Building A Model That Helps Locating Displaced People

Tom Lever

06/08/2023

Introduction

In this project, we build a model that would help us locate people displaced by the earthquake in Haiti in 2010. More specifically, we build in a timely manner an accurate model that classifies pixels in geo-referenced aerial images of Haiti in 2010 as depicting blue tarps or depicting objects that are not blue tarps. People whose homes were destroyed by the earthquake often created temporary shelters using blue tarps. Blue tarps were good indicators of where displaced people lived.

Data

Our training data was collected likely by applying a Region Of Interest (ROI) Tool to high-resolution, orthorectified / geo-referenced image of Haiti in 2010. The image is presented in Figure 1 and is sourced from `HaitiOrthorectifiedImage.tif` at Pixel Values from Images over Haiti. One ROI tool is described at Region of Interest (ROI) Tool. Classes may be assigned to pixels by defining Regions Of Interest.

Figure 1: Orthorectified Image Of Haiti In 2010

According to What is orthorectified imagery?, an orthorectified image is an accurately georeferenced image that has been processed so that all pixels are in an accurate (x, y) position on the ground. Orthorectified images have been processed to apply corrections for optical distortions from the sensor system, and apparent changes in the position of ground objects caused by the perspective of the sensor view angle and ground terrain.

Our training data frame is sourced from `HaitiPixels.csv` at Pixel Values from Images over Haiti. Our training data frame consists of $63,241$ observations. Each observation consists of a class in the set

$\{Vegetation,\ Soil,\ Rooftop,\ Various\ Non-Tarp,\ Blue\ Tarp\}$ and a pixel. A pixel is a colored dot. A pixel is represented by a tuple of intensities of color *Red*, *Green*, and *Blue* in the range 0 to 255. See below head and tail of our training data frame.

```
data_frame_of_classes_and_pixels <- read.csv(
    file = "Data_Frame_Of_Classes_And_Pixels.csv"
)
head(x = data_frame_of_classes_and_pixels, n = 2)
```

```
#        Class Red Green Blue
# 1 Vegetation  64    67   50
# 2 Vegetation  64    67   50
```

```
tail(x = data_frame_of_classes_and_pixels, n = 2)
```

```
#           Class Red Green Blue
# 63240 Blue Tarp 132   139  149
# 63241 Blue Tarp 133   141  153
```

We conduct exploratory data analysis by examining the distribution of classes (such as *Blue Tarp*) in a space defined by intensities of color *Red*, *Green*, and *Blue*. See Figure 2. The intensity space for pixels representing blue tarps is distinct from the intensity sapce for pixels representing objects that are not blue tarps.

```
distribution_of_classes_in_intensity_space <- plotly::plot_ly(
    data = data_frame_of_classes_and_pixels,
    x = ~Red,
    y = ~Green,
    z = ~Blue,
    color = ~Class,
    type = "scatter3d",
    mode = "markers",
    colors = c("blue", "black", "brown", "orange", "green")
)
htmlwidgets::saveWidget(
    widget = distribution_of_classes_in_intensity_space,
    file = "distribution_of_classes_in_intensity_space.html"
)
webshot2::webshot(
    url = "distribution_of_classes_in_intensity_space.html",
    file = "distribution_of_classes_in_intensity_space.png"
)
```

