Disaster Relief Project

Spring 2023: DS 6030

Abigail Snyder

Introduction

The purpose of this project is to identify a method for locating displaced persons during a national disaster from aerial imagery, such as in the 2010 earthquake in Haiti.

To accomplish this, machine learning models were fit to training data from the HaitiPixels.csv file, which was collected from Haiti shortly following the earthquake. The data contains aerial imagery of Haiti which is separated into Red, Green, and Blue values and categorized by what those pixel groups were identified as (either Vegetation, Rooftop, Blue Tarp, Soil, or Various Non-Tarp). It was known that many of the displaced people were using blue tarps to create shelters, and so the goal of fitting each model is to use the image data in order to predict the locations of the blue tarps.

After training the models, holdout data was introduced, which contained additional aerial imagery in various .txt files, as well as several image files themselves. After appropriately cleaning and organizing the data in order to identify which columns mapped to Red, Green, and Blue values and labeling the data according to whether the pixel was classified as containing a blue tarp, each selected model was tested on the holdout data in order to confirm its validity for predicting the locations of the blue tarps.

Exploratory Data Analysis

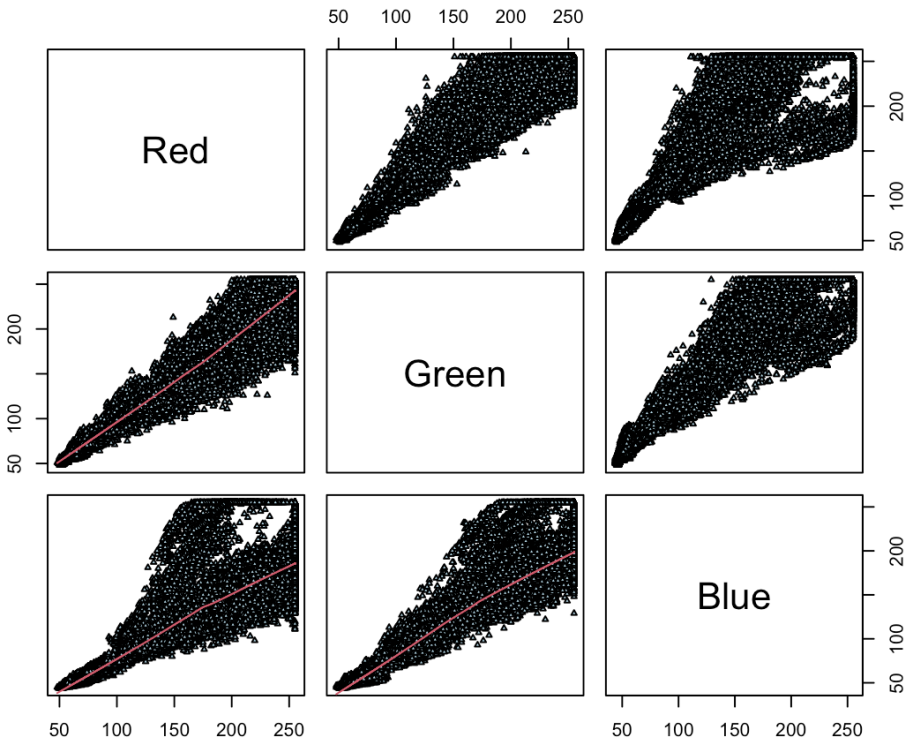
To begin, I loaded the necessary packages in R, and loaded the training dataset from the csv file. The data included four fields, “Class”, “Red”, “Blue, and”Green.” The first column, “Class” indicates the classification of the pixel, while the other three columns indicate the RGB (Red, Blue, Green) color quantities of each pixel. The color values are what will be used as predictors in each of the models.

Because I am only interested in the location of the blue tarps, I first create a new variable containing only the classification of the blue tarps, called BlueClass, which will serve as the response variable.

Before fitting any kind of model to the data, it is first necessary to get a cursory understanding of the information the data contains. To do that, I started by exploring some summary information, and then creating some visualizations to help better understand the data.

```
##     Class                Red             Green            Blue        BlueClass
##  Length:63241       Min.   : 48   Min.   : 48.0   Min.   : 44.0   No :61219
##  Class :character   1st Qu.: 80   1st Qu.: 78.0   1st Qu.: 63.0   Yes: 2022
##  Mode  :character   Median :163   Median :148.0   Median :123.0
##                     Mean   :163   Mean   :153.7   Mean   :125.1
##                     3rd Qu.:255   3rd Qu.:226.0   3rd Qu.:181.0
##                     Max.   :255   Max.   :255.0   Max.   :255.0
```

From this initial summary, it appears as if all three colors range from about 44 to 255, with Blue having the lowest median value, of 123, as well as the lowest mean, 125. This information may be valuable as the goal of the analysis is to identify where blue tarps are located in order to provide aid. It is also clear that there are far more images without blue tarps than there are images with blue tarps. Images with blue tarps make up only 3% of the total number of images.



A scatter plot of the colors shows an interesting non-linear correlation between Blue and Red. Where Green and Red and Green and Blue both seem to have similar patterns of correlation, Red and Blue seem to increase parabolicaly, which again, could have interesting connections to the goal of this analysis.

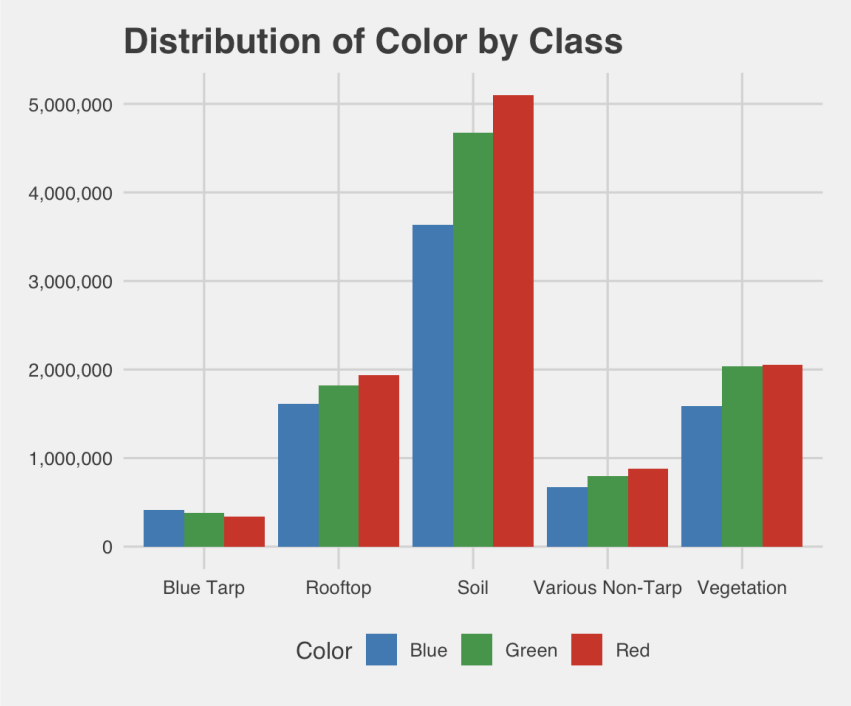
Before looking at only the image files for blue tarps, I thought it would be interesting to compare the quantities of red, blue, and green values in the various classes. To do that, I modified the dataset to group all data for each class, then sum the total values for each color and display it as a percentage of total image datapoints for that class.

Class <chr>	Red <dbl>	Green <dbl>	Blue <dbl>
Blue Tarp	30.24	33.22	36.54
Rooftop	36.05	33.93	30.03
Soil	38.01	34.89	27.10
Various Non-Tarp	37.37	34.13	28.50
Vegetation	36.17	35.93	27.90

5 rows

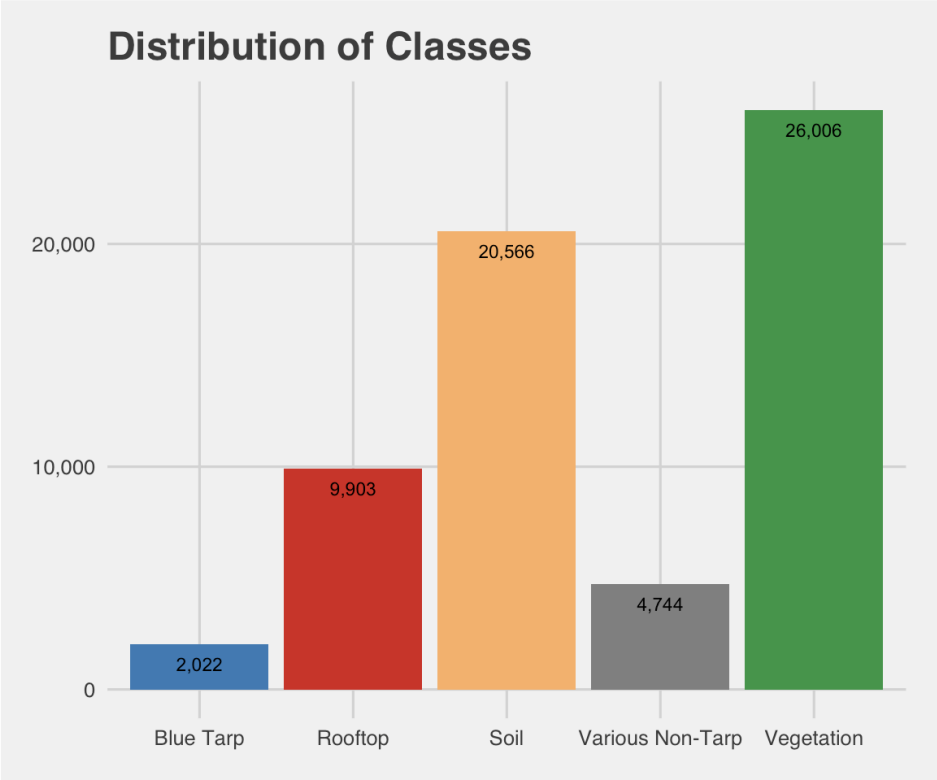
For the images classified as blue tarps, it is evident that there is a significantly higher percentage of blue pixels in each image, than for any of the other classes. Based on this, it should be possible to accurately differentiate which images are of blue tarps and which are not.

