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 and holdout data were collected likely by applying a Region Of Interest (ROI) Tool to high-resolution, orthorectified / geo-referenced images of Haiti in 2010. Our training 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 a CSV file like `HaitiPixels.csv` at Pixel Values from Images over Haiti. Our training data frame consists of $63,241$ observations. Each training 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.

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

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

```
tail(x = training_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.

Please explore TomLeversRPackage.

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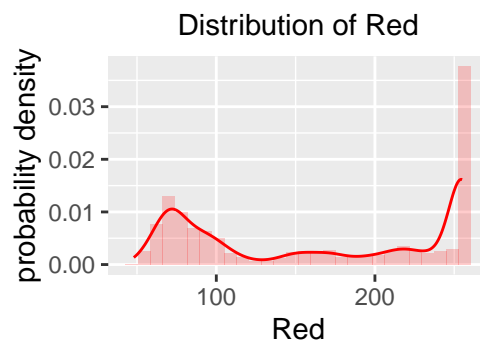

Figure 2: Distribution Of Intensities Of Color Red

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

Figure 3: Distribution Of Intensities Of Color Green

```
plot_distribution(training_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 = training_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"
)
```
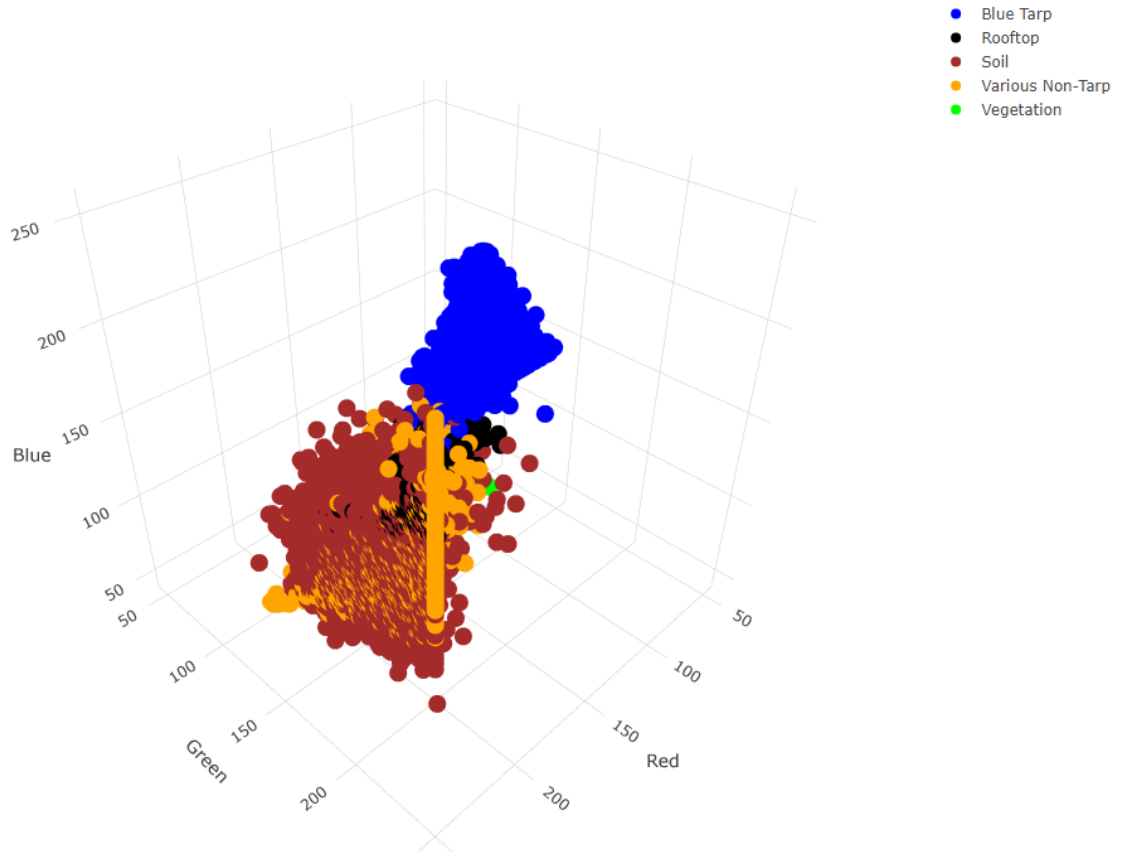
Figure 5: Distribution Of Classes In Intensity Space

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.

Data, Continued

Our holdout data frame is sourced from CSV files like `orthovnir057_ROI_NON_Blue_Tarps.txt`, `orthovnir067_ROI_Blue_Tarps.txt`, `orthovnir067_ROI_NOT_Blue_Tarps.txt`, `orthovnir069_ROI_Blue_Tarps.txt`, `orthovnir069_ROI_NOT_Blue_Tarps.txt`, and `orthovnir078_ROI_NON_Blue_Tarps.txt` at the above link and `orthovnir078_ROI_Blue_Tarps.txt` from Dr. Peter Gedeck. Each holdout observation consists of a class in the set {*Not Blue Tarp*, *Blue Tarp*} and a pixel. See below head and tail of our holdout data frame.

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

```
#           Class Red Green Blue
# 1 Not Blue Tarp 104    89   63
# 2 Not Blue Tarp 101    80   60
```

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

```
#                 Class Red Green Blue
# 2004176 Not Blue Tarp  88    75   73
# 2004177 Not Blue Tarp  72    65   62
```

Methods, Continued

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."

According to Optimal Thresholding of Classifiers to Maximize F1 Measure, "The [balanced] harmonic mean of precision and recall, the F1 measure is widely used to evaluate the success of a binary classifier when one class is rare." According to The truth of the F-measure, "the F-measure was first introduced to evaluation

tasks of information extraction technology at the Fourth Message Understanding Conference (MUC-4) in 1992."

The average of two ratios (e.g., precision and recall) is the balanced harmonic mean of those two ratios. According to Harmonic mean, "For instance, if a vehicle travels a certain distance d outbound at a speed s_1 (e.g., 60 km/h) and returns the distance at a speed s_2 (e.g., 20 km/h), then its average speed is the harmonic mean of s_1 and s_2 (30 km/h), not the arithmetic mean (40 km/h). The total travel time is the same as if it had traveled the whole distance at that average speed. This can be proven as follows:

$$Average\ speed\ for\ the\ entire\ journey = \frac{Total\ distance\ traveled}{Sum\ of\ time\ for\ each\ segment} = \frac{D}{T} = \frac{D}{t_1 + t_2} = \frac{2d}{\frac{d}{s_1} + \frac{d}{s_2}} = \frac{2}{\frac{1}{s_1} + \frac{1}{s_2}}$$

"[I]f each sub-trip covers the same distance, then the average speed is the [balanced] harmonic mean of [each] sub-trip speed." If one sub-trip covers a greater distance than the other sub-trip, "then a weighted harmonic mean is needed." To return to the F1 measure, if each of precision and recall has the same weight, then the F1 measure is the balanced harmonic mean of each of precision and recall. This balanced harmonic mean is

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

If recall has more weight than precision, then a weighted harmonic mean is needed. This weighted harmonic mean is

$$F_{w_P,\ w_R} = \frac{w_P + w_R}{\frac{w_P}{P} + \frac{w_R}{R}}$$

When searching for our optimal binary classifier, we choose as our optimal binary classifier the binary classifier with the highest F1 measure and $F_{w_P=1,\ w_R=1}$.

Regarding the weight of PPV and the weight of TPR, we prioritize TPR at least as much as PPV. We prioritize identifying as many pixels corresponding to blue tarps correctly as possible at least as much as having predictions that pixels correspond to blue tarps be correct. As we move toward providing resources for refugees, we may wish to prioritize identifying as many pixels corresponding to blue tarps as possible more than having predictions that pixels correspond to blue tarps be correct.

Data Frame For Modeling

In order to build binary classifiers, we create a composite data frame of our training data frame and our holdout data frame; substitute a column of classes for a column of indicators of whether of not a pixel depicts a blue tarp; add columns corresponding to normalized intensities and intensities normalized after transforming by the natural logarithm, the square root, the square, and interactions; and divide the composite data frame into training and holdout data frames again. We normalize as opposed to standardize intensities given that distributions of intensity are not normal. We shuffle the rows in our training and holdout data frames.

