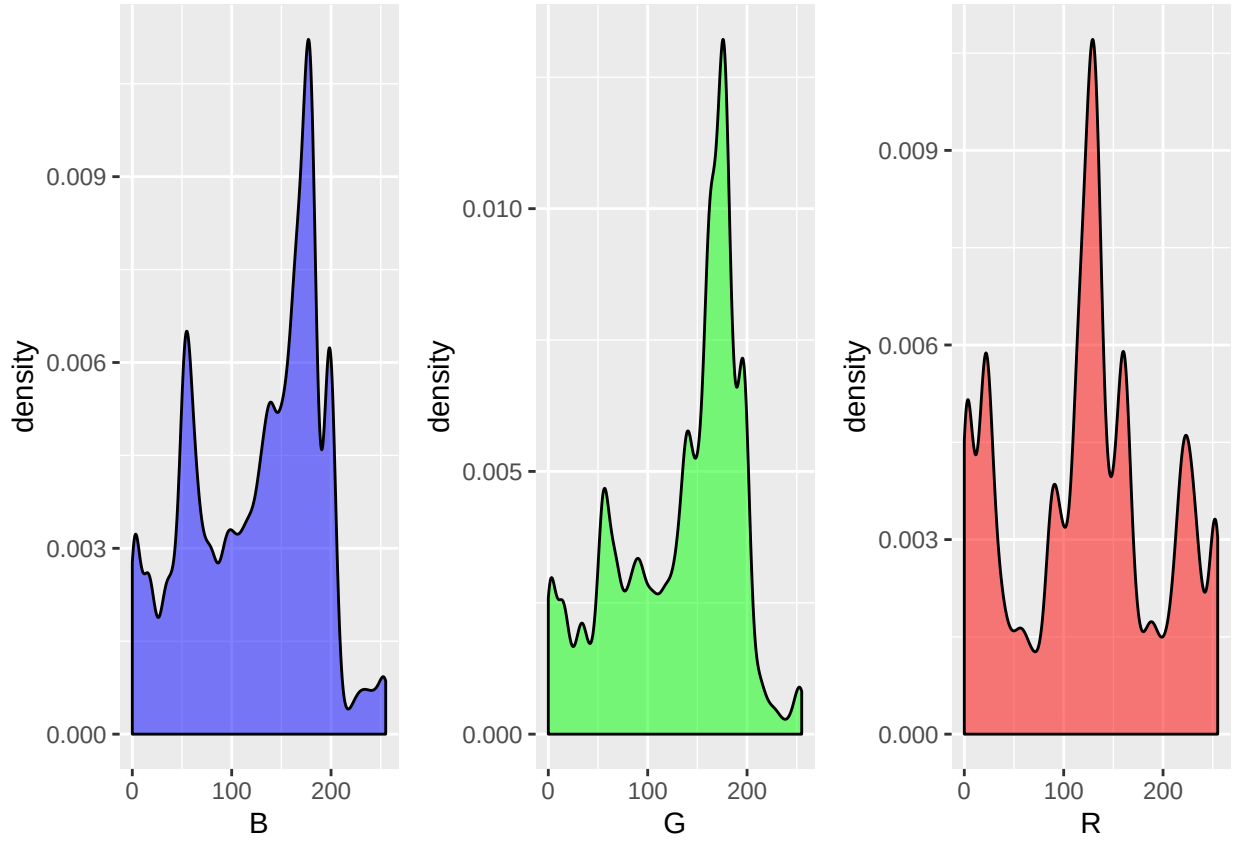
Skin represented by value 1 makes up around 20 percent of the samples and non skin represented by value 2 contributes the remaining 80 percent. Using visualisation techniques this dataset can be analysed further. The package ggplot2 which is part of the tidyverse will be utilised for this purpose. The grid arrange function from the gridExtra package will also be used to align the plots side by side. This significantly aides plot comparisons.

```
# Create histograms with ggplot for each parameter then use grid.arrange to present plots side by side
p1 <- sknnon %>% ggplot(aes(B))+
  geom_histogram(bins = 30, color = "white", fill = "blue")

p2 <- sknnon %>% ggplot(aes(G))+
  geom_histogram(bins = 30, color = "white", fill = "green")

p3 <- sknnon %>% ggplot(aes(R))+
  geom_histogram(bins = 30, color = "white", fill = "red")

grid.arrange(p1,p2,p3,ncol = 3)
```