In order to visualize this, I create a bar chart of each class and their respective color values.



This bar chart shows a few key details. One, as was previously noted, there are far fewer pixels identified as being blue tarps than there are rooftops, soil, various non-tarp, and vegetation. Too, as noted in the chart above, the pixels identified as blue tarps are the only class where the blue data points are greater than red or green.

Out of curiosity, I note briefly the frequency of each class:



This bar graph quantifies what was noted earlier: the significantly lower frequency of blue tarps to any other class in the images.

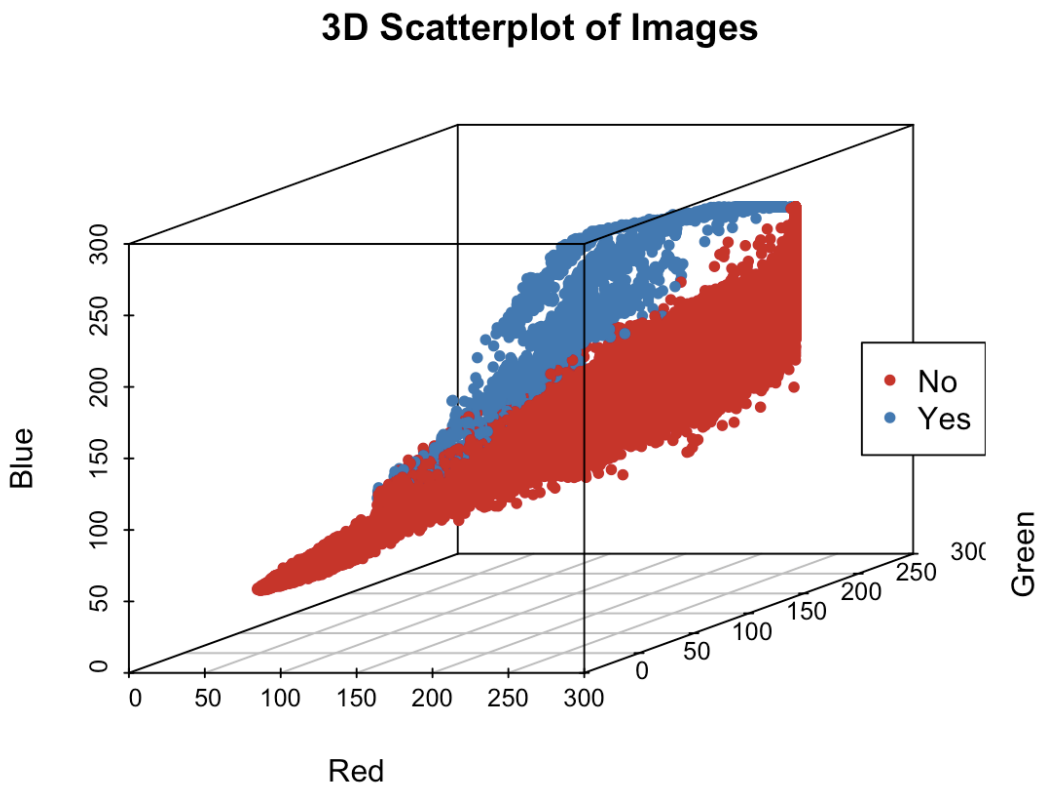
After observing some general trends in the data as a whole, it is time to turn our attention to just the class containing images identified as blue tarps, which I have put into a new variable: BlueClass.

First, I consider the correlation between the three color values.

This correlation matrix gives some interesting insight into the color striation in the images classified as having blue tarps. As images begin to have red pixel values over about 200, they are less likely to have blue tarps. There are almost no images with blue tarps that have green or blue pixel values below 100, and images containing a blue value of 200 are more seem to be exclusively classified as containing a blue tarp.

Finally, for visualization purposes, we fit the data to a 3d scatterplot in order to see how all three colors interact with each other.[1]

[1] Dr. Gedeck suggested the inclusion of a 3D model to confirm results I had gotten in my first logistic model



This plot is especially revealing. There is a very clear plane separating the images with and without blue tarps based on the amount of red, blue, and green values in each pixel. With this information, we can be confident that it is possible to accurately predict which images will contain blue tarps based on their RGB color values.

Fitting the Models

In order to effectively classify the presence of blue tarps, I used the training data to fit several models in R. In the interest of writing clean and easily reproducible code, I created a variable for set.seed() to be used when formulating each model, as well as creating a trainControl() object to be used on each model.[2] The models were fit using the caret package in R and each model was cross-validated using k=10 for the training data (and, eventually, the holdout data).

[2] The idea of setting the seed for reproducibility through multiple analyses: https://www.rpubs.com/christianaaronschroeder/788860

The goal of developing these algorithms is to be able to predict the location of the blue tarps where people may be stranded and waiting for aid. As such, the response variable is BlueClass, which is a categorical variable of images classified as having or not having a blue tarp, and the predictors are the three color identifiers: "Red", "Blue", and "Green".

For each model type, I will fit one model using just the three predictor variables, as well as a second model using interaction terms between the colors. Comparing these two models may help to identify whether interactions between the colors are useful in identifying the presence of blue tarps.

In evaluating each model, I considered various performance metrics: accuracy, false positive rate, false negative rate, and kappa. When studying imbalanced data such as this, where there are far more observations that are not blue tarps than observations which are blue tarps, Cohen's Kappa can provide a more realistic view of the model performance than just accuracy. Unfortunately, this metric does not give much data regarding the prediction performance of the models, only compares the observed accuracy of the model with the accuracy that might be observed by chance. Due to the nature of the data, it is possible that a model could always predict either "yes" or "no" and still have an accuracy of over 95%. [3] With the goal of the project in mind, of being able to rescue as many displaced persons as possible (and with no specific limitations on resources or personnel noted), I placed special emphasis on minimizing the false negative rate, while balancing for maximum accuracy, when tuning the thresholds of each model.

Logistic Regression

Fitting the Model

The first model I fit to the training data is a logistic regression model. I fit both models and then compare the summary results, specifically the Accuracy and Kappa metrics, for each model:

```
##                Accuracy  Kappa
## [1,] "Model 1" 0.9952404 0.9198727
## [2,] "Model 2" 0.9960469 0.9352958
```

[3] Research on Cohen's Kappa which greatly aided my understanding of this metric: https://thenewstack.io/cohens-kappa-what-it-is-when-to-use-it-and-how-to-avoid-its-pitfalls/

Having used the caret package for training a logistic model, we can see that the results indicate an accuracy of 99.53% and 99.6% respectively, which for most data sets would seem improbable (and is close enough to 1 to trigger a warning message from R). However, considering the visualizations earlier which indicated a clear differentiation between image data with and without blue tarps, it seems probable that a logistic regression would indeed be able to accurately predict the presence of a blue tarp based on image data.

The accuracy of the model with interaction effects is slightly higher than the model with individual predictors, but both models show extremely strong performance based on accuracy. We also note that the Kappa of the first model is .92, and the Kappa of the second model (with interaction terms) is .935, both of which are high, and indicate a strong performance of the models.

A confusion matrix validates these observations:

Model 1 (without interaction terms):

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.7  0.4
##        Yes  0.1  2.8
##
##  Accuracy (average) : 0.9952
```

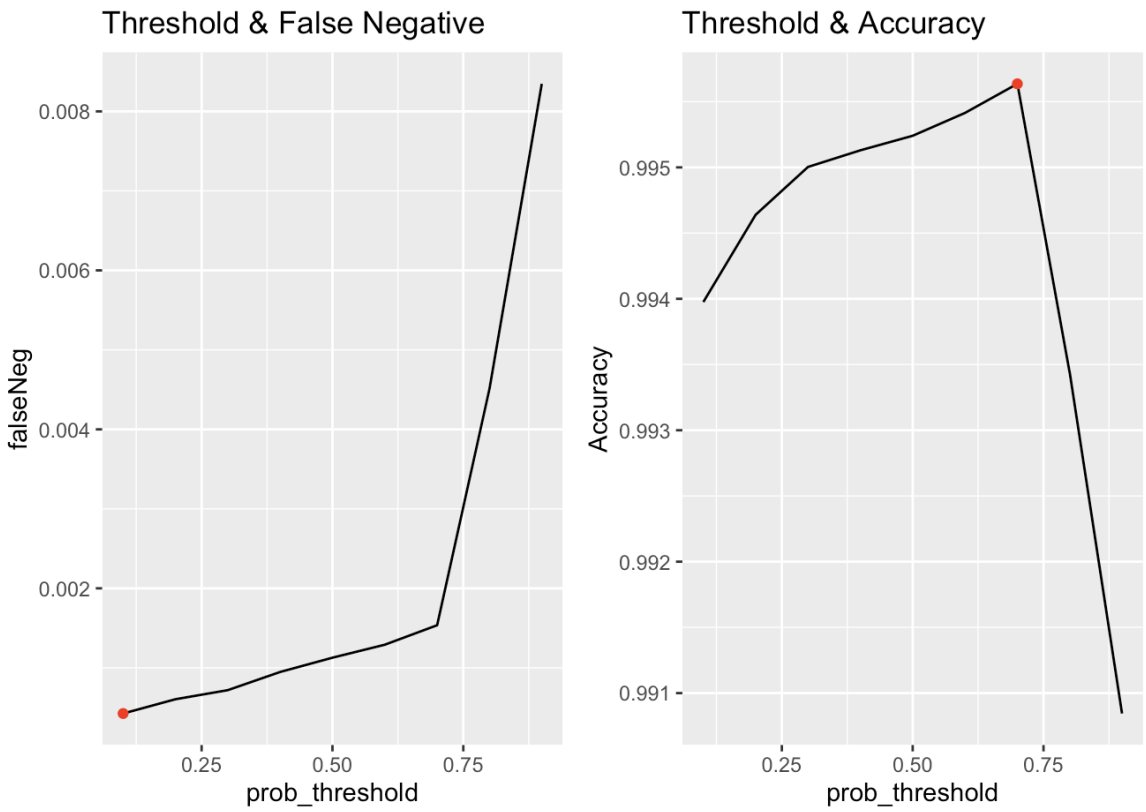
Model 2 (with interaction terms):

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.6  0.2
##        Yes  0.2  3.0
##
##  Accuracy (average) : 0.996
```

This leads us to the consideration of threshold values in order to fine-tune the models for optimal performance.

