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

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

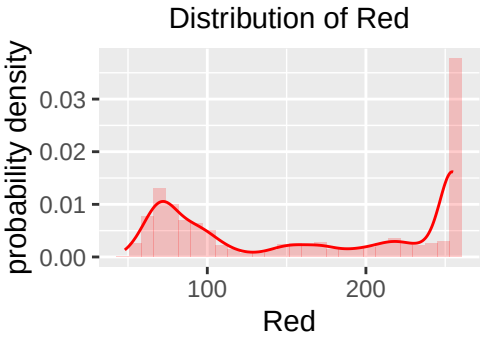

Figure 2: Distribution Of Intensities Of Color Red

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

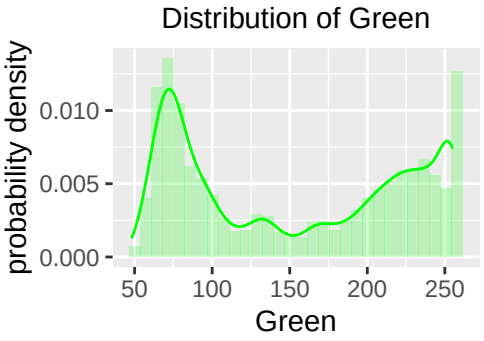

Figure 3: Distribution Of Intensities Of Color Green

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

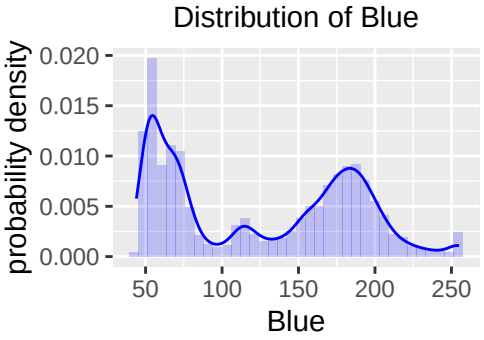

Figure 4: Distribution Of Intensities Of Color Blue

We examine 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"
)
```

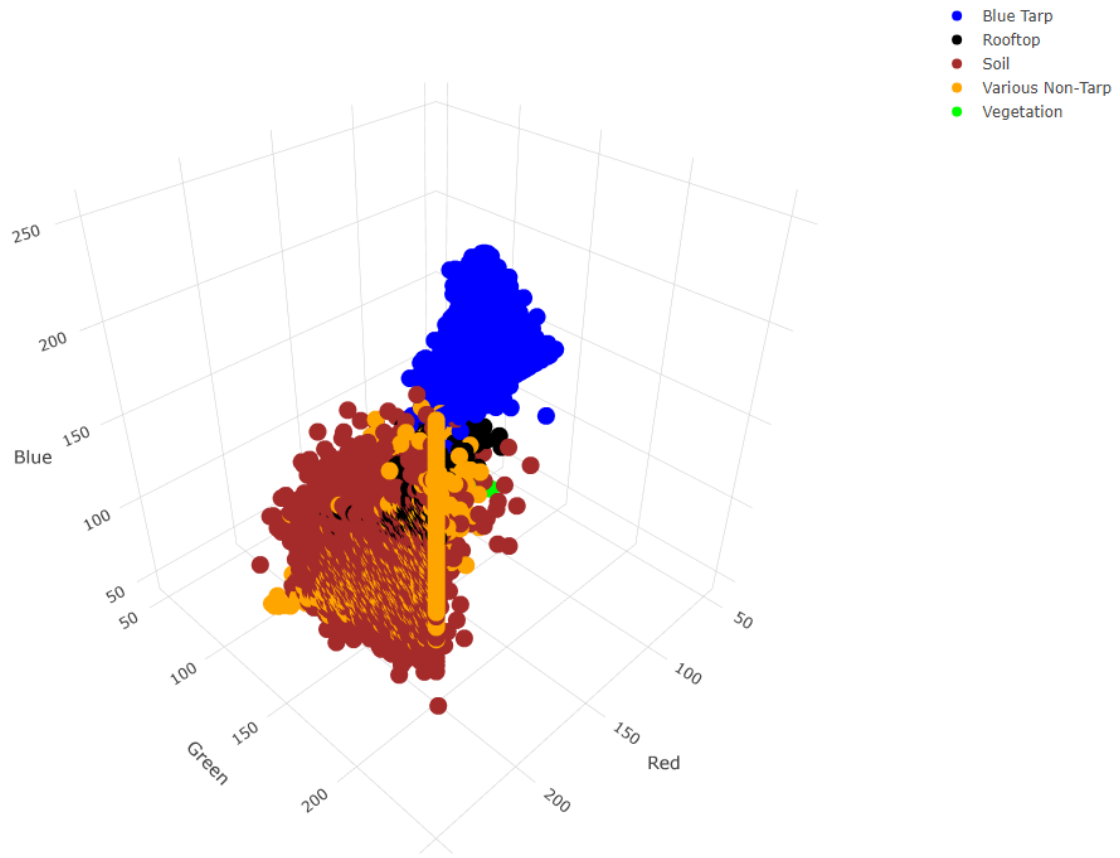

Figure 5: Distribution Of Classes In Intensity Space

Methods

Choosing A Binary Classifier

Since the intensity space for pixels representing blue tarps is distinct from the intensity space for pixels representing objects that are not blue tarps, we consider our optimal model to be a binary classifier that classifies 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.

Choosing A Performance Metric

A threshold is a probability. A model classifies a pixel as representing a blue tarp if the probability that the pixel represents a blue tarp that the model predicts is greater than the threshold. False Positive Rate (FPR) is the ratio of false positives to actual negatives. Precision / Positive Predictive Value (PPV) is the ratio of true positives to predicted positives. Recall / True Positive Rate (TPR) is the ratio of true positives to actual positives. An F1 measure is the harmonic mean of PPV and TPR. A decimal of true positives is the ratio of number of true positives to total number of predictions. All of these quantities lie between 0 and 1.

Receiver Operator Characteristic (ROC) graphs are graphs of TPR Vs. FPR for different thresholds. According to ROC and AUC, Clearly Explained!, the Area Under The Curve (AUC) of an ROC graph "makes it easy to compare one ROC curve to another" and to compare one binary classifier to another. If the AUC for one ROC curve is greater than the AUC for another ROC curve, the binary classifier with the former ROC curve is better than the binary classifier with the latter ROC curve. A given model's threshold may be tuned so that its FPR is close to 0 and its TPR is close to 1. We may consider maximizing ROC AUC among binary classifiers and tuning a binary classifier's thresholds to minimize the distance between the corresponding point (FPR, TPR) and (0, 1).