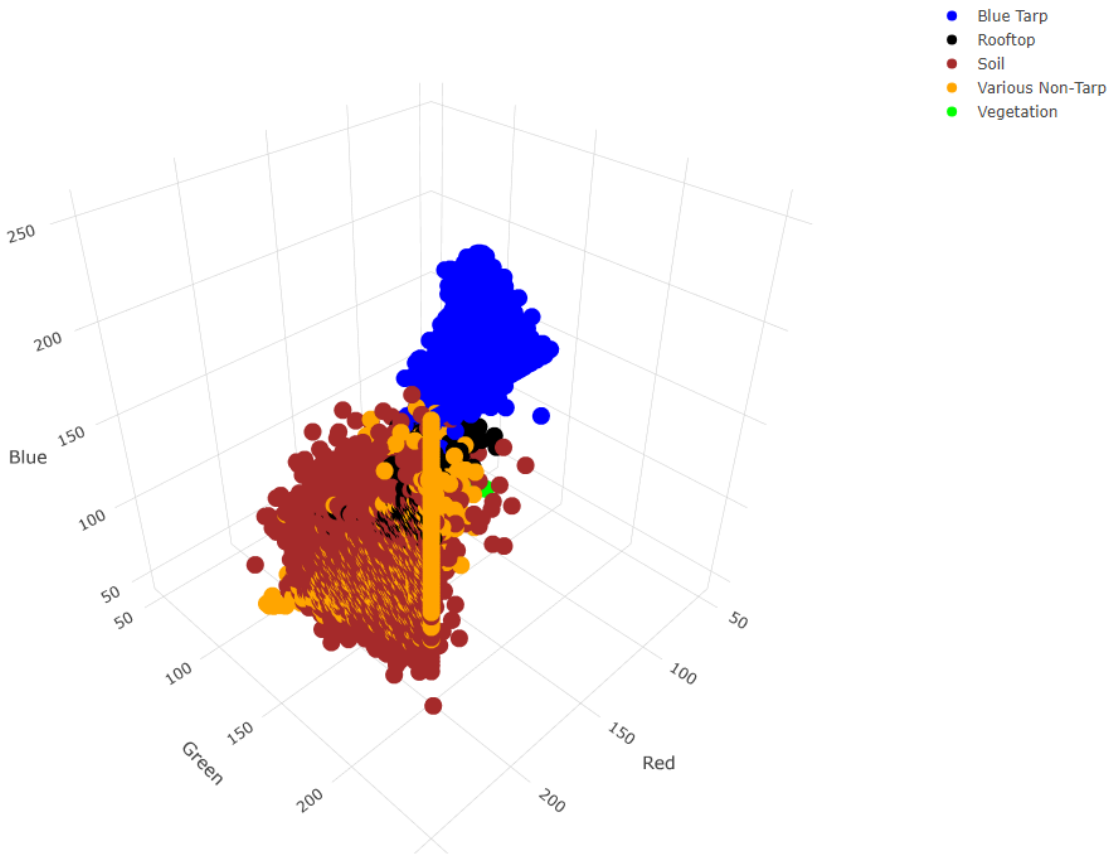

Figure 2: Distribution Of Classes In Intensity Space

We consider the distributions of intensities of color *Red*, *Green*, and *Blue*. See Figures 3, 4, and 5. No distribution of intensities of color is normal.

```
TomLeversRPackage::plot_distribution(data_frame_of_classes_and_pixels, "Red", "red")
```

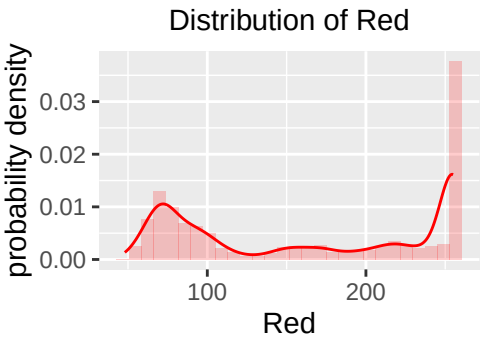

Figure 3: Distribution Of Intensities Of Color Red

```
TomLeversRPackage::plot_distribution(data_frame_of_classes_and_pixels, "Green", "green")
```

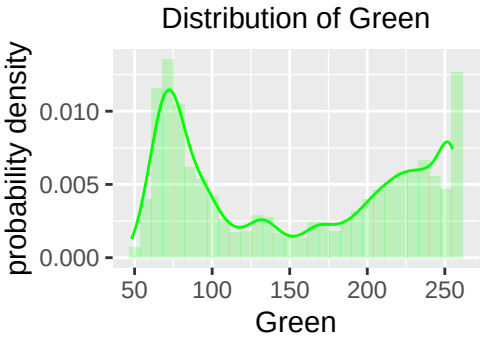

Figure 4: Distribution Of Intensities Of Color Green

```
TomLeversRPackage::plot_distribution(data_frame_of_classes_and_pixels, "Blue", "blue")
```

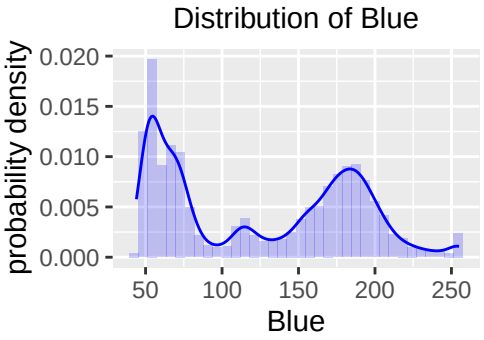

Figure 5: Distribution Of Intensities Of Color Blue

Methods

We build binary classifiers that classify pixels as depicting blue tarps or depicting objects that are not blue tarps. We may ignore non-binary classifiers that predict probabilities for all classes and may be used to locate pixels that more likely depict blue tarps than objects that are not blue tarps. We may ignore non-binary classifiers since the intensity space for pixels representing blue tarps is distinct from the intensity space for pixels representing objects that are not blue tarps.