Threshold Evaluation

The next step in tuning the models is to choose the threshold which produces the highest accuracy for each model. To do that, we can use the thresholder() function in the caret package, which I put within a function in order to quickly reproduce for each model to be tested.[4]

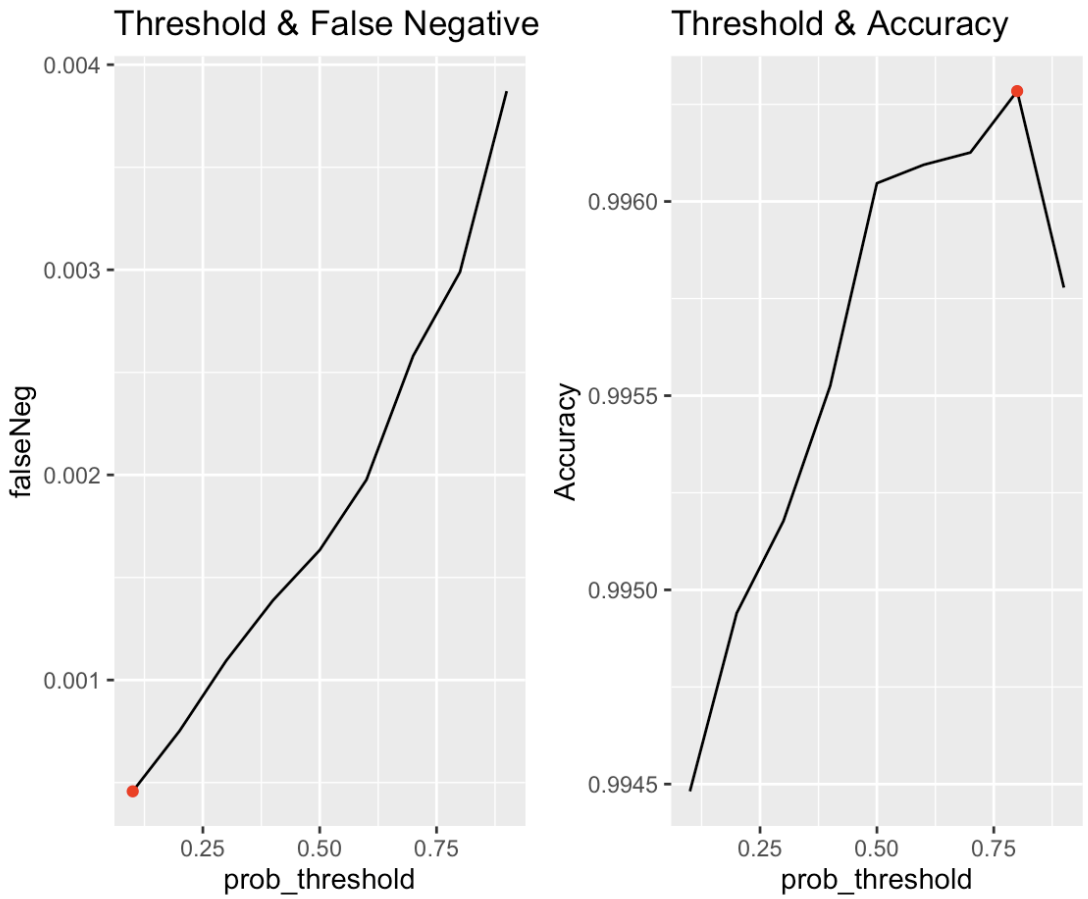


For the first model, the threshold with the highest accuracy is .7. This threshold also gives a sensitivity rate of .998 and specificity of .91. In the case of identifying the images containing blue tarps, it seems most prudent to minimize the false negative rate in order to ensure that as many people as possible receive aid. (However, if there were restraints in budget and/or time, this is something that may need to be reconsidered). Thus, we note that the false negative rate is 0.0015 at a threshold of .7. The plots indicate that a lower threshold would

[4] Inspiration to build a function for threshold testing rather than reproducing the code for each model: https://www.rpubs.com/christianaaronschroeder/788860. Detail and research on thresholder function: https://rdrr.io/cran/caret/man/thresholder.html

reduce the false negative rate, but that such reduction parallels a reduction in accuracy at lower thresholds.

It is also worth noting that the range of accuracy throughout the possible threshold values is .990 to .993. At any threshold, the model still has an extremely high degree of accuracy!

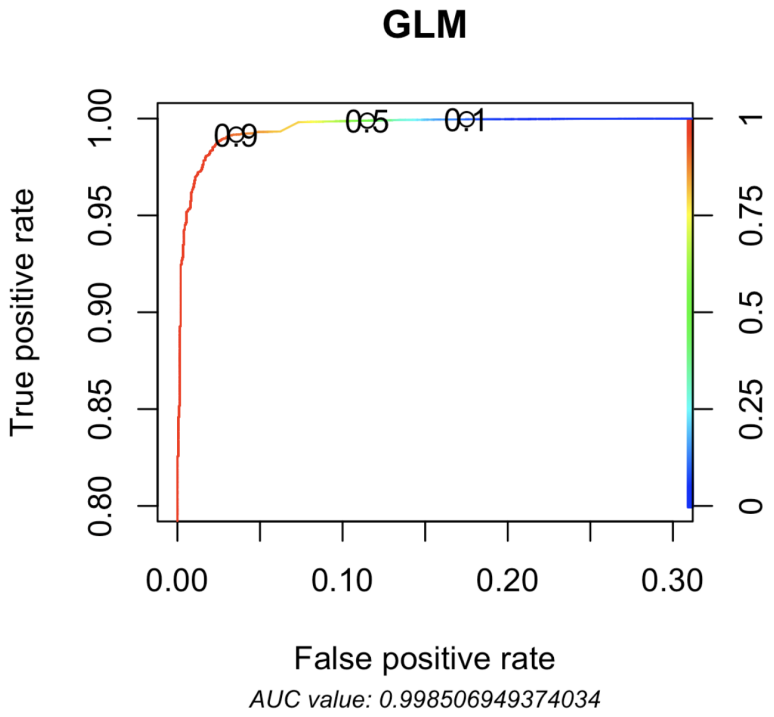


For the second model (with interaction terms), the threshold with the highest accuracy is .8. However, much like with Model 1, the range of accuracy values across thresholds is extremely small, indicating that the model performs well at any threshold. Still, we want to maximize accuracy, even if only with minimal gains, so choose .8. This threshold also gives a sensitivity rate of .997 and specificity of .97, which is slightly more balanced than the prior model, but as such, raises the false negative rate slightly.

Because the false negative rate is lower in the first model (without interaction terms) and the other metrics for model performance change only minimally, the best logistic regression model seems to be the first model, with a threshold of .7.

ROC Plot

We confirm this with a ROC plot:[5]



The strong performance of the logistic model is evident in the ROC plot, as the curve nearly perfectly traces the far upper left reaches of the plot, showing that there is very little difference between .1 and .9 thresholds; all models have a relatively high accuracy and maximize both sensitivity and specificity. We can observe, however, the slight bump in performance right around the .7 mark. The AUC score, printed at the base of the plot, also indicates, in a classification-threshold-invariant way, the high quality of the model's predictions. Because AUC ranges from 0 to 1, an AUC value higher than .99 validates the high level of prediction accuracy that the ROC curve shows.

Model Selection

Because the logistic model without interaction terms performs so strongly, we choose to keep the first logistic model with a threshold of .7 to move forward with our analysis. This model is summarized below:

---

[5] The idea for creating a function for creating ROC plots for reproducibility: https://www.rpubs.com/christianaaronschroeder/788860

Parameter	Probability Threshold	Sensitivity	Specificity	Accuracy
none	0.7	0.9984645	0.9099985	0.9956357
Kappa	**Precision**	**Detection Rate**	**False Neg**	**False Pos**
0.927959	0.9970318	0.9665407	0.001535483	0.09000146

LDA (Linear Discriminant Analysis)

Fitting the Model

Now we will fit an LDA model to the data, and once again create two models: one with interaction terms and one without.

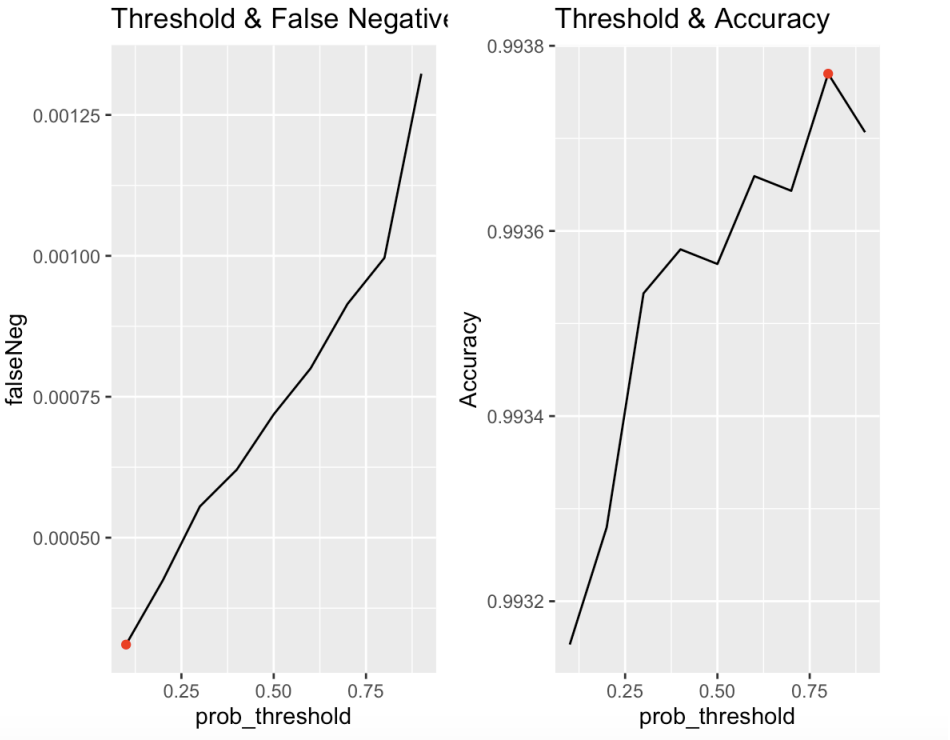
```
##                 Accuracy  Kappa
## [1,] "Model 1" 0.9839187 0.7530147
## [2,] "Model 2" 0.9935643 0.8872589
```

Similarly to the logistic regression, LDA results in a high level of accuracy in both models. In this case, the two models are even closer, varying only by 0.01 in accuracy, though the Kappa of the model with interaction terms does seem to be stronger. Again, we note that this tells us very little about the prediction of the model, considering the strong imbalance of the data. Thus, we proceed with both models and consider improvements that might be made by threshold tuning.