```
training_data_frame_of_indicators_and_pixels <- NULL
should_generate_training_and_holdout_data_frames_of_indicators_and_pixels <- FALSE
if (should_generate_training_and_holdout_data_frames_of_indicators_and_pixels) {
    composite_data_frame_of_classes_and_pixels <- rbind(
        training_data_frame_of_classes_and_pixels,
        holdout_data_frame_of_classes_and_pixels
    )
    number_of_training_observations <-
        nrow(training_data_frame_of_classes_and_pixels)
    number_of_holdout_observations <-
        nrow(holdout_data_frame_of_classes_and_pixels)
    number_of_training_and_holdout_observations <-
```

```r
    nrow(composite_data_frame_of_classes_and_pixels)
column_of_indicators <- rep(0, number_of_training_and_holdout_observations)
condition <- composite_data_frame_of_classes_and_pixels$Class == "Blue Tarp"
column_of_indicators[condition] <- 1
factor_of_indicators <- factor(column_of_indicators)
composite_data_frame_of_indicators_and_pixels <- data.frame(
    Indicator = factor_of_indicators,
    Normalized_Red = normalize_vector(composite_data_frame_of_classes_and_pixels[, "Red"]),
    Normalized_Green = normalize_vector(composite_data_frame_of_classes_and_pixels[, "Green"]),
    Normalized_Blue = normalize_vector(composite_data_frame_of_classes_and_pixels[, "Blue"]),
    Normalized_Natural_Logarithm_Of_Red =
        normalize_vector(log(composite_data_frame_of_classes_and_pixels[, "Red"])),
    Normalized_Natural_Logarithm_Of_Green =
        normalize_vector(log(composite_data_frame_of_classes_and_pixels[, "Green"])),
    Normalized_Natural_Logarithm_Of_Blue =
        normalize_vector(log(composite_data_frame_of_classes_and_pixels[, "Blue"])),
    Normalized_Square_Root_Of_Red =
        normalize_vector(sqrt(composite_data_frame_of_classes_and_pixels[, "Red"])),
    Normalized_Square_Root_Of_Green =
        normalize_vector(sqrt(composite_data_frame_of_classes_and_pixels[, "Green"])),
    Normalized_Square_Root_Of_Blue =
        normalize_vector(sqrt(composite_data_frame_of_classes_and_pixels[, "Blue"])),
    Normalized_Square_Of_Red =
        normalize_vector(I(composite_data_frame_of_classes_and_pixels[, "Red"]^2)),
    Normalized_Square_Of_Green =
        normalize_vector(I(composite_data_frame_of_classes_and_pixels[, "Green"]^2)),
    Normalized_Square_Of_Blue =
        normalize_vector(I(composite_data_frame_of_classes_and_pixels[, "Blue"]^2)),
    Normalized_Interaction_Of_Red_And_Green = normalize_vector(
        as.numeric(
            interaction(
                composite_data_frame_of_classes_and_pixels$Red,
                composite_data_frame_of_classes_and_pixels$Green
            )
        )
    ),
    Normalized_Interaction_Of_Red_And_Blue = normalize_vector(
        as.numeric(
            interaction(
                composite_data_frame_of_classes_and_pixels$Red,
                composite_data_frame_of_classes_and_pixels$Blue
            )
        )
    ),
    Normalized_Interaction_Of_Green_And_Blue = normalize_vector(
        as.numeric(
            interaction(
                composite_data_frame_of_classes_and_pixels$Green,
                composite_data_frame_of_classes_and_pixels$Blue
            )
        )
    )
)
```

```r
        training_data_frame_of_indicators_and_pixels <-
            composite_data_frame_of_indicators_and_pixels[1:number_of_training_observations, ]
        holdout_data_frame_of_indicators_and_pixels <-
            composite_data_frame_of_indicators_and_pixels[
                (number_of_training_observations + 1):
                number_of_training_and_holdout_observations,
            ]
        set.seed(1)
        vector_of_random_indices <- sample(1:number_of_training_observations)
        training_data_frame_of_indicators_and_pixels <-
            training_data_frame_of_indicators_and_pixels[vector_of_random_indices, ]
        set.seed(1)
        vector_of_random_indices <- sample(1:number_of_holdout_observations)
        holdout_data_frame_of_indicators_and_pixels <-
            holdout_data_frame_of_indicators_and_pixels[vector_of_random_indices, ]
        write.csv(
            training_data_frame_of_indicators_and_pixels,
            "Training_Data_Frame_Of_Indicators_And_Pixels.csv",
            row.names = FALSE
        )
        write.csv(
            holdout_data_frame_of_indicators_and_pixels,
            "Holdout_Data_Frame_Of_Indicators_And_Pixels.csv",
            row.names = FALSE
        )
} else {
        training_data_frame_of_indicators_and_pixels <-
            read.csv("Training_Data_Frame_Of_Indicators_And_Pixels.csv", header = TRUE)
        training_data_frame_of_indicators_and_pixels$Indicator <-
            factor(training_data_frame_of_indicators_and_pixels$Indicator)
        holdout_data_frame_of_indicators_and_pixels <-
            read.csv("Holdout_Data_Frame_Of_Indicators_And_Pixels.csv", header = TRUE)
        holdout_data_frame_of_indicators_and_pixels$Indicator <-
            factor(holdout_data_frame_of_indicators_and_pixels$Indicator)
}
t(training_data_frame_of_indicators_and_pixels[1, ])
```

```
#                                              1
# Indicator                                   "0"
# Normalized_Red                              "0.1622807"
# Normalized_Green                            "0.1629956"
# Normalized_Blue                             "0.1217391"
# Normalized_Natural_Logarithm_Of_Red         "0.3843573"
# Normalized_Natural_Logarithm_Of_Green       "0.3812405"
# Normalized_Natural_Logarithm_Of_Blue        "0.3235532"
# Normalized_Square_Root_Of_Red               "0.2602766"
# Normalized_Square_Root_Of_Green             "0.2595016"
# Normalized_Square_Root_Of_Blue              "0.2078738"
# Normalized_Square_Of_Red                    "0.05236718"
# Normalized_Square_Of_Green                  "0.05356392"
# Normalized_Square_Of_Blue                   "0.03391304"
# Normalized_Interaction_Of_Red_And_Green     "0.1629604"
# Normalized_Interaction_Of_Red_And_Blue      "0.1218807"
```