In order to build binary classifiers, we create a data frame with a column of indicators of whether of not a pixel depicts a blue tarp instead of a column of classes. We normalize as opposed to standardize intensities given that distributions of intensity are not normal. We add columns corresponding to standardized intensities and intensities standardized after transforming by the natural logarithm, the square root, the square, and interactions.

```
number_of_observations <- nrow(data_frame_of_classes_and_pixels)
column_of_indicators <- rep(0, number_of_observations)
condition <- data_frame_of_classes_and_pixels$Class == "Blue Tarp"
```

```
column_of_indicators[condition] <- 1
factor_of_indicators <- factor(column_of_indicators)
data_frame_of_indicators_and_pixels <- data.frame(
    Indicator = factor_of_indicators,
    Red = data_frame_of_classes_and_pixels$Red,
    Green = data_frame_of_classes_and_pixels$Green,
    Blue = data_frame_of_classes_and_pixels$Blue
)
vector_of_random_indices <- sample(1:number_of_observations)
data_frame_of_indicators_and_pixels <- data_frame_of_indicators_and_pixels[vector_of_random_indices, ]
library(TomLeversRPackage)
data_frame_of_indicators_and_pixels[, "Scaled_Red"] <- normalize_vector(data_frame_of_indicators_and_pi
data_frame_of_indicators_and_pixels[, "Scaled_Green"] <- normalize_vector(data_frame_of_indicators_and_
data_frame_of_indicators_and_pixels[, "Scaled_Blue"] <- normalize_vector(data_frame_of_indicators_and_pi
data_frame_of_indicators_and_pixels[, "Scaled_Natural_Logarithm_Of_Red"] <- normalize_vector(
 log(data_frame_of_indicators_and_pixels[, "Red"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Natural_Logarithm_Of_Green"] <- normalize_vector(
 log(data_frame_of_indicators_and_pixels[, "Green"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Natural_Logarithm_Of_Blue"] <- normalize_vector(
 log(data_frame_of_indicators_and_pixels[, "Blue"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Root_Of_Red"] <- normalize_vector(
 sqrt(data_frame_of_indicators_and_pixels[, "Red"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Root_Of_Green"] <- normalize_vector(
 sqrt(data_frame_of_indicators_and_pixels[, "Green"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Root_Of_Blue"] <- normalize_vector(
 sqrt(data_frame_of_indicators_and_pixels[, "Blue"])
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Of_Red"] <- normalize_vector(
 I(data_frame_of_indicators_and_pixels[, "Red"]^2)
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Of_Green"] <- normalize_vector(
 I(data_frame_of_indicators_and_pixels[, "Green"]^2)
)
data_frame_of_indicators_and_pixels[, "Scaled_Square_Of_Blue"] <- normalize_vector(
 I(data_frame_of_indicators_and_pixels[, "Blue"]^2)
)
data_frame_of_indicators_and_pixels[, "Scaled_Interaction_Of_Red_And_Green"] <- normalize_vector(
 as.numeric(interaction(data_frame_of_indicators_and_pixels$Red, data_frame_of_indicators_and_pixels$Gr
)
data_frame_of_indicators_and_pixels[, "Scaled_Interaction_Of_Red_And_Blue"] <- normalize_vector(
 as.numeric(interaction(data_frame_of_indicators_and_pixels$Red, data_frame_of_indicators_and_pixels$Bl
)
data_frame_of_indicators_and_pixels[, "Scaled_Interaction_Of_Green_And_Blue"] <- normalize_vector(
 as.numeric(interaction(data_frame_of_indicators_and_pixels$Green, data_frame_of_indicators_and_pixels$
)
```

We use 10-fold cross-validation to evaluate the performance of 5 classifiers. A classifier will classify a pixel as depicting a blue tarp or depicting an object that is not a blue tarp. We consider graphs each with an

increasing purple curve representing Average Precision vs. Threshold, a decreasing red curve representing Average Recall vs. Threshold, a forest-green curve representing Average F1 Measure vs. Threshold, and a teal curve representing Average Decimal Of True Positives vs. Threshold.