Threshold Evaluation

Like before, we will analyze the possible threshold values for each model to help us determine the best possible fit:

For the first model (without interaction terms), 0.1 is clearly the point of lowest false negative rate and highest accuracy.



The second model (with interaction terms) differs dramatically in threshold value depending on which performance metric is of most interest. The lowest false negative rate is at .1, while the highest accuracy is at .8, with considerable decrease in each when tuning for one or the other.

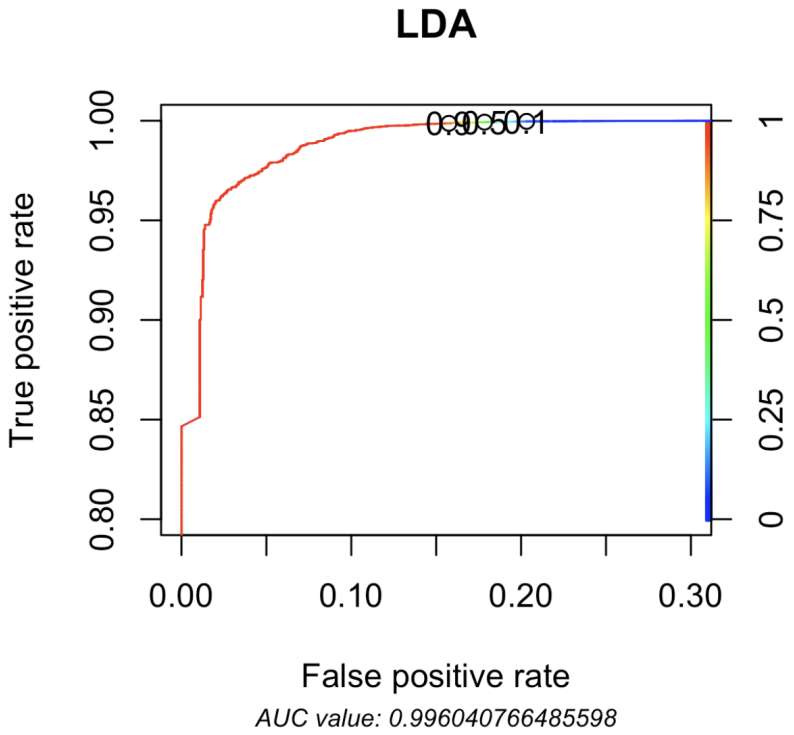
Like with the logistic regression models, both LDA models only range in accuracy above .9, which means that our threshold tuning, while improving model performance, does so only minimally.

Because the second model has almost no false negative rate and an overall higher accuracy compared with the first model, we continue with the second model (with interaction terms) with a threshold of .75 (which maximizes accuracy, while minimizing false negative rate).

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  96.7  0.6
##        Yes  0.1  2.6
##
##  Accuracy (average) : 0.9936
```

In the confusion matrix above, we see that there is a slightly higher false positive rate from the LDA model as a whole (without the threshold tuning) than there was in the logistic model. The false negative rate is also very slightly higher than the LDA model.

ROC Plot



The ROC plot for the LDA model confirms what we identified using thresholder() above: that any threshold between 0.1 and 0.9 would have similar results. The graph does appear to indicate that the logistic model has a better fit overall. The AUC value at the bottom of the plot confirms this, with .996 being slightly lower than the .999 AUC of the logistic regression model.

Model Selection

The final LDA model, which is fit with interaction terms and with a threshold of .7 is summarized below:

Parameter	Probability Threshold	Sensitivity	Specificity	Accuracy
none	0.7	0.9990853	0.8288689	0.9936434
Kappa	**Precision**	**Detection Rate**	**False Neg**	**False Pos**
0.8894463	0.9943754	0.9671416	0.0009147471	0.1711311

QDA (Quadratic Discriminant Analysis)

Fitting the Model

The next model we will fit to the data is a quadratic model, using QDA. Once again, we will fit two models.

```
##                 Accuracy  Kappa
## [1,] "Model 1" 0.9945605 0.9049272
## [2,] "Model 2" 0.9888205 0.8443475
```

The first model shows a higher accuracy and Kappa value, indicating that the interaction terms are not as relevant to a QDA model. The results of the first model show an accuracy of 99.46%, which is marginally higher than the previous models.

In this instance, we will move forward with just the first model, as both accuracy and kappa are higher without interaction terms, thus it seems prudent to keep the simpler model.

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.8  0.5
##        Yes  0.0  2.7
##
##  Accuracy (average) : 0.9946
```

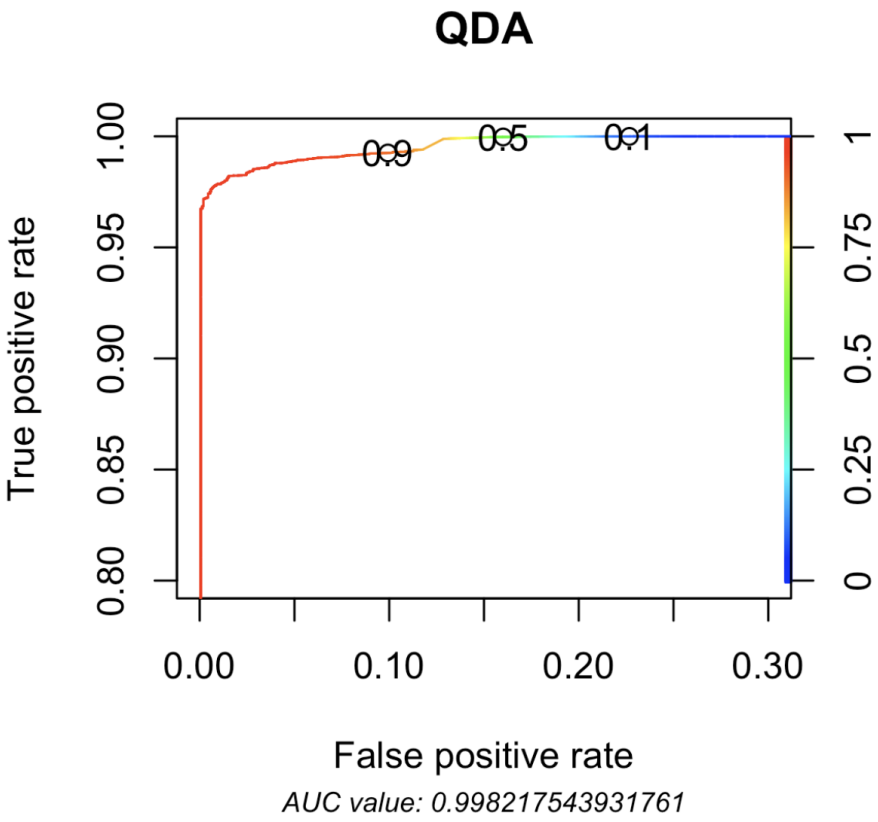
From the confusion matrix, it is evident that the model performs well, even without fine-tuning. Still, we will move forward with threshold evaluation to attempt to maximize model performance by minimizing the false negative rate.

Threshold Evaluation

In the case of the QDA model, the threshold with lowest false negative rate is at .1, while the threshold with highest accuracy is .6. If we examine the plots of threshold versus accuracy and false negative rate, it is evident that a threshold of .5 would be an effective compromise between minimizing false negatives and maximizing accuracy.

Once again, it is worth noting that the range of values in both accuracy and false negative rates is extremely small, indicating a model with very strong performance even before any kind of tuning.

ROC Plot



The appearance of the ROC plot is more similar to the logistic regression model than the LDA model, with the line closely following the upper left reaches of the plot, indicating a high level of accuracy regardless of threshold. The AUC value of .998 confirms.

One thing to note in the comparison of the QDA and LDA model ROC plots is how the QDA model more closely follows the upper left reaches of the plot than did the LDA model. One explanation for this is the greater flexibility of a QDA decision boundary, which, unlike the linear decision boundary of the LDA model, can bend into quadratic shapes. This increased flexibility is likely part of the reason for the improved performance in the QDA model over the LDA model.

Model Selection

Based on the ROC plot above and the results of the threshold evaluation, the QDA model with the best balance of accuracy and low false negative rate is model 1, with a threshold of .5.