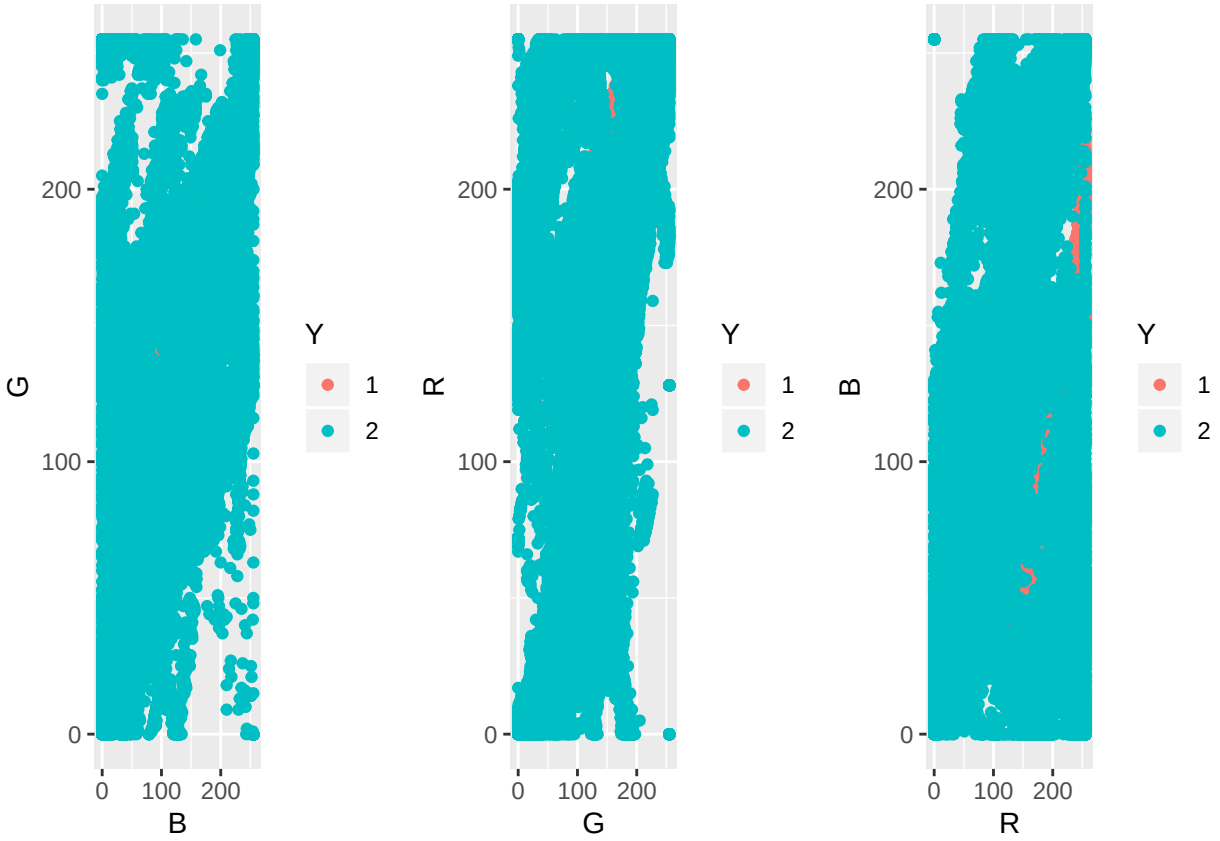
The histograms indicate that B and G values share similarities with distribution while the R values are more widespread. A density plot should make this more clear, it can be created similarly using the below code

```
# Create histograms with ggplot for each parameter then use grid.arrange to present plots side by side

d1 <- sknnon %>% ggplot(aes(B))+
  geom_density(fill = "blue", alpha = 0.5)

d2 <- sknnon %>% ggplot(aes(G))+
  geom_density(fill = "green", alpha = 0.5)

d3 <- sknnon %>% ggplot(aes(R))+
  geom_density(fill = "red", alpha = 0.5)

grid.arrange(d1,d2,d3,ncol = 3)
```



Scatterplots of these variables should provide more information on their distribution. The ggplot2 package helps to do this as well, the points in the plots will be colored based on the Y label.

```
# Create scatter plots of the variables using Y values to color the points

s1 <- sknnon %>% ggplot(aes(B,G, color = Y))+
  geom_point()
s2 <- sknnon %>% ggplot(aes(G,R, color = Y))+
  geom_point()
s3 <- sknnon %>% ggplot(aes(R,B, color = Y))+
  geom_point()
grid.arrange(s1,s2,s3,ncol=3)
```



The scatterplots show something quite interesting in all the plots. Based on their class distributions the points are not interspersed and widespread, instead they are localised and clustered. This makes it a good target for machine learning algorithms. It is assumed at this time that high prediction accuracies can thus be obtained based on these characteristics.

-

Application of Machine learning models

The standard methodology of applying machine learning models on a dataset is to first split the dataset into 2 parts. The larger subset would be used for training purposes and the smaller subset would be reserved exclusively for testing and gaging accuracy. Using this method the sknnon dataset will be partitioned into an initial train and test set by applying a function from the caret package as below.

```
# Set seed
set.seed(1,sample.kind = "Rounding")

# Create data partition
skindex <- createDataPartition(sknnon$Y,1,p=0.8,list = FALSE)

# Use index to create train and test objects
sktrain <- sknnon[skindex,]
sktest <- sknnon[-skindex,]
```

Once the subsets are created the next step is to choose the machine learning model to be applied. The first model applied will be the 'Generalized Linear Model'. The model will be called from within the train function of caret which support 238 models at this time. More information is available in this link. This model will allow a baseline accuracy to be computed. Post this the next model to be applied will the 'Linear Discriminant Analysis' from the MASS package. This will be followed by application of the 'Penalized Discriminant Analysis' from the mda package. Finally 'Quadratic Discriminant Analysis' from the MASS package will be applied. Codes used and results obtained are discussed in detail in the results section