```
# Normalized_Interaction_Of_Green_And_Blue "0.1218836"
```

Grid Search

We conduct a grid search for an optimal binary classifier according to the *F1measure*. The primary dimension of our grid is type of classifier. We compare Logistic Regression (LR), Logistic Ridge Regression (LRR), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K Nearest Neighbors (KNN), Random Forest (RF), and Support-Vector Machine (SVM) classifiers. Generally speaking, LR, LRR, LDA, and QDA classifiers are relatively inflexible with high bias and low variance; KNN, RF, and SVM classifiers are relatively flexible with low bias and high variance. KNN classifiers may be flexible when K is approximately 1 or inflexible as K approaches the number of observations in our training data set n.

The secondary dimension of our grid is the set of predictive terms. We consider classifiers with the terms in the above output other than *Indicator*. Per University Of Virginia courses Linear Models For Data Science and Introduction To Statistical Learning, these predictive terms involve common transformations of predictors.

The third dimension of our grid is value of primary hyperparameter. For LRR classifiers, the primary hyperparameter is λ. LRR classifiers are penalized for inclusion of predictive terms proportionally to λ; setting λ to be greater than 0 may decrease the variance and increase the performance of LRR classifiers. For KNN classifiers, the primary hyperparameter is K. The class of a test observation depends on determining the classes of the K nearest neighboring observations in the training data set. A KNN classifier with $K = 1$ is relatively flexible with low bias and high variance; such a model may overfit the training data and identify patterns with performance that is less than ideal. A KNN classifier with $K = n$ predicts that all observations have one class and has high bias and low variance. A KNN classifier with $1 < K < n$ may perform best.

The fourth dimension of our grid is value of threshold. Varying the threshold according to which a classifier classifies an observations as corresponding to a blue tarp or not will result in different numbers of false negatives, false positives, true negatives, and true positives and different F1 measures. We seek to maximize F1 measure.

According to *An Introduction to Statistical Learning (Second Edition)* by James et al., for models with categorical responses such as binary classifiers, "k-fold cross validation involves randomly dividing a set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds." A performance metric like F1 measure is "computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the" performance metric, $F_1, F_2, ..., F_k$. "The k-fold cross validation estimate is computed by averaging these values".

"[W]e are interested only in the location of the [maximum] point in the estimated test [F1 measure] curve. This is because we might be performing cross-validation on a number of statistical learning methods, or on a single method using different [hyperparameters], in order to identify the method that results in the [highest F1 measure. D]espite the fact that they sometimes [misestimate] the true test [F1 measure, most] of the CV curves [of F1 measure vs. threshold] come close to identifying the correct [threshold] corresponding to the [highest F1 measure]."

"[T]here is some variability in the CV estimates as a result of the variability in how the observations are divided into ten folds. But this variability is typically much lower than the variability in the test [F1 measures] that result from the validation set approach."

As described below, we apply cross validation to averaging performance metrics and choosing λ for LRR classifiers and K for KNN classifiers. We apply cross validation to averaging performance metrics using `rsample::vfold_cv` by passing `vfold_cv` our full training data frame of indicators and pixels and specifications of numbers of partitions of the data set and times to repeat v-fold partitioning. We apply cross validation to choosing λ and K using `caret::train` by passing `train` a `caret::trainControl` constructed with `method = "cv"`.

To search for an optimal LR classifier, we perform manual bidirectional selection. For each formula, we perform 10-fold cross validation. For each fold, we record 100 thresholds and corresponding performance metrics including F1 measure. For each formula, we record 100 thresholds and corresponding average performance metrics. For each formula, we find the maximum average F1 measure. For each formula, we choose the threshold that yields the highest maximum average F1 measure. We choose the LR classifier with the formula and threshold with the highest maximum average F1 measure. For our optimal LR classifier with optimal formula and optimal threshold, we provide a plot of Average Performance Metrics Vs. Threshold, a Precision-Recall Curve, an ROC curve, and a data frame of optimal performance metrics.

To describe bidirectional selection in detail, we determine the maximum average F1 measure for LR classifiers with formula $Indicator \sim normalize(Red)$. Similarly, we calculate the maximum average F1 measure for LR classifiers with formula $Indicator \sim normalize(Green), Indicator \sim normalize(Blue), ..., Indicator \sim normalize\left(Blue^2\right)$. The LR classifiers with formula $Indicator \sim normalize(Blue)$ have the highest maximum average F1 measure of 0.445. We consider the maximum average F1 measures for LR classifiers with formulas $Indicator \sim normalize(Blue) + normalize(Red), ..., Indicator \sim normalize(Blue) + normalize(Green : Blue)$. The LR classifiers with formula $Indicator \sim normalize(Blue) + normalize(Red)$ have the highest maximum average F1 measure of 0.931. We have already considered the maximum average F1 measures for LR classifiers with formulas $Indicator \sim normalize(Red)$ and $Indicator \sim normalize(Blue)$. We consider the maximum average F1 measures for LR binary classifiers with formulas $Indicator \sim normalize(Blue) + normalize(Red) + normalize(Green), ..., Indicator \sim normalize(Blue) + normalize(Red) + normalize(Green : Blue)$. The LR classifiers with formula $Indicator \sim normalize(Blue) + normalize(Red) + normalize\left(Green^2\right)$ have the highest maximum average F1 measure of 0.942. The maximum average F1 measure for LR classifiers with formula $Indicator \sim normalize(Red) + normalize\left(Green^2\right)$ is 0.449. We consider the maximum average F1 measures for LR 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 maximum average F1 measures greater than 0.942.

Our optimal LR classifier has formula $Indicator \sim normalize(Blue) + normalize(Red) + normalize\left(Green^2\right)$. The maximum average F1 measure is 0.942; the corresponding threshold is 0.18.

Average accuracy is nearly 1 for about 90 percent of thresholds. Average F1 measure is nearly 0.942 for thresholds between about 0.1 and about 0.4. Average PPV increases linearly from a threshold of about 0.1 to 1. Average TPR increases linearly from a threshold of about 0.95 to 0. A Precision-Recall curve has an AUC of 0.984. An ROC curve has an AUC of 0.999.

```
set.seed(1)
summary_of_performance <- summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "Random Forest",
    formula = Indicator ~ Normalized_Red + Normalized_Green + Normalized_Blue,
    data_frame = training_data_frame_of_indicators_and_pixels
)
```

```
# [1] "Summary for model of type Random Forest"
# [1] "Indicator ~ "
# [1] "    Normalized_Red"
# [1] "    + Normalized_Green"
# [1] "    + Normalized_Blue"


# [1] "Minimum test error rate: 0.00205566097406705"
# [1] "Value of mtry corresponding to minimum test error rate: 2"
# [1] "Number of trees corresponding to minimum test error rate: 5"


# Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
```
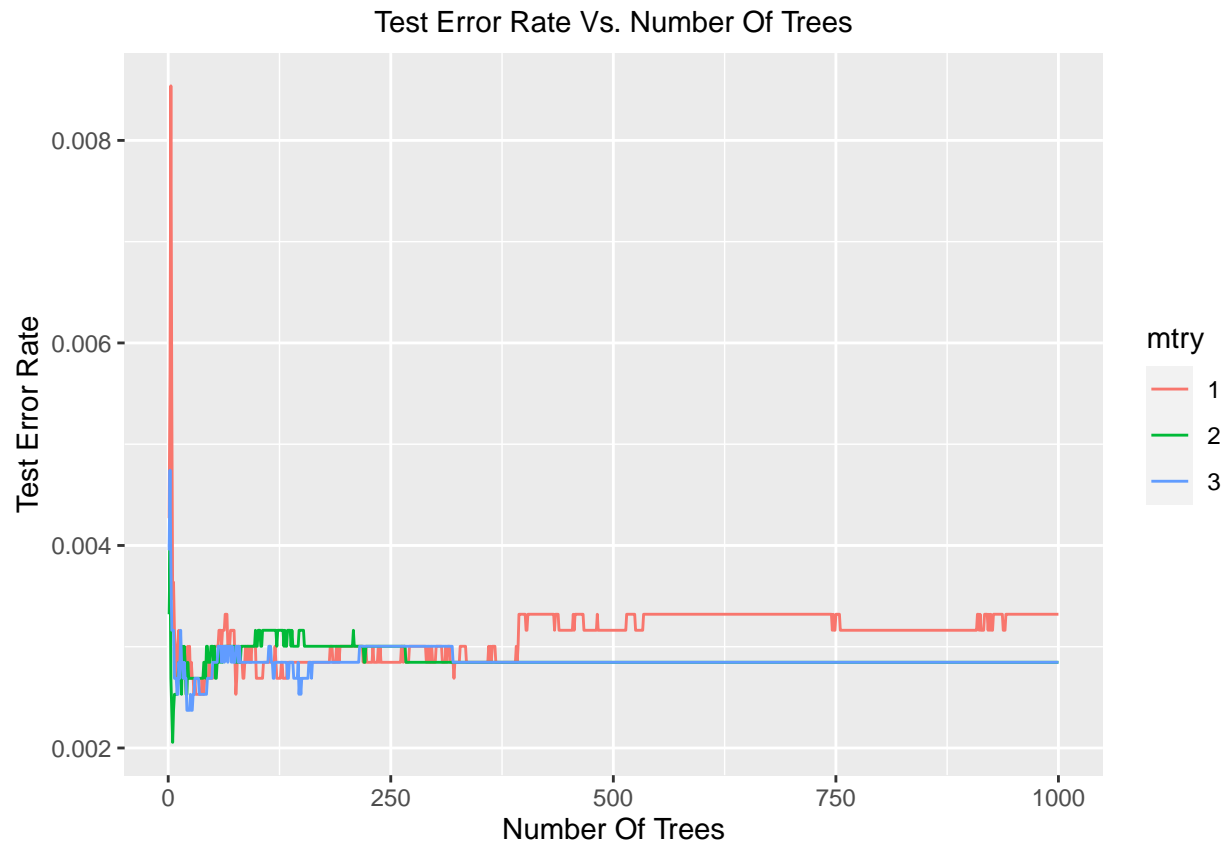
```
# collapsing to unique 'x' values

# Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
# collapsing to unique 'x' values
```

Test Error Rate Vs. Number Of Trees


```
summary_of_performance$plot_of_performance_metrics_vs_threshold
```
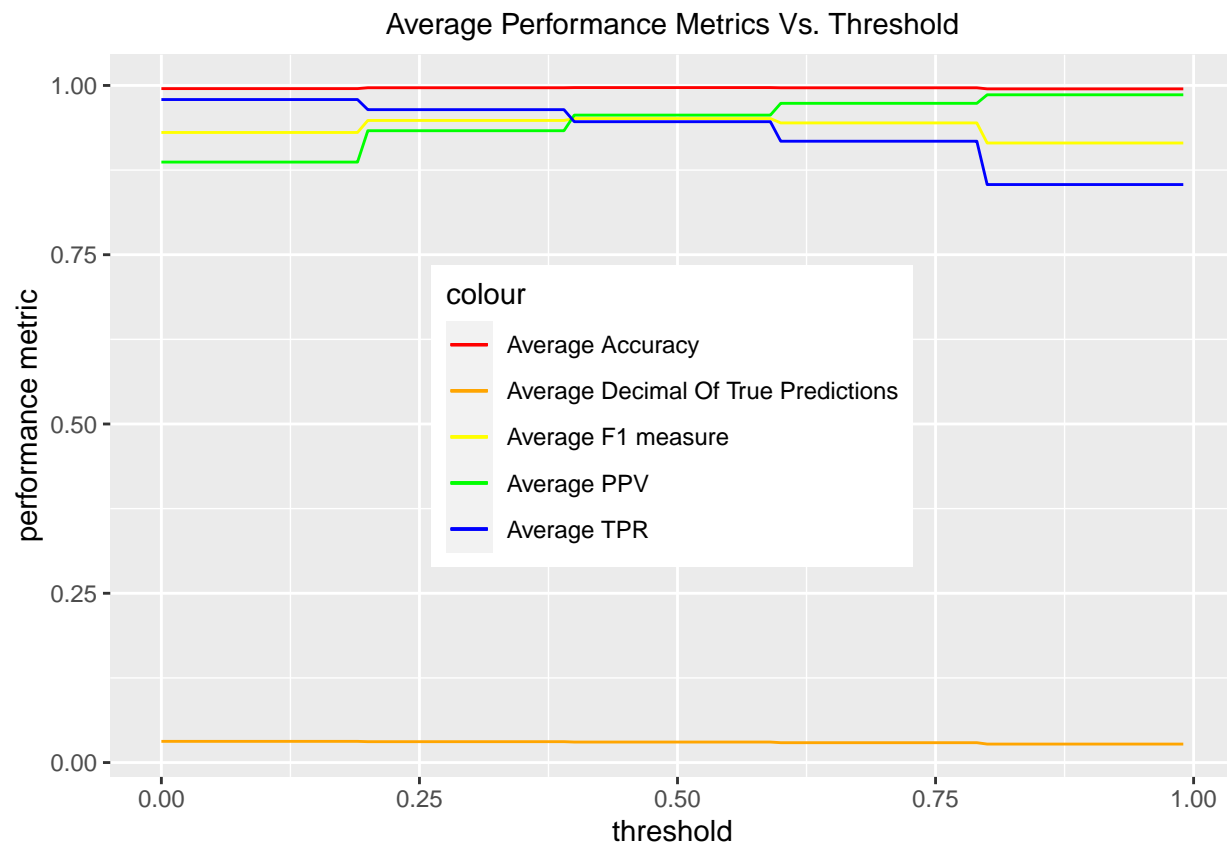
Figure 6: Prediction Metrics Vs. Thresholds For Random Forest Binary Classifier With Formula Indicator vs. Normalized Red + Normalized Green + Normalized Blue

```
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
```

```
#                              [,1]
# corresponding_threshold 0.400000000
# alpha                            NA
# optimal_lambda                   NA
# optimal_K                        NA
# optimal_PR_AUC          0.978220080
# optimal_ROC_AUC         0.989221289
# corresponding_accuracy  0.996884923
# corresponding_TPR       0.946444717
# corresponding_FPR       0.001454342
# corresponding_PPV       0.956196080
# optimal_F1_measure      0.951165702
```

```
data_frame_of_fold_ids_and_maximum_F1_measures_for_Random_Forest <-
    summary_of_performance$data_frame_of_fold_ids_and_maximum_F1_measures
```
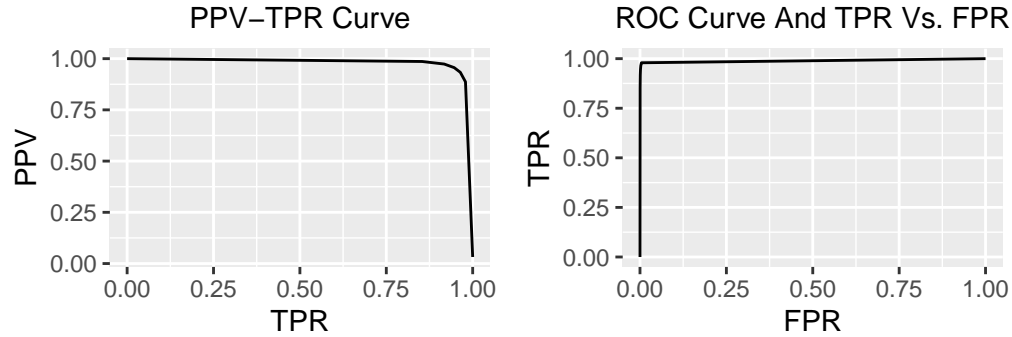
Figure 7: PR And ROC Curves For Random Forest Binary Classifier With Formula Indicator vs. Normalized Red + Normalized Green + Normalized Blue

```r
# "glm.fit: fitted probabilities numerically 0 or 1" occurs due to predicted
# probabilities of the Logistic Regression binary classifier being close to 0 and 1.
# "collapsing to unique 'x' values" is an artifact of MESS::auc.
set.seed(1)
summary_of_performance <- 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 = training_data_frame_of_indicators_and_pixels
)
```

```r
summary_of_performance$plot_of_performance_metrics_vs_threshold
```

```r
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
data_frame_of_fold_ids_and_maximum_F1_measures_for_Logistic_Regression <-
    summary_of_performance$data_frame_of_fold_ids_and_maximum_F1_measures
```

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 $\lambda = 0.000123$ 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 Ridge Regression. We may 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.

13

Unfortunately, `glmnet` chooses a sequence of values of *lambda* with a minimum of 0.0035. `caret::train` considers the optimal value of λ to be this minimum. We assume that the true optimal value of *lambda* is less than 0.0035. We use `glmnet::cv.glmnet` to graph *Misclassification ErrorVs.ln*(λ) for $-9 < ln(-\lambda) < -8$. Misclassification error is an approximation of F1 measure. Our software fails to provide graphs of Performance Metrics Vs. Threshold and PR and ROC curves for some values of $ln(\lambda)$ less than -9. We assume that the optimal value of λ is less than $exp(-9) = 0.000123$.

Our optimal Logistic Ridge Regression binary classifier has formula *Indicator* $\sim$ *normalize*(*Blue*) + *normalize*(*Red*) + *normalize*$\left(Green^2\right)$. Our optimal value for λ is 0.000123; the maximum average F1 measure is 0.934; the corresponding threshold is 0.18.

Average accuracy is nearly 1 for about 90 percent of thresholds but in general is not quite as good as for our optimal Logistic Regression binary classifier. Average F1 measure is nearly 0.934 for maybe 6 percent of thresholds; the maximum average F1 measure is lower than that for our optimal Logistic Regression binary classifier and about the maximum for fewer thresholds. Average PPV increases logarithmically from a threshold of about 0.1 to 1. From about the optimal threshold PPV is greater than that for our optimal Logistic Regression binary classifier. Average TPR increases in the manner of $y = sqrt(x)$ from a threshold of about 1 to 0. Average TPR is generally less than that for our optimal Logistic Regression binary classifier. We prioritize TPR over precision but under F1 measure. A Precision-Recall curve has an AUC of 0.981. An ROC curve has an AUC of 0.999. These AUC's are less than those for our optimal Logistic Regression binary classifier.