Parameter	Probability Threshold	Sensitivity	Specificity	Accuracy
none	0.5	0.9996896	0.8392845	0.9945605
Kappa	**Precision**	**Detection Rate**	**False Neg**	**False Pos**
0.9049272	0.9947185	0.9677266	0.0003103614	0.1607155

kNN (k-Nearest Neighbor)

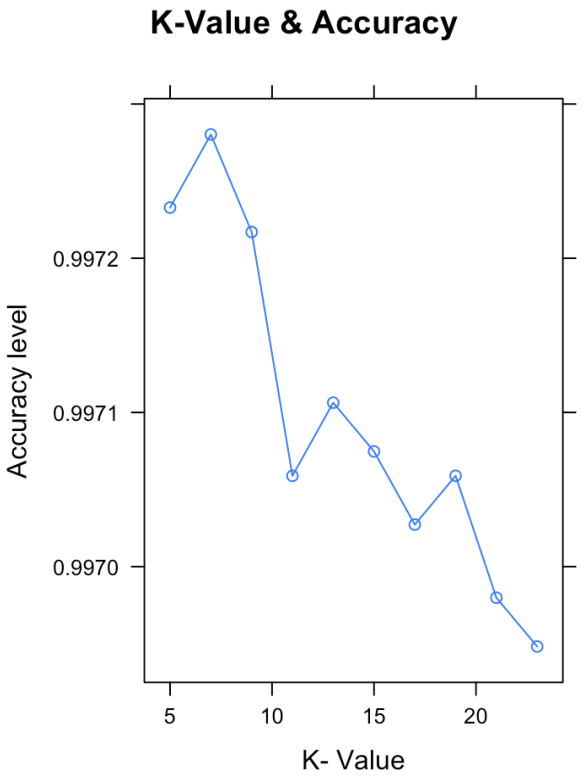
Fitting the Model

Next, we use k-Nearest Neighbor to fit a model. For the kNN model, I am choosing to fit just one model, without any interaction terms, as the greater variance of the kNN model would become overly complex with the addition of interaction terms. Additionally, the prior models have indicated that adding interaction terms does not necessarily benefit the model's accuracy.

```
## k-Nearest Neighbors
##
## 63241 samples
##     3 predictor
##     2 classes: 'No', 'Yes'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 56918, 56917, 56917, 56917, 56916, 56917, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9972328  0.9553436
##    7  0.9972803  0.9562512
##    9  0.9972170  0.9551516
##   11  0.9970589  0.9526162
##   13  0.9971063  0.9533309
##   15  0.9970747  0.9528944
##   17  0.9970273  0.9520676
##   19  0.9970589  0.9526091
##   21  0.9969798  0.9513209
##   23  0.9969482  0.9508038
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

The kNN cross-validation results selected k=7 as the optimal value, giving an accuracy of 99.73%, which is the highest of the models tested thus far. (Interestingly enough, the range of accuracy values seem to only be from 0.9970 to 0.9973, which is very tight). When considered in the context of providing aid to those in need, this seems to be a solid option for choosing as our best model to move forwards.

The plot of k-Value & Accuracy shows clearly that k=7 is a significant improvement in accuracy in comparison to other values of k. As stated before "significant" may be an overstatement when the range of accuracy values is less than 0.001.



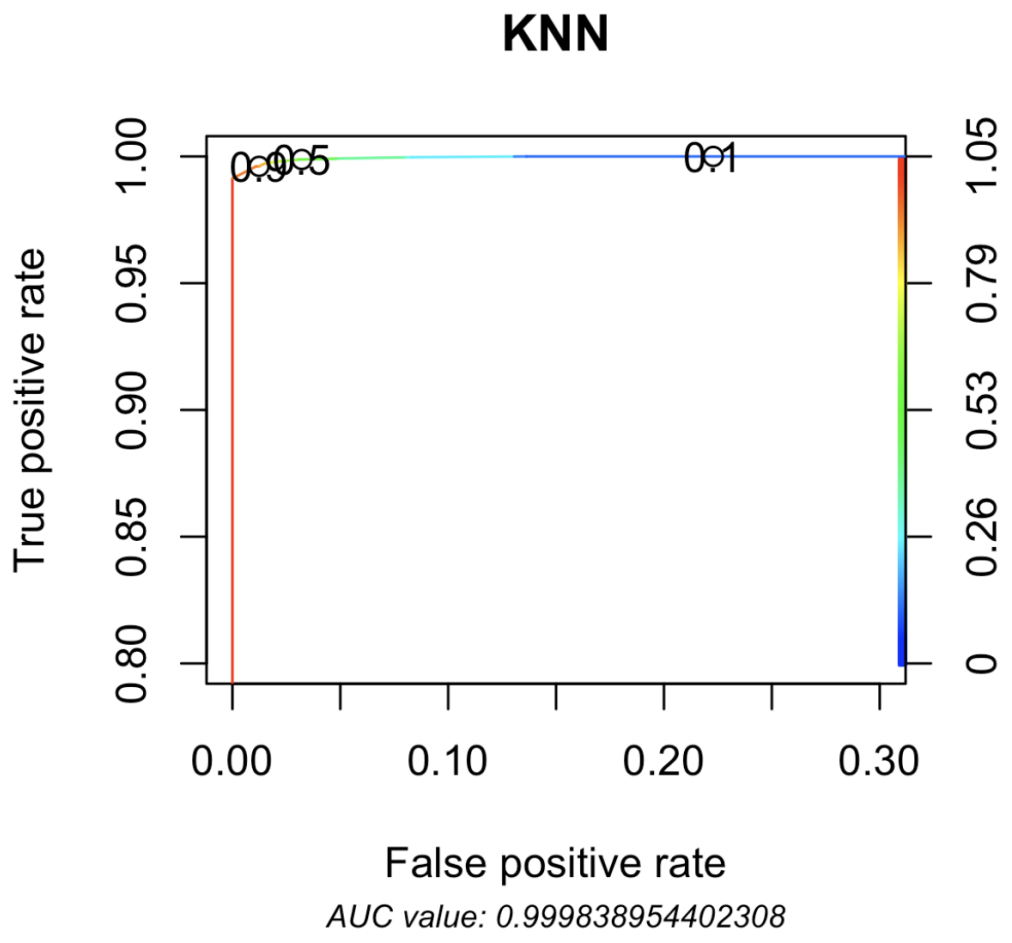
Threshold Evaluation

The highest accuracy is at a threshold of .5, which still results in a strong false negative rate for the model. However, in evaluating the plots above, it seems as if choosing a threshold of .4 would be more effective. It would very slightly reduce overall accuracy, but would more significant minimize the false negative rate.

A confusion matrix which gives an average of the cell counts across resamples confirms that the accuracy of the model, even without any specific tuning, is extremely high.

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.6  0.1
##        Yes  0.2  3.1
##
##  Accuracy (average) : 0.9973
```

ROC Plot



Similarly to the logistic and QDA models, the ROC curve confirms a strong performance by the kNN model. Notably, the ROC curve comes together at almost a 90-degree angle, which evidences the lack of parabolity in the model. The AUC value of 0.9998 is the highest of any model evaluated thus far.

Model Selection

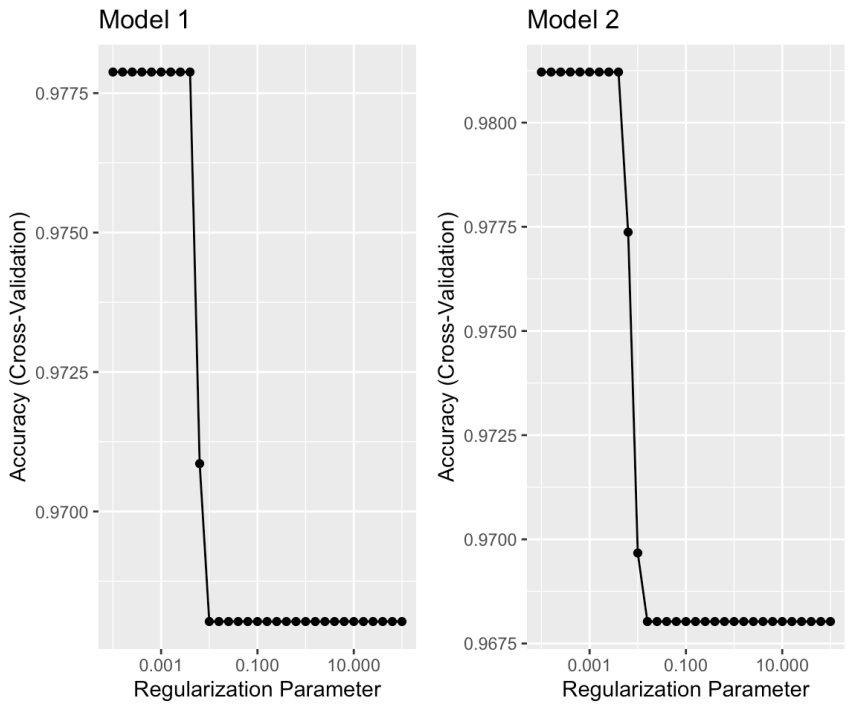
Based on the threshold evaluation and ROC curve, the kNN model with k=7 and a threshold of 0.4 is the best kNN model for this data.

k	Probability Threshold	Sensitivity	Specificity	Accuracy
7	0.4	0.9988729	0.9426328	0.9970747
Kappa	**Precision**	**Detection Rate**	**False Neg**	**False Pos**
0.9521129	0.9981071	0.966936	0.001127099	0.05736721

Penalized Logistic Regression (elastic net penalty)

The next model that we are going to fit to the data is a penalized logistic regression (elastic net penalty) model. To do this, we will use the ridge regression technique. Like with prior models, we will fit two models, one with interaction terms and one without.