"People often replace the False Positive Rate with Precision... If there were lots of samples that were not [Blue Tarps] relative to the number of [Blue Tarp] samples, then Precision might be more useful than the False Positive Rate. This is because Precision does not include the number of True Negatives in its calculation, and is not effected by the imbalance. In practice, this sort of imbalance occurs when studying a rare [occurrence]. In this case, the study will contain many more [pixels not corresponding to blue tarps than corresponding to blue tarps]. We may consider maximizing PR AUC among binary classifiers and tuning a binary classifier's thresholds to minimize the distance between the corresponding point (TPR, PR) and (1, 1).

According to 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.

When tuning hyperparameter threshold, λ for Logistic Ridge Regression models, and K for KNN models, we prioritize TPR at least as much as PPV. 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 PPV will decrease more rapidly than TPR will increase for thresholds less than t. Our ideal threshold may be t or may be the highest threshold less than t at which the derivative of Average PPV vs. Threshold is 1.

Data Frame For Modeling

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 add columns corresponding to normalized intensities and intensities normalized after transforming by the natural logarithm, the square root, the square, and interactions. We normalize as opposed to standardize intensities given that distributions of intensity are not normal. We shuffle the rows in the data frame.

```
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,
    Normalized_Red = normalize_vector(data_frame_of_classes_and_pixels[, "Red"]),
    Normalized_Green = normalize_vector(data_frame_of_classes_and_pixels[, "Green"]),
    Normalized_Blue = normalize_vector(data_frame_of_classes_and_pixels[, "Blue"]),
    Normalized_Natural_Logarithm_Of_Red =
        normalize_vector(log(data_frame_of_classes_and_pixels[, "Red"])),
    Normalized_Natural_Logarithm_Of_Green =
        normalize_vector(log(data_frame_of_classes_and_pixels[, "Green"])),
    Normalized_Natural_Logarithm_Of_Blue =
        normalize_vector(log(data_frame_of_classes_and_pixels[, "Blue"])),
    Normalized_Square_Root_Of_Red =
```