```
formula <-
    Indicator ~ Normalized_Natural_Logarithm_Of_Blue + Normalized_Square_Root_Of_Red
full_model_matrix <-
    model.matrix(object = formula, data = data_frame_of_indicators_and_pixels)[, -1]
sequence_of_lambda_values <- exp(seq(-9, -8, length = 100))
the_cv.glmnet <- glmnet::cv.glmnet(
    x = full_model_matrix,
    y = data_frame_of_indicators_and_pixels$Indicator,
    family = "binomial",
    type.measure = "class",
    alpha = 0,
    lambda = sequence_of_lambda_values
)
plot(the_cv.glmnet)
#Gedeck, Peter (tux3md) Thursday, July 6, 2023 6:54 PM
#In class, the question was raised on how to create the lambda tuning graph for cv.glmnet using ggplot2
#df = data.frame(
#    lambda=cvob$lambda,
#    cvm=cvob$cvm,
#    cvup=cvob$cvup,
#    cvlo=cvob$cvlo
#)
#ggplot(df, aes(x=lambda, y=cvm, ymin=cvlo, ymax=cvup)) +
#    geom_point() +
#    scale_x_log10() +
#    geom_errorbar()
optimal_lambda <- the_cv.glmnet$lambda.min
optimal_lambda
summary_of_performance <- summarize_performance_of_cross_validated_models_using_dplyr(
    type_of_model = "Logistic Ridge Regression",
    formula = formula,
    data_frame = data_frame_of_indicators_and_pixels,
    optimal_lambda = optimal_lambda
```

```
)
```

```
summary_of_performance$plot_of_performance_metrics_vs_threshold
```

Average Performance Metrics Vs. Threshold

Figure 8: Prediction Metrics Vs. Thresholds For Logistic Ridge Regression Model With Formula Indicator Vs. Normalized Natural Logarithm Of Blue + Normalized Square Root Of Red

```
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
```

```
#                              [,1]
# corresponding_threshold 0.400000000
# alpha                            NA
# optimal_lambda                   NA
# optimal_K                        NA
# optimal_PR_AUC         0.978220080
# optimal_ROC_AUC        0.989221289
# corresponding_accuracy 0.996884923
# corresponding_TPR      0.946444717
# corresponding_FPR      0.001454342
# corresponding_PPV      0.956196080
# optimal_F1_measure     0.951165702
```
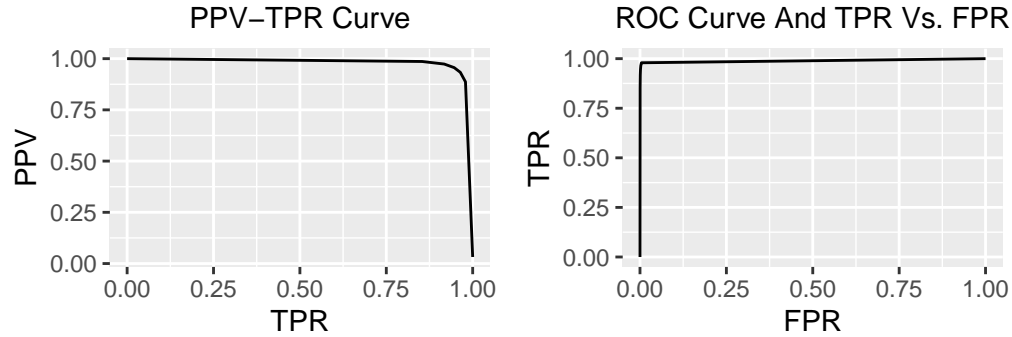
15

Figure 9: PR And ROC Curves For Logistic Ridge Regression Model With Formula Indicator Vs. Normalized Natural Logarithm Of Blue + Normalized Square Root Of Red

To search for an optimal LDA binary classifier, we perform manual bidirectional selection as above. Our optimal LDA binary classifier has formula $Indicator \sim normalize(Blue) + normalize(Red^2) + normalize\left(Green^2\right)$. The maximum average F1 measure is 0.891; the corresponding threshold is 0.69.

Average accuracy is nearly 1 for about 95 percent of thresholds but in general is not quite as good as that for our optimal Logistic Regression binary classifier or that for our optimal Logistic Ridge Regression binary classifier. Average F1 measure is nearly 0.891 for maybe 6 percent of thresholds; the maximum average F1 measure is lower than that for our optimal Logistic Regression binary classifier and that for our optimal Logistic Ridge Regression binary classifier. The maximum average F1 measure is about 0.891 for about the same percent of thresholds as for our optimal Logistic Ridge Regression binary classifier. Average PPV increases logarithmically from a threshold of about 0 to our optimal threshold of 0.69, at which point average PPV jumps up to 1. From about a threshold of 0.69 PPV is greater than that for our optimal Logistic Regression binary classifier and equal to that for our optimal Logistic Ridge Regression binary classifier. For thresholds lower than our optimal threshold of 0.69 PPV is less than those for our optimal Logistic Regression and Logistic Ridge Regression binary classifiers. Average TPR generally has a similar trend and is less than that for our optimal Logistic Regression binary classifier. Average TPR is generally less than that for our optimal Logistic Ridge Regression binary classifier for thresholds less than 0.38 and greater than that for our optimal Logistic Ridge Regression binary classifier for thresholds greater than 0.38. We prioritize TPR over precision but under F1 measure. A Precision-Recall curve has an AUC of 0.903. An ROC curve has an AUC of 0.949. These AUC's are less than those for our optimal Logistic Regression and Logistic Ridge Regression binary classifiers.

```
summary_of_performance <- 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
)


summary_of_performance$plot_of_performance_metrics_vs_threshold
```

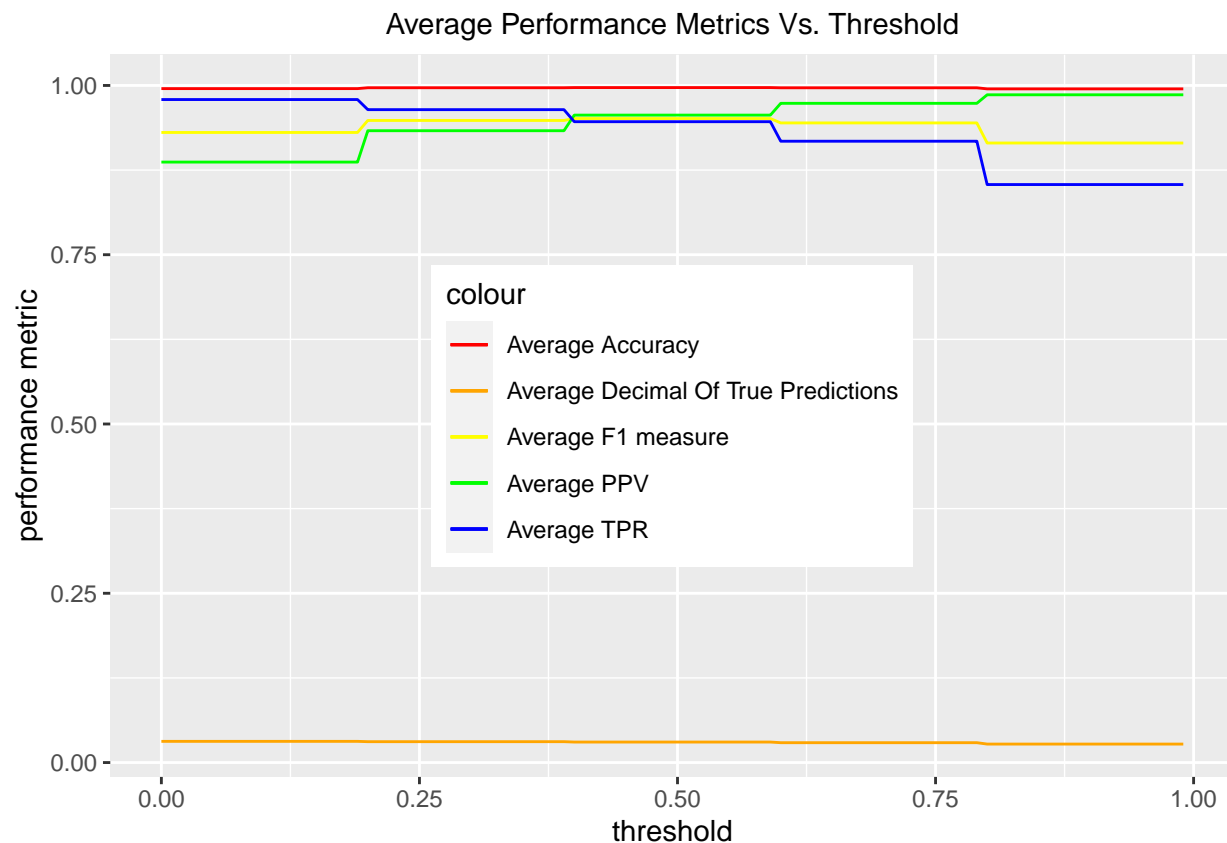## Average Performance Metrics Vs. Threshold

Figure 10: Prediction Metrics Vs. Thresholds For Linear Discriminant Analysis Model With Formula Indicator Vs. Normalized Blue + Normalized Square Of Red + Normalized Square Of Green