A threshold is a probability between 0 and 1. A model classifies a pixel as representing a blue tarp if the model the probability that the pixel represents a blue tarp that the model predicts is greater than the threshold. Precision is the ratio of true positives to predicted positives. Recall is the ratio of true positives to actual positives. An F1 measure is the harmonic mean of precision and recall. A decimal of true positives is the ratio of number of true positives to total number of predictions.

When tuning the thresholds of our models we prioritize recall at least as much as precision. We prioritize identifying as many positives correctly as possible over having predicted positives be correct. We recommend models with thresholds less than or equal to the threshold t that corresponds to the maximum F1 measure and that is least. Note that precision will decrease more rapidly than recall will increase for thresholds less than t. Our ideal threshold may be t or may be the highest threshold at which the derivative of Average Precision vs. Threshold is 1.

We consider how the number of refugees to visit may change with precision. We assume that the testing data consists of actual indicators and random pixels from the training image. Suppose precision $PPV = TP/\hat{P} = 0.944$. Suppose decimal of true positives $TPD = TP/n = 0.0296$. The decimal of predicted positives $PPD = \hat{P}/n = TPD/PPV = \frac{TP/n}{TP/\hat{P}} = 0.0296/0.944 = 0.0313$. The number of pixels in our training image is $4441 \times 6833 = 30,345,353$. The number of pixels in our training image that might be predicted to represent blue tarps is $949,810$. Considering a blue tarp in the top left corner of our training image, the number of pixels representing one blue tarp may be $36 \times 36 = 1,296$. The predicted number of blue tarps in our training image may be 733. We assume that there is one refugee per blue tarp. If we change precision by a factor of -0.139 to 0.8125, the predicted number of blue tarps in our training image may increase by 120 to 853. The number of true positives may increase by 4 and the number of false negatives may decrease by 4.

Suppose the width of our training image is $1,500\ ft$. Then our training image's height is about $2,308\ ft$ and its area is about $0.124\ mi^2$. Suppose the area of Léogâne is \$148.7 mi^2. There are about $1,200$ areas like that depicted in our training image in Léogâne.

We use 10-fold cross-validation to evaluate the performance of logistic-regression models. We perform manual bidirectional selection. According to https://www.ibm.com/docs/en/contentclassificatio/8.8?topic=analysis-category-graph-precision-recall-vs-threshold, "The ideal threshold setting is the highest possible recall and precision rate. This goal is not always achievable, because the higher the recall rate, the lower the precision rate, and vice versa. Setting the most appropriate threshold for a category is a trade-off between these two rates." We consider the ideal threshold setting to correspond to the highest F1 measure. First, we consider the addition to an intercept only logistic regression model of one of the predictive terms Red, $Green$, $Blue$, $log(Red)$, $log(Green)$, $log(Blue)$, $sqrt(Red)$, $sqrt(Green)$, $sqrt(Blue)$, Red^2, $Green^2$, and $Blue^2$. The logistic regression model with formula $Indicator \sim Blue^2$ has the highest F1 measure. We consider the addition of each of the above predictive terms as well as $Red : Green$, $Red : Blue$, and $Green : Blue$. The logistic regression model with formula $Indicator \sim Blue^2 + Red^2$ has the highest F1 measure. We do not remove either predictive term from this model. We follow this process until we arrive at a logistic regression model with formula $Indicator \sim normalizeBlue^2) + normalizeRed^2) + Green : Blue + normalizesqrt(Blue))$ with F1 measure 0.946. Neither removing nor adding a predictor to this model results in a model with higher F1 measure.

Following the same process, our best Logistic Ridge Regression model has formula $Indicator \sim normalizelog(Blue)) + normalizesqrt(Red))$ and F1 measure 0.922.

Our best Linear Discriminant Analysis model has formula $Indicator \sim log(Blue) + sqrt(Red)$ and F1 measure 0.892.

Our best Quadratic Discriminant Analysis model has formula $Indicator \sim Blue^2 + Red^2 + Red : Green + Red + Blue + Green$ and F1 measure 0.939.

Our best KNN model has formula $Indicator \sim normalizeRed : Blue) + normalizeGreen : Blue) + normalizesqrt(Blue))$ and F1 measure 0.943.

Our Logistic Regression model has the best performance according to F1 measures, followed by our KNN, QDA, Logistic Ridge Regression, and LDA models.