```
        normalize_vector(sqrt(data_frame_of_classes_and_pixels[, "Red"])),
    Normalized_Square_Root_Of_Green =
        normalize_vector(sqrt(data_frame_of_classes_and_pixels[, "Green"])),
    Normalized_Square_Root_Of_Blue =
        normalize_vector(sqrt(data_frame_of_classes_and_pixels[, "Blue"])),
    Normalized_Square_Of_Red =
        normalize_vector(I(data_frame_of_classes_and_pixels[, "Red"]^2)),
    Normalized_Square_Of_Green =
        normalize_vector(I(data_frame_of_classes_and_pixels[, "Green"]^2)),
    Normalized_Square_Of_Blue =
        normalize_vector(I(data_frame_of_classes_and_pixels[, "Blue"]^2)),
    Normalized_Interaction_Of_Red_And_Green = normalize_vector(
        as.numeric(
            interaction(
                data_frame_of_classes_and_pixels$Red,
                data_frame_of_classes_and_pixels$Green
            )
        )
    ),
    Normalized_Interaction_Of_Red_And_Blue = normalize_vector(
        as.numeric(
            interaction(
                data_frame_of_classes_and_pixels$Red,
                data_frame_of_classes_and_pixels$Blue
            )
        )
    ),
    Normalized_Interaction_Of_Green_And_Blue = normalize_vector(
        as.numeric(
            interaction(
                data_frame_of_classes_and_pixels$Green,
                data_frame_of_classes_and_pixels$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, ]
t(data_frame_of_indicators_and_pixels[1, ])
```

```
#                                      49149
# Indicator                            "0"
# Normalized_Red                       "0.5072464"
# Normalized_Green                     "0.4347826"
# Normalized_Blue                      "0.3554502"
# Normalized_Natural_Logarithm_Of_Red  "0.6941278"
# Normalized_Natural_Logarithm_Of_Green "0.6323432"
# Normalized_Natural_Logarithm_Of_Blue "0.5662447"
# Normalized_Square_Root_Of_Red        "0.6018587"
# Normalized_Square_Root_Of_Green      "0.5330599"
# Normalized_Square_Root_Of_Blue       "0.4579804"
# Normalized_Square_Of_Red             "0.336490...."
```

```
# Normalized_Square_Of_Green                "0.266896...."
# Normalized_Square_Of_Blue                 "0.193773...."
# Normalized_Interaction_Of_Red_And_Green   "0.4351032"
# Normalized_Interaction_Of_Red_And_Blue    "0.3561482"
# Normalized_Interaction_Of_Green_And_Blue "0.3557934"
```

Grid Search

We conduct a grid search for an optimal binary classifier according to the $F1measure$. We also compare models according to Area Under The Precision-Recall Curve (PR AUC); Area Under The ROC Curve (ROC AUC); graphs of Prediction Metrics Vs. Thresholds, Precision-Recall Curves, and ROC Curves; and a Performance Table.

The primary dimension of our grid is type of binary classifier. We compare Logistic Regression, Logistic Ridge Regression, LDA, QDA, and KNN binary classifiers.

The secondary dimension of our grid is the set of predictive terms. We consider binary classifiers with the terms in the above output other than $Indicator$.

The third dimension of our grid is value of primary hyperparameter. For Logistic Ridge Regression binary classifiers, the primary hyperparameter is λ. For K Nearest Neighbors (KNN) binary classifiers, the primary hyperparameter is K.

To search for an optimal Logistic Regression binary classifier, we perform manual bidirectional selection. Through 10-fold cross validation we calculate the average F1 measure for 10 Logistic Regression binary classifiers with formula $Indicator \sim normalize(Red)$ that were trained on 9 tenths of our data frame of indicators and pixels and tested on 1 tenth of our data frame. Similarly, we calculate the average F1 measure for Logistic Regression binary classifiers with formula $Indicator \sim normalize(Green), Indicator \sim normalize(Blue), ..., Indicator \sim normalize\left(Blue^2\right)$. On one knit of this paper, the Logistic Regression binary classifiers with formula $Indicator \sim normalize(Blue)$ have the highest average F1 measure of 0.445. We consider the average F1 measures for Logistic Regression binary classifiers with formulas $Indicator \sim normalize(Blue) + normalize(Red), ..., Indicator \sim normalize(Blue) + normalize(Green : Blue)$. The Logistic Regression binary classifiers with formula $Indicator \sim normalize(Blue) + normalize(Red)$ have the highest average F1 measure of 0.931. We have already considered the average F1 measures for logistic binary classifiers with formulas $Indicator \sim normalize(Red)$ and $Indicator \sim normalize(Blue)$. We consider the average F1 measures for Logistic Regression binary classifiers with formulas $Indicator \sim normalize(Blue) + normalize(Red) + normalize(Green), ..., Indicator \sim normalize(Blue) + normalize(Red) + normalize(Green : Blue)$. The Logistic Regression binary classifiers with formula $Indicator \sim normalize(Blue) + normalize(Red) + normalize\left(Green^2\right)$ have the highest average F1 measure of 0.942. The average F1 measure for Logistic Regression binary classifiers with formula $Indicator \sim normalize(Red) + normalize\left(Green^2\right)$ is 0.449. We consider the average F1 measures for Logistic Regression binary classifiers with formulas $Indicator \sim normalize(Blue)+normalize(Red)+normalize\left(Green^2\right)+normalize(Green), ..., Indicator \sim normalize(Blue) + normalize(Red) + normalize\left(Green^2\right) + normalize(Green : Blue)$. None of these binary classifiers have average F1 measures greater than 0.942.