```
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
```

```
#                              [,1]
# corresponding_threshold 0.400000000
# alpha                            NA
# optimal_lambda                   NA
# optimal_K                        NA
# optimal_PR_AUC          0.978220080
# optimal_ROC_AUC         0.989221289
# corresponding_accuracy  0.996884923
# corresponding_TPR       0.946444717
# corresponding_FPR       0.001454342
# corresponding_PPV       0.956196080
# optimal_F1_measure      0.951165702
```
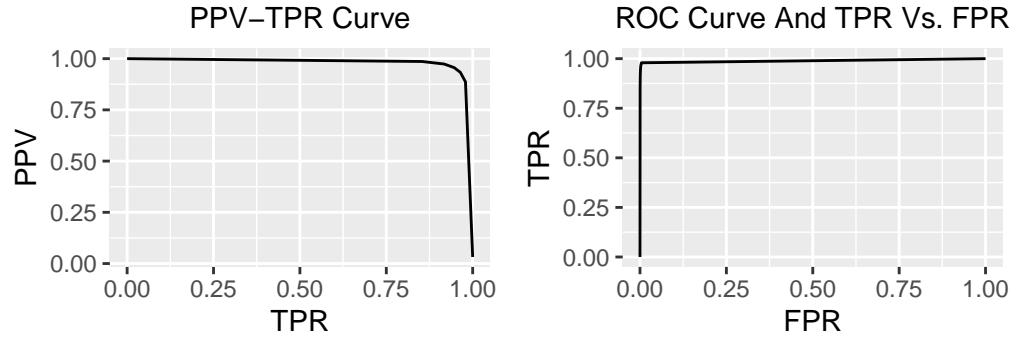
Figure 11: PR And ROC Curves For Linear Discriminant Analysis Model With Formula Indicator Vs. Normalized Blue + Normalized Square Of Red + Normalized Square Of Green

To search for an optimal QDA binary classifier, we perform manual bidirectional selection as above. Our optimal QDA binary classifier has formula $Indicator \sim normalize\left(Blue^2\right) + normalize\left(Red^2\right) + normalize\left(Red\right)$. The maximum average F1 measure is 0.939; the corresponding threshold is 0.960.

Average accuracy is nearly 1 for about 95 percent of thresholds but in general is not quite as good as that for our optimal Logistic Regression binary classifier or that for our optimal Logistic Ridge Regression binary classifier. In generally average accuracy is better than for our LDA binary classifier. Average F1 measure is nearly 0.939 for few thresholds; the maximum average F1 measure is lower than that for our optimal Logistic Regression binary classifier and that for our optimal Logistic Ridge Regression binary classifier but higher than that for our LDA binary classifier. Average PPV increases in the manner of $y = tan(x)$ from a threshold of about 0 to 1. In general PPV is less than that for one of the preceding models. Average TPR is nearly 1 for thresholds less than 0.75. We prioritize TPR over precision but under F1 measure. A Precision-Recall curve has an AUC of 0.966. An ROC curve has an AUC of 0.998. These AUC's are less than those for our optimal Logistic Regression and Logistic Ridge Regression binary classifiers but greater than that for our LDA binary classifier.

```
summary_of_performance <- 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
)
```

```
summary_of_performance$plot_of_performance_metrics_vs_threshold
```
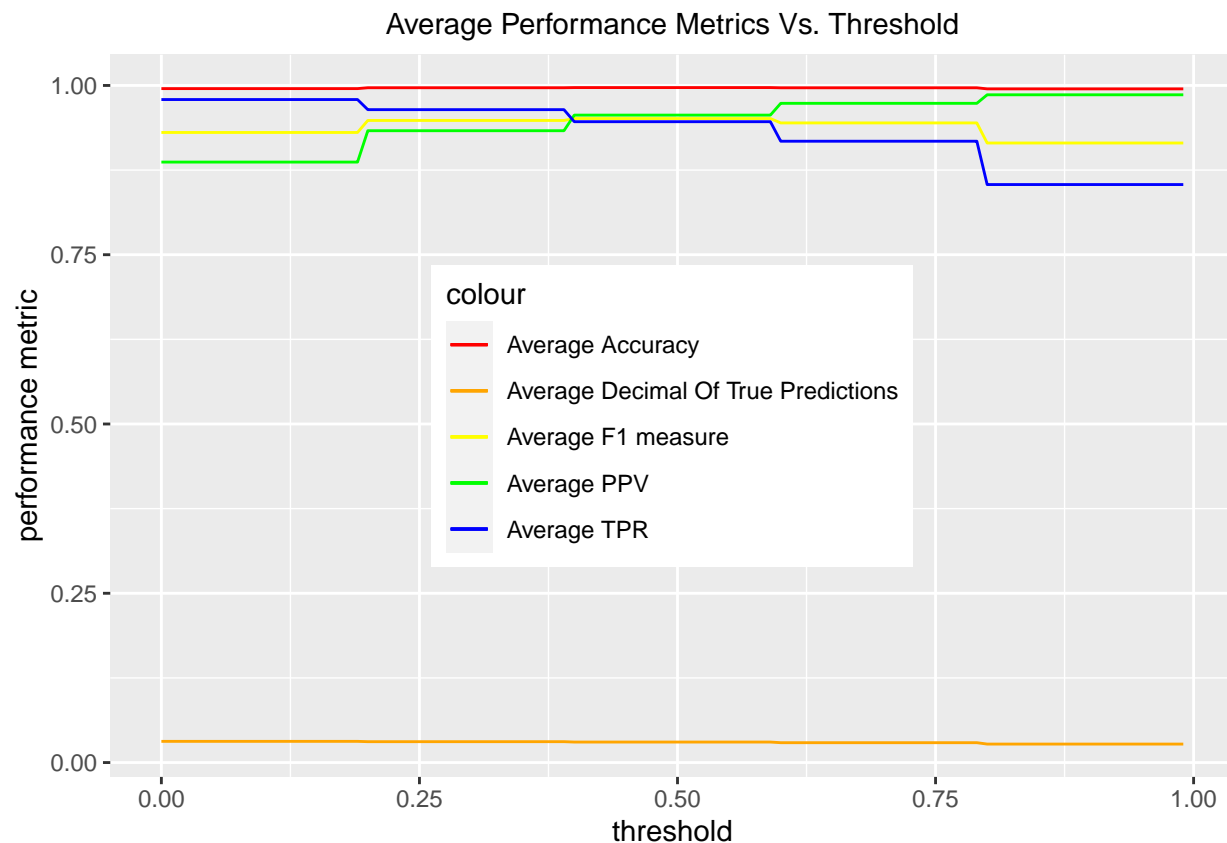
Figure 12: Prediction Metrics Vs. Thresholds For Quadratic Discriminant Analysis Model With Formula Indicator Vs. Normalized Square Of Blue + Normalized Square Of Red + Normalized Red + Normalized Natural Logarithm Of Blue