Fitting the Model



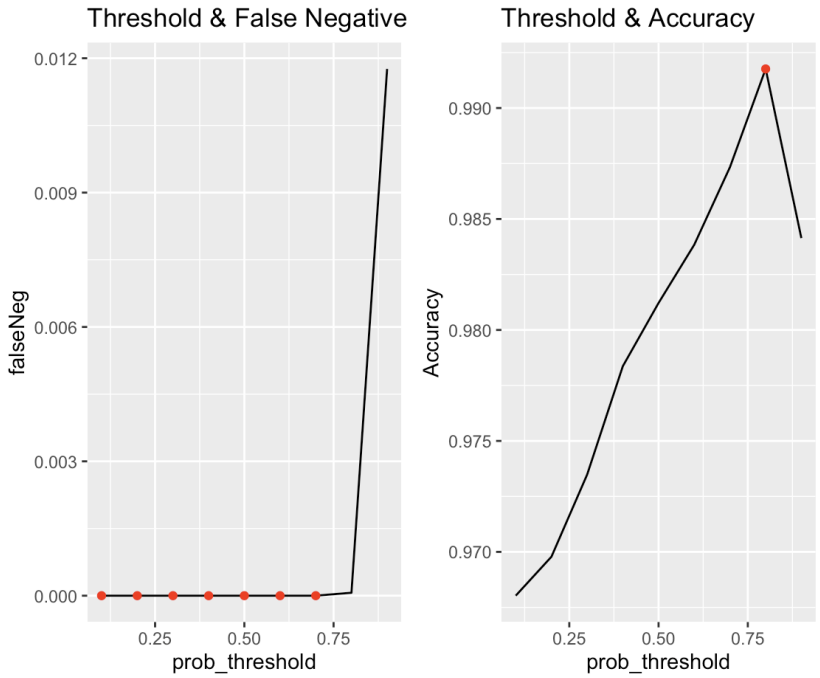
We start by fitting the models using a cross-validation technique for lambda. For both models, the lambda values with the highest accuracy range from 0.0001 to 0.0039, with a significant decrease in accuracy as lambda increases. It is also worth noting that both the accuracy and the kappa values of the second model (with interaction terms) are higher. With that in mind, we will continue with the second model.

Threshold Evaluation

The next step is to evaluate the various thresholds for the model. We start with a confusion matrix:

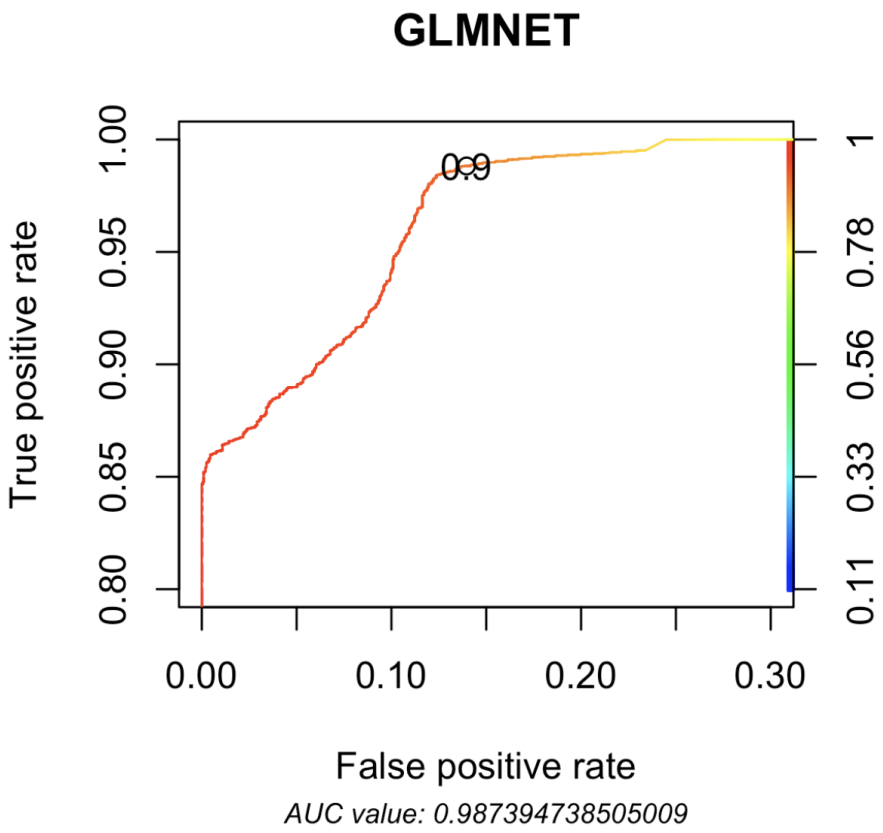
```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.8  1.9
##        Yes  0.0  1.3
##
##  Accuracy (average) : 0.9812
```

We can see from the confusion matrix that the overall accuracy of the model is over 98%, which is quite strong. This is confirmed when we examine the threshold plots:



Interestingly, the threshold results for the lowest false negative rate are level from 0 all the way though 0.7. The highest accuracy very distinctly comes at a threshold of 0.8, though the range of accuracy is above 0.97 regardless of threshold value. When we dig a little bit deeper, we can see that the sensitivity at a threshold of .8 is still 0.9999, or a 0.000065 false negative rate. Thus, we can feel confident moving forward with a threshold of 0.8.

ROC Plot



The ROC plot dips considerably after 0.09, but shows a consistently high level of accuracy at lower thresholds. The AUC of .98 is still high, but somewhat lower than the kNN and logistic models. When we remember that the accuracy of a model classifying all observations as just one class would be at least 95%, the lower accuracy of this model becomes more significant.

Model Selection

Based on the threshold evaluation and ROC curve, the PLR ridge regression model with interaction terms, a lambda of 0.000398, and a threshold of 0.8 is the best ridge regression fit for the data.

Alpha	Lambda	Probability Threshold	Sensitivity	Specificity	Accuracy
0	0.003981072	0.8	0.9999347	0.7443423	0.9917617

Kappa	Precision	Detection Rate	False Neg	False Pos
0.8480063	0.991626	0.9679638	0.0000653812	0.2556577

Support Vector Machine

The final model that we are going to fit to the data is a support vector machine model. Based on our exploratory data analysis, we choose to fit just a linear SVM model, as the data does not appear to follow any kind of polynomial or radial patterns. We use caret to tune the cost, and select the model with the highest accuracy.

Fitting the Model

```
## Support Vector Machines with Linear Kernel
##
## 63241 samples
##     3 predictor
##     2 classes: 'No', 'Yes'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 56918, 56917, 56917, 56917, 56916, 56917, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00        NaN        NaN
##   0.01  0.9910343  0.8549687
##   0.05  0.9951772  0.9175566
##   0.10  0.9952563  0.9194315
##   0.25  0.9953195  0.9210654
##   0.50  0.9953986  0.9224700
##   0.75  0.9953669  0.9220690
##   1.00  0.9954144  0.9228066
##   1.25  0.9953986  0.9225601
##   1.50  0.9954302  0.9231453
##   1.75  0.9953827  0.9223157
##   2.00  0.9953986  0.9225159
##   5.00  0.9953986  0.9225242
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 1.5.
```

The cross-validated SVM model with a linear kernel results in an accuracy of 99.55 and kappa of 92.4 with a tuned C value of 1.25.

Threshold Evaluation

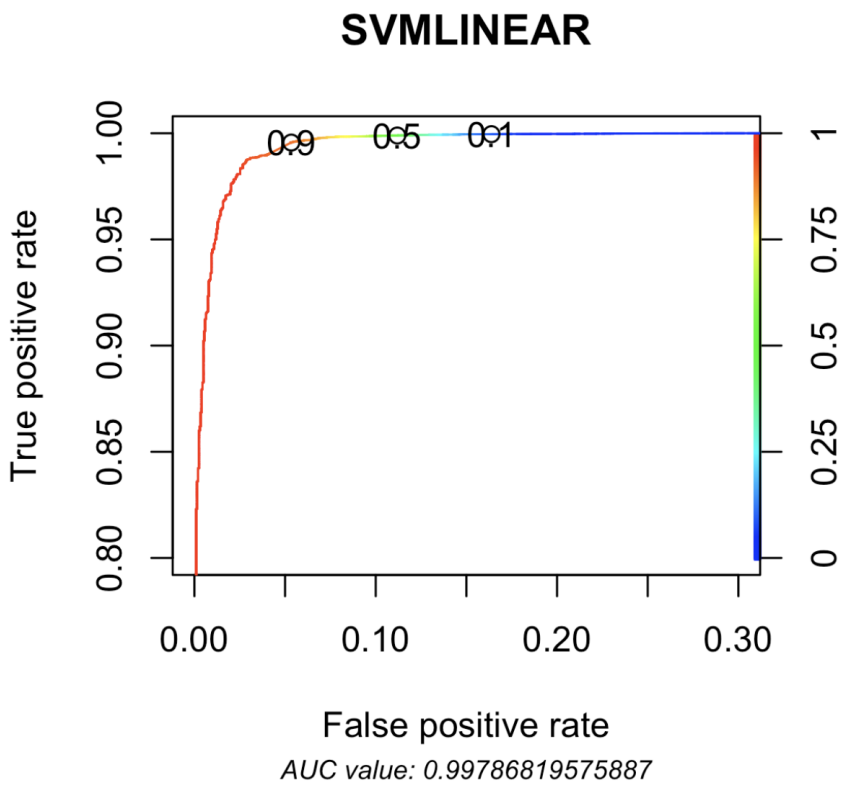
The next step is to evaluate the various thresholds for the model. We start with a confusion matrix:

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
##        No  96.7  0.4
##        Yes  0.1  2.8
##
##  Accuracy (average) : 0.9954
```

We can see from the confusion matrix that the overall accuracy of the model is over 99%, which is quite strong. This is confirmed by the threshold evaluation.

Though the graphics may at first glance seem to show a wide gap between minimizing the false negative rate and maximizing accuracy, a glance at the scale shows otherwise. The lowest accuracy reported is 99.45, and the highest false negative rate is 0.0045. In that case, a balance between the two and a threshold of 0.5 seems to be a reasonable compromise.

ROC Plot



The ROC plot shows a consistently high level of accuracy throughout. The AUC of .998 is extremely high.

Model Selection

Based on the threshold evaluation and ROC curve, the SVM model with a linear kernel, cost of 1, and threshold of 0.5 is a strong SVM model fit for the data.

C	Probability Threshold	Sensitivity	Specificity	Accuracy
1.25	0.5	0.9989546	0.8896942	0.9954618
Kappa	**Precision**	**Detection Rate**	**False Neg**	**False Pos**
0.9237073	0.9963673	0.9670151	0.001045423	0.1103058

Performance Table

The next step is to evaluate model performance across each of the various models. From each group of models, I chose the best-performing model and collected those results into two data frames, one showing model metrics and the second showing the selected model parameters:

Model	Interaction Terms	k	C	alpha	lambda	prob_threshold
LOG	No	NaN	NaN	NaN	NaN	0.7
LDA	Yes	NaN	NaN	NaN	NaN	0.7
QDA	No	NaN	NaN	NaN	NaN	0.5
PLR-Ridge	Yes	NaN	NaN	0	0.004	0.8
kNN	No	7	NA	NA	NA	0.4
SVM	No	NA	1.25	NA	NA	0.5

Model	Sensitivity	Specificity	Accuracy	Kappa	Precision	Detection Rate	falseNeg	falsePos
LOG	0.9985	0.9100	0.9956	0.9280	0.9970	0.9665	0.0015	0.0900
LDA	0.9991	0.8289	0.9936	0.8894	0.9944	0.9671	0.0009	0.1711
QDA	0.9997	0.8393	0.9946	0.9049	0.9947	0.9677	0.0003	0.1607
PLR-Ridge	0.9999	0.7443	0.9918	0.8480	0.9916	0.9680	0.0001	0.2557
kNN	0.9989	0.9426	0.9971	0.9521	0.9981	0.9669	0.0011	0.0574
SVM	0.9990	0.8897	0.9955	0.9237	0.9964	0.9670	0.0010	0.1103

Determining the preferred model requires coming back to our goal of finding blue tarps in the image files in order to provide aid to those in need. As such, if there are no time, personnel, or budget constraints that would require limiting false positive results, it seems as if the preferred model would be one which aims to minimize false negatives while keeping overall accuracy high.

From the tables above, we can see that Ridge Regression is the model with the lowest false negative, though not the highest accuracy (0.9918). QDA would be the next preferred model, followed by the kNN model, which though it does not have the next-lowest false negative rate, has the highest overall accuracy of any of the models.

Conclusions Based on Training Data

Based on the above analysis, it is worthwhile to note that all five models are strong candidates for being able to correctly predict the presence of blue tarps. That being said, with a focus on reducing false negatives in order to ensure that as many people are aided as possible, we recommend the following models (in this order) for use in classifying image files as those containing Blue Tarps and those without Blue Tarps with the purpose of providing aid to those harmed in the disaster in Haiti.

1) PLR: Ridge Regression

The Ridge Regression model is the first choice of predictive model for this data. To tune this model, we chose a threshold of 0.8, alpha of 0, and lambda of 0.000398. This model was fit using interaction terms. Both accuracy and false negative rate were minimized, with a false negative rate of just 0.0001, and an accuracy over 99%.

2) QDA Regression

The second choice of model is the QDA regression model without interaction terms with a tuning threshold of 0.5. This model minimizes false negatives (0.0003), and has a very slightly improved accuracy rate from the Ridge Regression model, over 99%.

3) K-Nearest Neighbor

The third choice of model is the K-Nearest Neighbor (kNN) model. This model was fit without interaction terms and a threshold of 0.4, and though the false negative rate increased somewhat (to 0.0010), the accuracy is the highest of any of the models (99.71%). This model edges out the Logistic Regression Model as my third choice because the overall

accuracy of the kNN model is higher, and false positives are significantly reduced by the kNN model.

I am confident that any of these three models would be effective in identifying blue tarps from image data.

Holdout Data

Excluded from the initial data is a holdout data set comprised of seven different files. There are additionally three image files, which were labeled with similar notation as the text files. One of these appeared to be paired with a set of the .txt files, but without advanced image processing capabilities, did not seem to offer any clear insight into the data. Also worth noting is that most of the .txt files appeared to be paired, with file naming identifying both a tarp and non-tarp file for all but one of the files, which was labeled as non-tarp, but did not have a paired tarp file. I chose to include this data anyway, with the consideration that the training data also included far more non-tarp observations than tarp, and that presumably, the same would be true in the holdout data. After making these preliminary decisions, I loaded the data from the .txt files into dataframes, and then combined them.

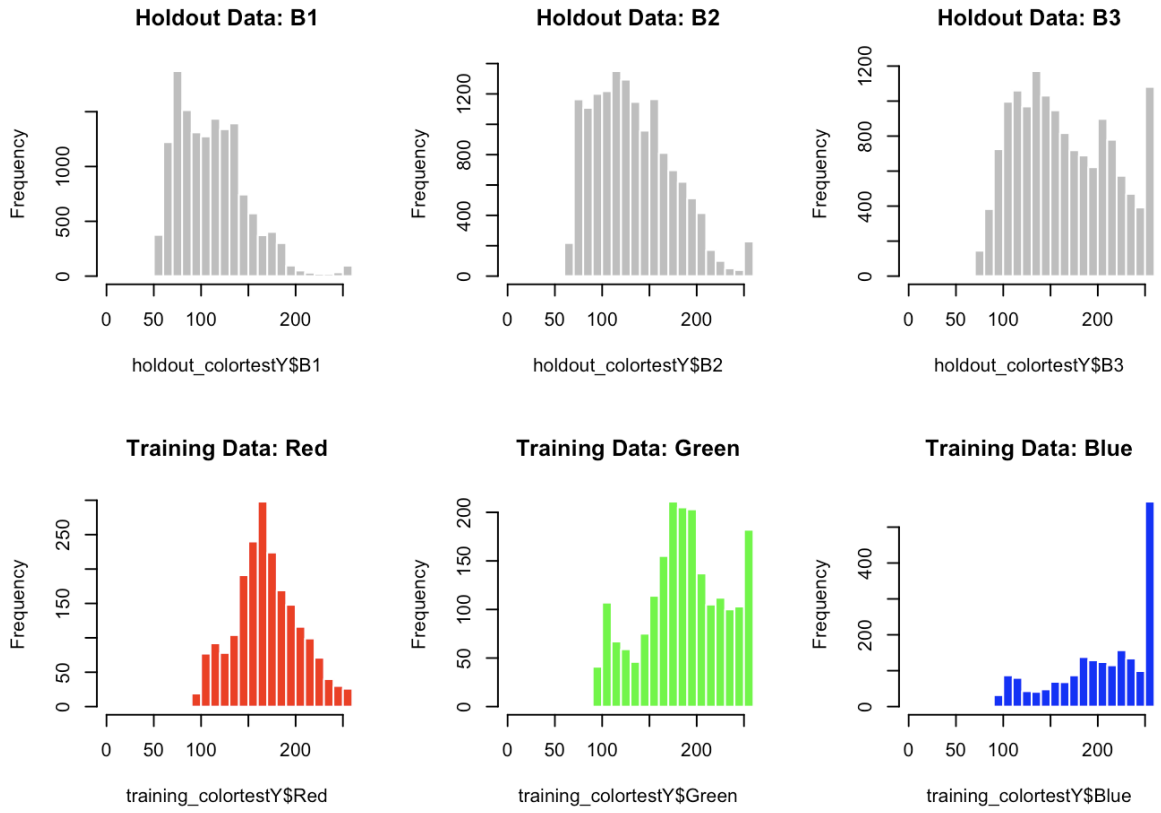
Before moving on, I made a few basic observations about the holdout data.