```
summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "Logistic Regression",
    formula = Indicator ~ Normalized_Blue + Normalized_Red + Normalized_Square_Of_Green,
    data_frame = data_frame_of_indicators_and_pixels
)
```

To search for an optimal Logistic Ridge Regression binary classifier, we perform manual bidirectional selection as above. For a given formula, before cross validation, we search for one value of hyperparameter λ. We use `caret::train` to choose $\lambda = 0.0035$ for all of our Logistic Ridge Regression models. According to

the documentation for `caret::train`, `caret::train` "sets up a grid of tuning parameters for a number of classification and regression routines, fits each model, and calculates a resampling based performance measure". Our inputs include our formula specifying response and predictive terms, our full data frame of indicators and pixels, the method "glmnet", the metric "F1_measure", a `trainControl` object, and a data frame containing one row for each combination of $\alpha = 0$ and value of λ to be evaluated. We construct the `trainControl` object with method "cv" denoting resampling method cross validation and summary function `calculate_F1_measure` to compute F1 measures across resamples. $\alpha = 0$ indicates a combination of ridge regression and lasso regression that is effectively ridge regression. λ is a constant that determines the importance of squares of coefficients to the quantity to be minimized when conducting Logistic Regression. We use $glmnet :: glmnet$ to choose a sequence of values of $lambda$ to be evaluated. According to the documentation for `glmnet::glmnet`, `glmnet` fits "a generalized linear model via penalized maximum likelihood." Our inputs include a matrix of the values of the predictive terms in our data frame of indicators and pixels, a vector of response values, an indication that the generalized linear model is a Logistic Regression model, and $alpha = 0$.

Our optimal Logistic Ridge Regression binary classifier has formula $Indicator \sim normalize(log(Blue)) + normalize(sqrt(Red))$. On one knit of this paper, the maximum average F1 measure is 0.921.

```
library(TomLeversRPackage)
summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "Logistic Ridge Regression",
    formula = Indicator ~ Normalized_Natural_Logarithm_Of_Blue + Normalized_Square_Root_Of_Red,
    data_frame = data_frame_of_indicators_and_pixels
)
```

To search for an optimal Logistic Ridge Regression binary classifier, we perform manual bidirectional selection as above. Our optimal Logistic Ridge Regression binary classifier has formula $Indicator \sim normalize(Blue) + normalize\left(Red^2\right) + normalize\left(Green^2\right)$. On one knit of this paper, the maximum average F1 measure is 0.891.

```
library(TomLeversRPackage)
summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "LDA",
    formula = Indicator ~ Normalized_Blue + Normalized_Square_Of_Red + Normalized_Square_Of_Green,
    data_frame = data_frame_of_indicators_and_pixels
)
```

To search for an optimal Logistic Ridge Regression binary classifier, we perform manual bidirectional selection as above. Our optimal Logistic Ridge Regression binary classifier has formula $Indicator \sim normalize\left(Blue^2\right) + normalize\left(Red^2\right) + normalize\left(Red\right)$. On one knit of this paper, the maximum average F1 measure is 0.938.