```
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
```

```
#                              [,1]
# corresponding_threshold 0.400000000
# alpha                           NA
# optimal_lambda                  NA
# optimal_K                       NA
# optimal_PR_AUC          0.978220080
# optimal_ROC_AUC         0.989221289
# corresponding_accuracy  0.996884923
# corresponding_TPR       0.946444717
# corresponding_FPR       0.001454342
# corresponding_PPV       0.956196080
# optimal_F1_measure      0.951165702
```
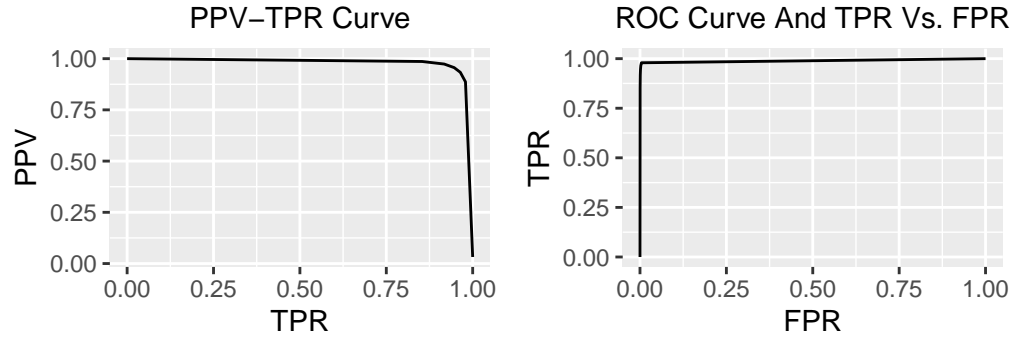
Figure 13: PR And ROC Curves For Quadratic Discriminant Analysis Model With Formula Indicator Vs. Normalized Square Of Blue + Normalized Square Of Red + Normalized Red + Normalized Natural Logarithm Of Blue

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 $K = 1$ 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 $K \in [1, 25]$. 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 observations used to determine the probability that a testing observation corresponds to a blue tarp.

See below graph of average F1 measure vs. K for 10 cross-validated KNN binary classifiers trained on the full data frame of indicators and pixels. `caret::train` chose the value of K for all KNN binary classifiers corresponding to the maximum average F1 measure.

Our optimal KNN binary classifier has formula $Indicator \sim normalize(Red : Blue) + normalize(Green : Blue) + normalize\left(\sqrt{Blue}\right)$. The maximum average F1 measure is 0.946; the corresponding threshold is 0.080.

Average accuracy is nearly 1 for about 100 percent of thresholds and is better than that for any of the previous binary classifiers. Average F1 measure is nearly 0.946 for 90 percent of thresholds; the maximum average F1 measure is higher than that for any previous binary classifier. Average F1 measure, PPV, and TPR all jump to about 0.942 for a threshold of about 0.1. A Precision-Recall curve has an AUC of 0.948. An ROC curve has an AUC of 0.997. These AUC's are less than those for our optimal Logistic Regression, Logistic Ridge Regression, and QDA binary classifiers but greater than that for our LDA binary classifier.

```
summary_of_performance <- 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_B
    data_frame = data_frame_of_indicators_and_pixels
)
```

```
summary_of_performance$plot_of_performance_metrics_vs_threshold
```
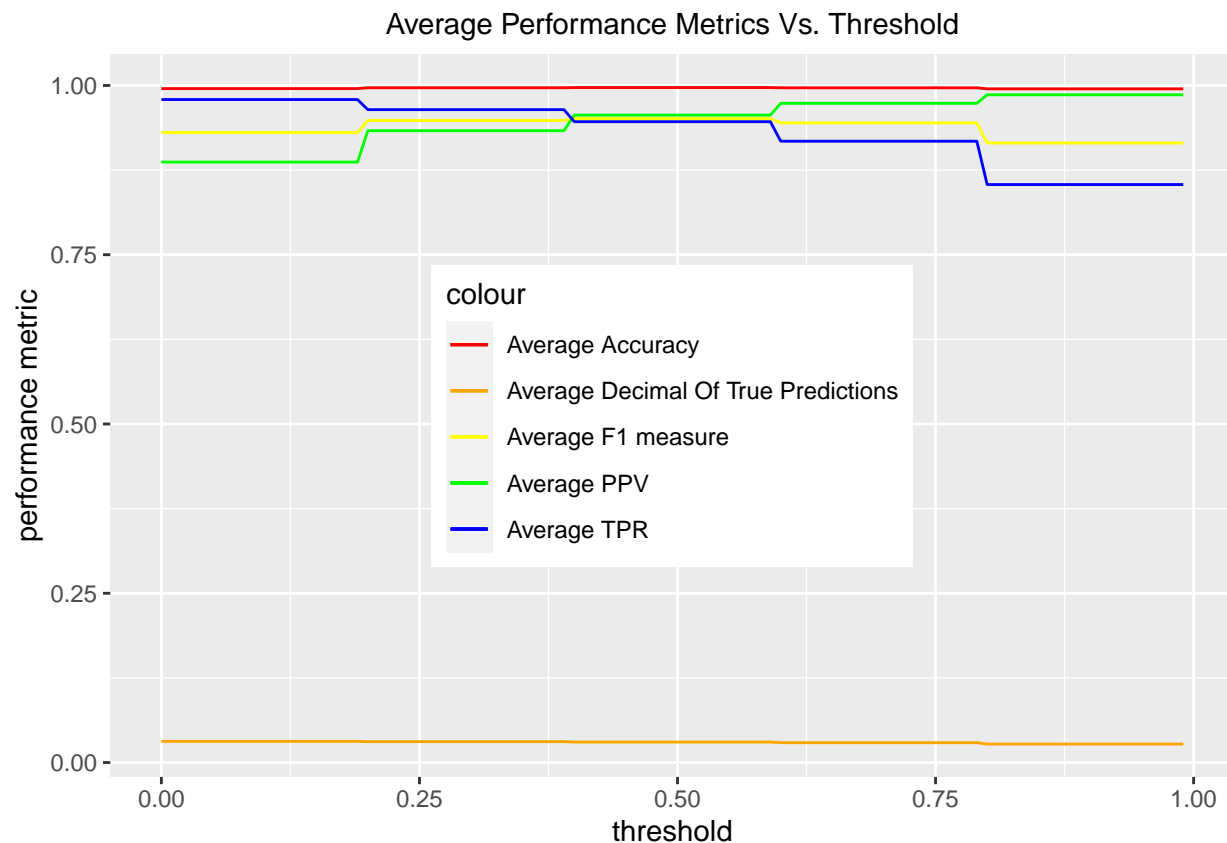
Figure 14: Prediction Metrics Vs. Thresholds For KNN Model With Formula Indicator Vs. Normalized Interaction Of Red And Blue And Normalized Interaction Of Green And Blue

```
summary_of_performance$PR_curve
summary_of_performance$ROC_curve
t(summary_of_performance$data_frame_of_optimal_performance_metrics)
```

```
#                            [,1]
# corresponding_threshold 0.400000000
# alpha                            NA
# optimal_lambda                   NA
# optimal_K                        NA
# optimal_PR_AUC          0.978220080
# optimal_ROC_AUC         0.989221289
# corresponding_accuracy  0.996884923
# corresponding_TPR       0.946444717
# corresponding_FPR       0.001454342
# corresponding_PPV       0.956196080
# optimal_F1_measure      0.951165702
```