Results

- **Baseline with the 'Generalized Linear Model'**

Using the train function of the caret package apply the model to the subsetted sktrain dataset.

```
# Run a baseline classification model using the train function
set.seed(1,sample.kind = "Rounding")
modglm <- train(Y~B+G+R,
                data = sktrain,
                method = "glm")
modglm
```

```
## Generalized Linear Model
##
## 196047 samples
##      3 predictor
##      2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 196047, 196047, 196047, 196047, 196047, 196047, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9190075  0.7569963
```

Using the predict function as below the trained object will be used to generate predictions on data available in the test subset.

```
# Use the trained model to generate predictions on the test subset of the sknnon dataset
yhatglm <- predict(modglm,sktest)
```

Once the predictions are available a confusion matrix can be used to calculate accuracy as below

```
# Calculate Accuracy using the confusion matrix
confusionMatrix(yhatglm,sktest$Y)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9195878
```

With a rounded accuracy of 92% it can be concluded that the initial assumption regarding the data distribution and its affect on higher prediction accuracies was correct. However can other models improve prediction accuracy?

- **Using 'Linear Discriminant Analysis'**

The next model to be applied is the 'Linear Discriminant Analysis' from the MASS package. As described earlier the model will first be trained using the train function, then predictions will be obtained using predict and finally using the confusion matrix the accuracy will be calculated

```
#using a linear discramanant analysis method to check predicted accuracy
set.seed(1,sample.kind = "Rounding")
modlda <- train(Y~B+G+R,method = "lda",data = sktrain)
modlda
```

```
## Linear Discriminant Analysis
##
## 196047 samples
##      3 predictor
##      2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 196047, 196047, 196047, 196047, 196047, 196047, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9320612  0.803344
```

```
yhatlda <- predict(modlda,sktest)
confusionMatrix(yhatlda,sktest$Y)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9320547
```

The results suggest a small improvement in prediction accuracy to 93.2%. Can this be improved further?

- **'Penalized Discriminant Analysis'**

'Penalized Discriminant Analysis' is the next model that will be applied from the mda package. The method of application remains the same first model will be trained using the train function, then predictions will be obtained using predict and finally using the confusion matrix the accuracy will be calculated

```r
#Using a penalised discriminant analysis method to check predicted accuracy
set.seed(1,sample.kind = "Rounding")
modpda <- train(Y~B+G+R,method = "pda",data = sktrain)
modpda
```

```
## Penalized Discriminant Analysis
##
## 196047 samples
##      3 predictor
##      2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 196047, 196047, 196047, 196047, 196047, 196047, ...
## Resampling results across tuning parameters:
##
##   lambda  Accuracy   Kappa
##   0e+00   0.9320612  0.803344
##   1e-04   0.9320612  0.803344
##   1e-01   0.9320612  0.803344
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was lambda = 0.
```

```r
yhatpda <- predict(modpda,sktest)
confusionMatrix(yhatpda,sktest$Y)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9320547
```

At a rounded 93.2% the accuracy from this model is better than the baseline GLM, however it remains the same as that obtained from LDA.

- **'Quadratic Discriminant Analysis'**

The next model to be applied is also from the MASS package and is the 'Quadratic Discriminant Analysis'. All the application steps remain the same with the exception of the method used.

```r
# Apply Quadratic discriminant analysis and gage predicted accuracy
set.seed(1,sample.kind = "Rounding")
modqda <- train(Y~B+G+R,method = "qda",data = sktrain)
modqda
```

```
## Quadratic Discriminant Analysis
##
## 196047 samples
##      3 predictor
```

```
##      2 classes: '1', '2'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 196047, 196047, 196047, 196047, 196047, 196047, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9832553  0.9477082
```

```
yhatqda <- predict(modqda,sktest)
confusionMatrix(yhatqda,sktest$Y)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9846766
```

Results from this model show a significant improvement in prediction accuracy. At a rounded 98.5% these results show a 6.5% improved accuracy over the baseline GLM model, accuracy difference of 5.3% is noted over the other 2 models. This model also shows the best performance in terms of system time needed to complete training and predictions.

Conclusions

This analysis was conducted on the 'Skin segmentation dataset'. Review of the dataset had revealed that datapoints were localised and clustered and therefore it was assumed that high prediction accuracies should be possible using machine learning algorithms on this data. This assumption was proved correct with application of the baseline GLM model showing a rounded accuracy of 92%. Application of other models improved this accuracy estimate. The Quadratic Discriminant Analysis model with a rounded accuracy of 98.4% was shown to be the best performing model on this type of data.

The biological properties of skin do differentiate it from other surfaces and hence using ml technologies to classify these differences will continue to be successful. Real world application however will require a significantly larger volume of data to be able to demonstrate comparable accuracies.

This report was created in R using the R Studio IDE and the R Markdown package