```
library(TomLeversRPackage)
summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "QDA",
    formula = Indicator ~ Normalized_Square_Of_Blue + Normalized_Square_Of_Red + Normalized_Red + Normal
    data_frame = data_frame_of_indicators_and_pixels
)
```

To search for an optimal K Nearest Neighbors binary classifier, we perform manual bidirectional selection as above. For a given formula, before cross validation, we search for one value of hyperparameter K. We use `caret::train` to choose $K = 3$ for all of our Logistic Ridge Regression models. According to the documentation for `caret::train`, `caret::train` "sets up a grid of tuning parameters for a number of classification and regression routines, fits each model, and calculates a resampling based performance

measure". Our inputs include our formula specifying response and predictive terms, our full data frame of indicators and pixels, the method "knn", the metric "F1_measure", a `trainControl` object, and a data frame containing one row for each K to be evaluated. Dr. Peter Gedeck recommended evaluating $K \in \{1, 2, 3\}$. We construct the `trainControl` object with method "cv" denoting resampling method cross validation and summary function `calculate_F1_measure` to compute F1 measures across resamples. K is a constant that determines the number of neighboring training vectors consisting of predictive terms used to determine the probability that a testing vector consisting of predictive terms corresponds to a blue tarp.

Our optimal KNN binary classifier has formula $Indicator \sim normalize(Red : Blue) + normalize(Green : Blue) + normalize\left(\sqrt{Blue}\right)$. On one knit of this paper, the maximum average F1 measure is 0.921.

```
library(TomLeversRPackage)
summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "KNN",
    formula = Indicator ~ Normalized_Interaction_Of_Red_And_Blue + Normalized_Interaction_Of_Green_And_
    data_frame = data_frame_of_indicators_and_pixels
)
```

Performance Table

Per our Performance Table, our Logistic Regression model with threshold $t = 0.18$ has the best performance according to Area Under The Precision-Recall Curve, Area Under The ROC Curve, accuracy, True Positive Rate, and F1 measure. Our Logistic Regression model also has the highest False Positive Rate and lowest Precision. Our KNN model with $K = 3$ and threshold $t = 0.08$ has the best performance according to False Positive Rate and the second-best performance according to accuracy, True Positive Rate, precision, and F1 measure. Our KNN model also has the second-worst performance according to Area Under The Precision-Recall Curve and Area Under The ROC Curve. Performance seems to worsen progressing from our QDA model with threshold $t = 0.96$ to our Logistic Ridge Regression model with $\alpha = 0$, $\lambda = 0.0035$, and threshold $t = 0.15$ to our LDA model with threshold $t = 0.811$.

The difference in True Positive Rate 0.028 between our Logistic Regression model with threshold $t = 0.18$ and our KNN model with $K = 3$ and threshold $t = 0.08$ is approximately equal to the difference in Precision 0.026. Because we prioritize True Positive Rate over Precision, we recommend our Logistic Regression Model with threshold $t = 0.18$.

Quantity	Method Of Determination	Logistic Regression	Logistic Ridge Regression	LDA	QDA	KNN
optimal threshold	value corresponding to optimal F1 measure	0.180	0.150	0.811	0.960	0.080
alpha	minimum	-	-	-	-	-
optimal lambda	lambda in sequence provided by glmnet::glmnet corresponding to maximum F1 measure	-	0.004	-	-	-
optimal K	K in sequence provided by Peter Gedeck corresponding to maximum F1 measure	-	-	-	-	3.000
optimal Area Under The Precision-Recall Curve	average	0.984	0.967	0.904	0.965	0.943
optimal Area Under The ROC Curve	average	0.999	0.998	0.950	0.998	0.973
optimal accuracy	value corresponding to optimal F1 measure	0.996	0.995	0.994	0.996	0.996
optimal True Positive Rate	value corresponding to optimal F1 measure	0.966	0.905	0.811	0.936	0.938
optimate False Positive Rate	value corresponding to optimal F1 measure	0.003	0.002	0.000	0.002	0.002
optimal precision	value corresponding to optimal F1 measure	0.918	0.938	0.991	0.941	0.944
optimal F1 measure	maximum average	0.942	0.921	0.892	0.938	0.941

Figure 6: Performance Table

Conclusions

We consider how the number of refugees to visit may change with PPV. We assume that the testing data consists of actual indicators and random pixels from the training image. Suppose PPV $PPV = TP/\hat{P} = 0.944$. 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 PPV 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.