```
data_frame_of_fold_ids_and_maximum_F1_measures_for_KNN <-
    summary_of_performance$data_frame_of_fold_ids_and_maximum_F1_measures
```
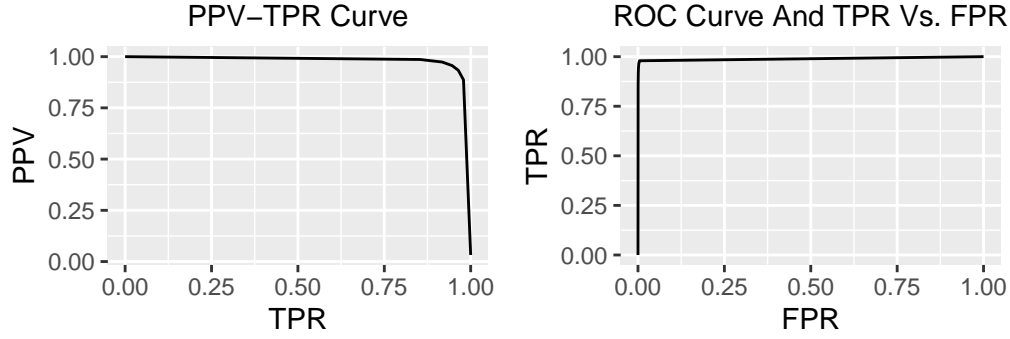
Figure 15: PR And ROC Curves For KNN Model With Formula Indicator Vs. Normalized Interaction Of Red And Blue And Normalized Interaction Of Green And Blue

Performance Table

Per our Performance Table, our KNN binary classifier with $K = 1$ and threshold $t = 0.080$ has the best optimal F1 measure and corresponding accuracy. Our KNN binary classifier has the second-best corresponding TPR, FPR, and PPV. However, our KNN binary classifier has the second-worst Area Under The Precision-Recall Curve and Area Under The ROC Curve. Our Logistic Regression binary classifier has the best Area Under The Precision-Recall Curve, Area Under The ROC Curve, and corresponding TPR. Our Logistic Regression binary classifier has the second-best corresponding optimal measure and corresponding accuracy. However, our Logistic Regression binary classifier has the worst corresponding FPR and PPV. Beautifully, our QDA binary classifier is completely middle of the road. Our Logistic Ridge Regression binary classifier is second-best in Area Under The Precision-Recall Curve and Area Under The ROC curve and second-worst in everything else. Our LDA binary classifier has the best corresponding FPR and PPV and is worst in everything else.

We calculate binary-classifier scores for each type of binary classifier. Each binary-classifier score is a weighted sum of the first 7 colored performance metrics above it in the Performance Table.

Quantity	Method Of Determination	Logistic Regression	Logistic Ridge Regression	LDA	QDA	KNN
corresponding threshold	value corresponding to optimal F1 measure	0.1800	0.1800	0.6900	0.9600	0.0800
alpha	minimum	-	0.0000	-	-	-
optimal lambda	lambda in sequence provided by glmnet::cv.glmnet	-	0.0001	-	-	-
optimal K	K in sequence provided by Peter Gedeck corresponding to maximum F1 measure	-	-	-	-	1
optimal Area Under The Precision-Recall Curve	average	0.9844	0.9811	0.9037	0.9656	0.9486
optimal Area Under The ROC Curve	average	0.9993	0.9992	0.9494	0.9983	0.9772
corresponding accuracy	value corresponding to optimal F1 measure	0.9962	0.9957	0.9937	0.9961	0.9965
corresponding True Positive Rate	value corresponding to optimal F1 measure	0.9667	0.9444	0.8097	0.9369	0.9457
corresponding False Positive Rate	value corresponding to optimal F1 measure	0.0029	0.0026	0.0002	0.0019	0.0018
corresponding precision	value corresponding to optimal F1 measure	0.9179	0.9232	0.9916	0.9412	0.9461
optimal F1 measure	maximum average	0.9416	0.9335	0.8912	0.9389	0.9458
binary-classifier score	weighted sum	25	18	15	21	26

Figure 16: Performance Table

Conclusions

Our KNN binary classifier has the highest binary classifier score, the highest optimal F1 measure, and the highest corresponding accuracy among our optimal binary classifiers. On these bases, we recommend our KNN binary classifier.

That being said, we determined the optimal binary classifier for each type of binary classifier based on an industry standard; i.e., the use of F1 measure. Leaders responsible for deploying resources to refugees may wish to replicate this research by choosing optimal binary classifiers for each type of binary classifier based

on an F measure where PPV and TPR have different weights. Leaders may wish to choose thresholds for a binary classifier that results in greater TPR than the TPR corresponding to the maximum average F1 measure for that binary classifier. In such a case our Logistic Regression binary classifier may outperform our KNN binary classifier.

Results may improve by hybridizing our Logistic Ridge Regression binary classifier with a Logistic Lasso Regression binary classifier by specifying *alpha* to be greater than 0 and less than or equal to 1. Results may improve by developing software that can find the minimum in a relatively continuous plot of F1 measure Vs. $log(\lambda)$. Results may improve by considering values of K between 1 and the number of observations in our full data frame of indicators and pixels n.

According to Data Scientist Stephen Allwright, "A good F1 score is dependent on the data you are working with and the use case. For example, a model predicting the occurrence of a disease would have a very different expectation than a custometer churn model. However, there is a general rule of thumb when it comes to F1 scores, which is as follows:"

- An F1 score greater than 0.9 is very good.
- An F1 score of 0.8 to 0.9 is good.
- An F1 score of 0.5 to 0.8 is okay.
- An F1 score less than 0.5 is not good.

All of our binary classifiers are adequately performing. Our Logistic Regression and KNN binary classifiers have a statistically insignificant difference in average maximum F1 measures. Following Text-Mining Evaluation: Statistical Tests, we train and test our Logistic Regression and KNN binary classifiers using 10-fold cross validation. We use the same folds for both systems. We calculate maximum F1 measures for each held-out fold. We construct a data frame of fold ID's and maximum F1 measure for each binary classifier and fold. We calculate the average maximum F1 measures across held-out folds. We consider the difference between the average maximum F1 measures to be a test statistic. We consider an alternate hypothesis to be that our Logistic Regression binary classifier has a higher average maximum F1 measure than our KNN binary classifier for all possible test sets. We consider a null hypothesis to be that our Logistic Regression binary classifier has an equal average maximum F1 measure to that of our KNN binary classifier. We consider a p value to be the probability of observing that the difference in average maximum F1 measures for a random sample / test set is greater than our test statistic. We consider our confidence level to be $CL = 0.95$ and our significance level to be $\alpha = 1 - CL = 0.05$. If the p value is greater than or equal to the significance level α, we fail to reject the null hypothesis that our Logistic Regression binary classifier has an average maximum F1 measure equal to that of our KNN binary classifier. If the p value is less than or equal to the significance level α, we reject the null hypothesis and have sufficient evidence to support the alternate hypothesis that our Logistic Regression binary classifier has an average maximum F1 measure greater than that of our KNN binary classifier.

To calculate our p value, we conduct Fisher's Randomization Test. We calculate our p value as the ratio of a count C and a number of iterations I. Our count C is the number of iterations $i \leq I$ for which a modified difference between an average maximum F1 measure for Logistic Regression and an average maximum F1 for KNN is greater than our test statistic. For each iteration, an average maximum F1 measure for Logistic Regression is computed effectively by tossing a fair coin for each row in our data frame of fold ID's and maximum F1 measures, if heads swapping the maximum F1 measures for the present fold, and calculating the mean of the maximum F1 measures in the column corresponding to Logistic Regression. An average maximum F1 measure for KNN is computed similarly. A modified difference between the average maximum F1 measure for Logistic Regression and the average maximum F1 measure for KNN is compared to our test statistic. If the modified difference is greater than our test statistic, count C is incremented. Our p value is the ratio of count C and number of iterations I.

The p value for our Logistic Regression and KNN binary classifiers is 0.4. The probability of observing a random test statistic greater than our test statistic is 0.4. Because our p value is greater than our significance level, we cannot confidently say that our test statistic is not due to random chance. Our difference between the average maximum F1 measures of our Logistic Regression and KNN models is statistically insignificant.

```
calculate_probability_for_randomization_test(
    data_frame_of_fold_ids_and_maximum_F1_measures_for_Logistic_Regression,
    data_frame_of_fold_ids_and_maximum_F1_measures_for_KNN
)
```

We are able to use predictive modeling tools as our data set of classes and pixels may be represented as a complete data frame of natural numbers. We are able to use classifiers as the data in column *Class* is nominal. We are able to use binary classifiers as pixels representing blue tarps exist in a distinct subspace defined by intensities of color *Red*, *Green*, and *Blue*. We are able to normalize intensities of color *Red*, *Green*, and *Blue*.

Our data set seems particularly well-suited to the class of prediction methods Binary Classifier as our data set is complete, observations may be represented as natural numbers, class of pixel is nominal, and pixels representing blue tarps exist in a distinct subspace defined by intensities of color *Red*, *Green*, and *Blue*.

Given that according to Stephen Allwright the maximum average F1 measures of our binary classifiers are very good, our research may be very effective in terms of locating people displaced by the earthquake in Haiti in 2010 and helping to save human life.