```
##        ID                X               Y             Map X
##  Min.   :     1   Min.   : 408   Min.   : 147   Min.   :769801
##  1st Qu.:121642   1st Qu.:1732   1st Qu.:1620   1st Qu.:769880
##  Median :246903   Median :2235   Median :1883   Median :772389
##  Mean   :326168   Mean   :2270   Mean   :2866   Mean   :771866
##  3rd Qu.:478234   3rd Qu.:2749   3rd Qu.:4027   3rd Qu.:773028
##  Max.   :979278   Max.   :4139   Max.   :6487   Max.   :775373
##      Map Y              Lat             Lon                B1
##  Min.   :2049517   Min.   :18.52   Min.   :-72.44   Min.   : 27.0
##  1st Qu.:2049648   1st Qu.:18.52   1st Qu.:-72.44   1st Qu.: 76.0
##  Median :2049866   Median :18.52   Median :-72.42   Median :107.0
##  Mean   :2049782   Mean   :18.52   Mean   :-72.42   Mean   :118.3
##  3rd Qu.:2049887   3rd Qu.:18.52   3rd Qu.:-72.41   3rd Qu.:139.0
##  Max.   :2050020   Max.   :18.52   Max.   :-72.39   Max.   :255.0
##        B2               B3          BlueClass
##  Min.   : 28.0   Min.   : 25.00   No :1989697
##  1st Qu.: 71.0   1st Qu.: 55.00   Yes:  14480
##  Median : 91.0   Median : 66.00
##  Mean   :105.4   Mean   : 82.36
##  3rd Qu.:117.0   3rd Qu.: 88.00
##  Max.   :255.0   Max.   :255.00
```

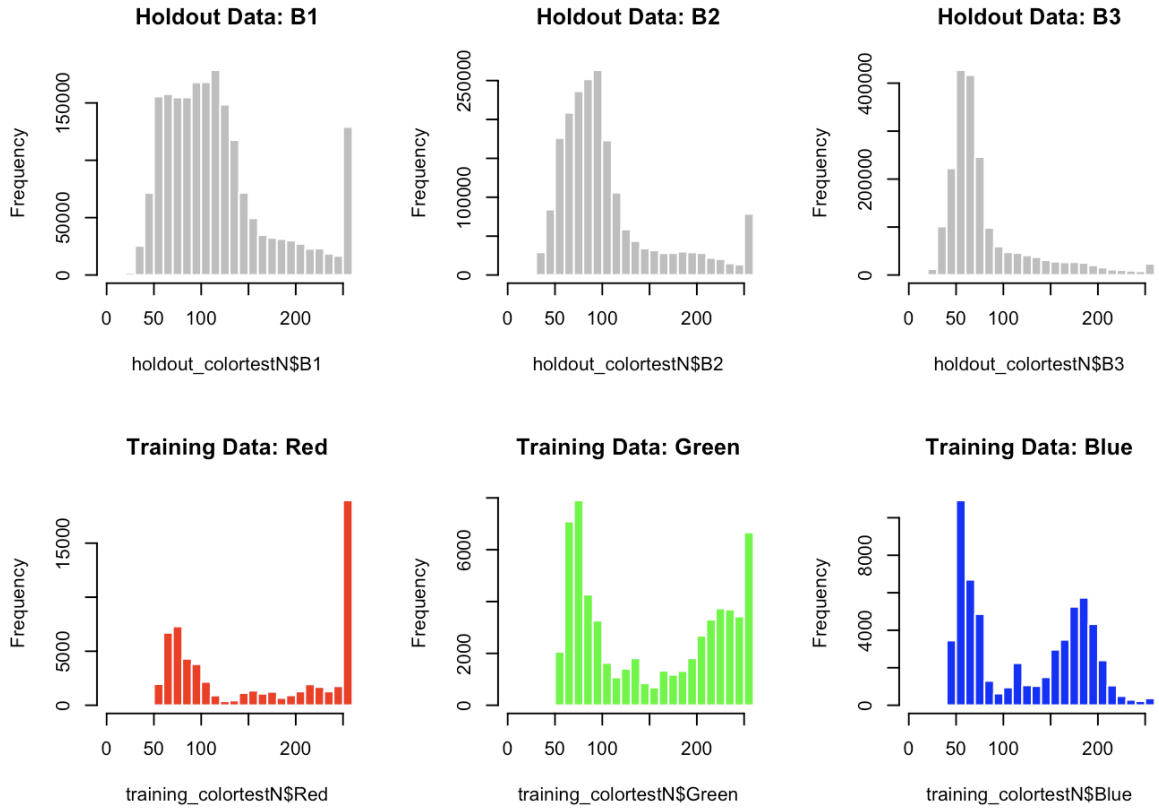
What stands out most from this information is the imbalance of the data. There are 1,989,697 non-tarp observations, and only 14,480 observations that are classified as being blue tarps. This is even more imbalanced than the training data, where approximately 3% of the observations were blue tarps. Here, only 0.7% of the observations are blue tarps. As a result, even a classification model that predicted all of the pixels to be of the same class could be expected to have an accuracy of over 99%.

Notably, the holdout data is not formatted in the same way as the original data and contains columns labeled B1, B2, and B3 rather than the RGB data columns seen in the training data. The working hypothesis is that these three columns in the holdout data are indeed designating color values within the pixel data, so to test this theory, I compare the distribution between the training and holdout datasets.

First, we observe the distribution in training and holdout data sets for the observations labeled as Blue Tarps.

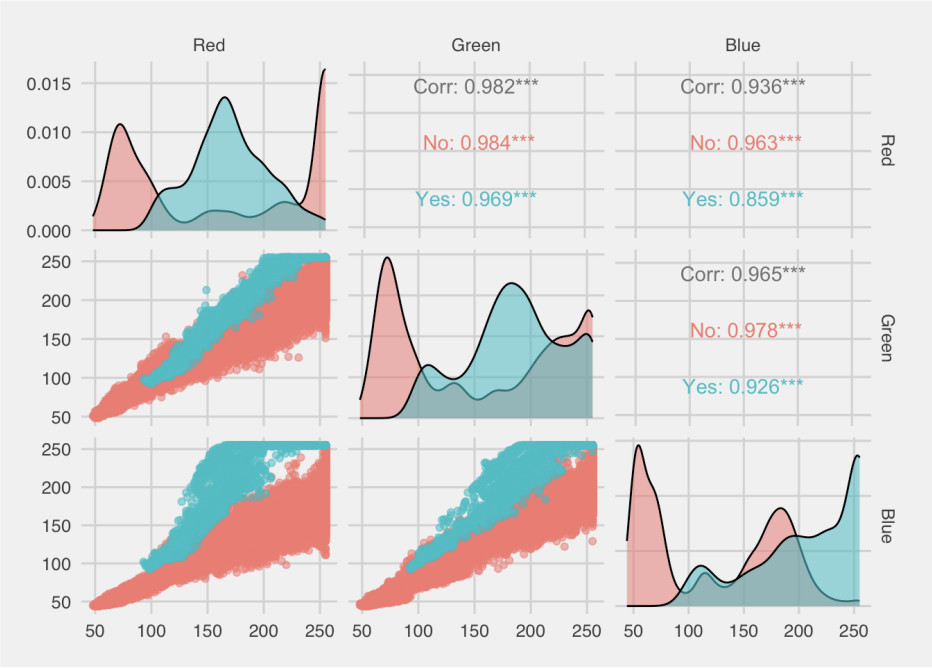


Then, we compare the observations that are labeled as non-tarp:

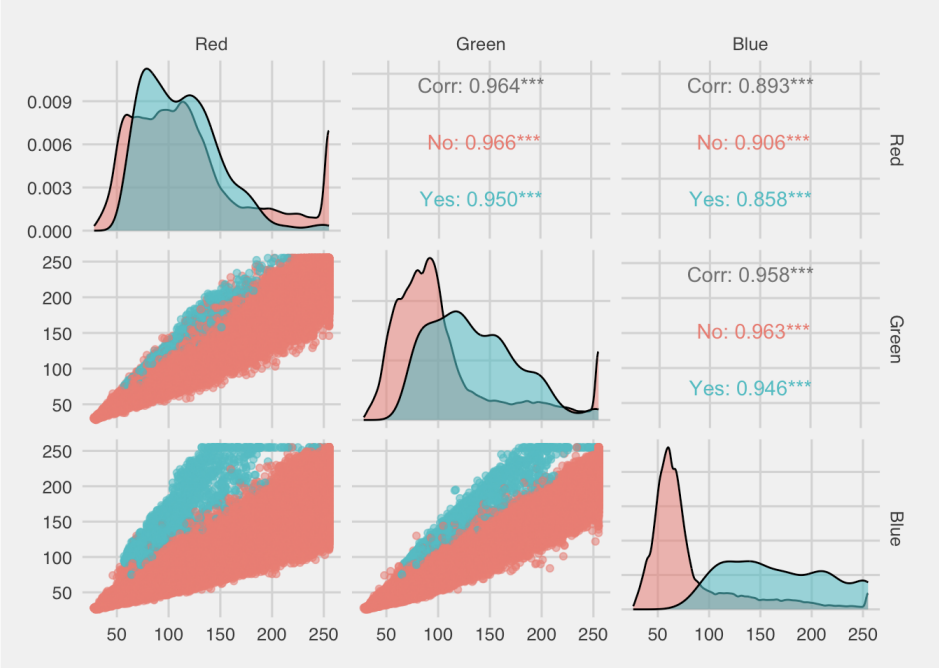


The distribution seems to give some potential indication of a correspondence between the Red, Green, and Blue values and the B1, B2, and B3 values in the holdout data set. To confirm, we will look at a correlation plot and compare the correlations in each set of data. To bring some clarity to the matrix, we build the graphic for the holdout data set based on a sample of 10% of the holdout data.

Correlation Plot of Training Data:



Correlation Matrix of 10% Sample of Holdout Data:



This makes it evident that the B3 band represents Blue. For further confirmation, I compared the mean values of the training and holdout data by class.

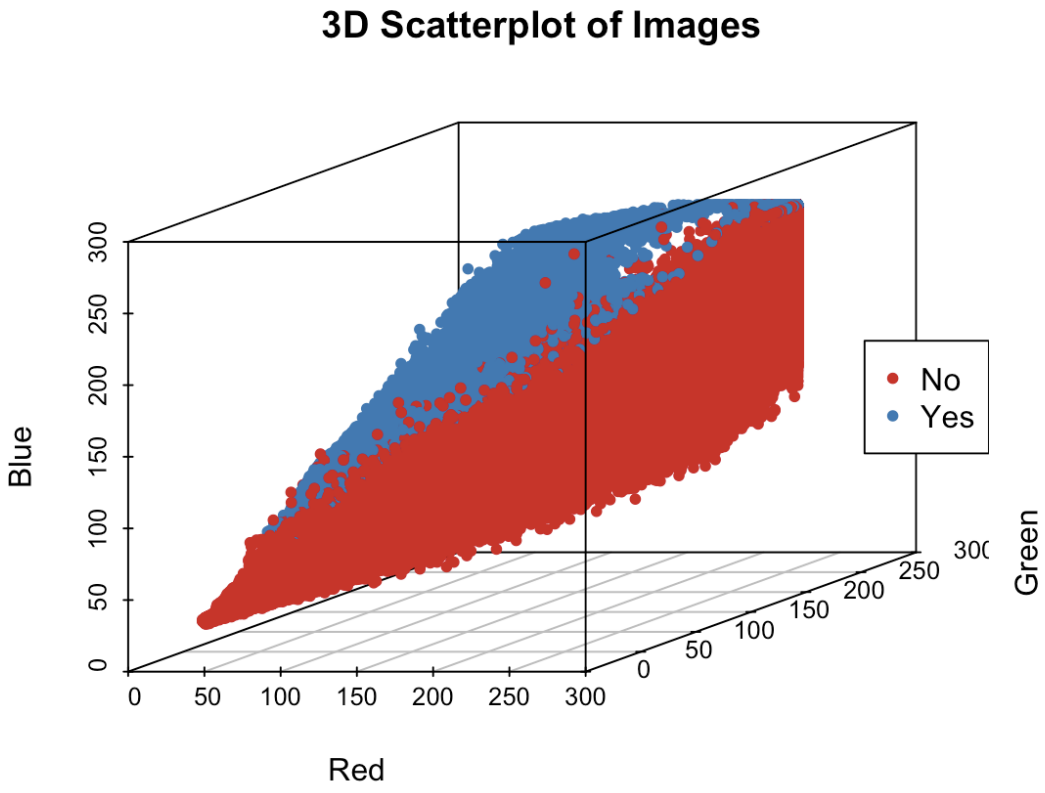
Data Set	Class	Mean Value	Mean Value	Mean Value
Holdout	BlueClass	**B1:** 111.6773	**B2:** 133.3961	**B3:** 165.2613
Training	BlueClass	**Red:** 169.6627	**Blue:** 205.0371	**Green:** 186.4149
Holdout	Non-Tarp	**B1:** 118.3081	**B2:** 105.1734	**B3:** 81.7588
Training	Non-Tarp	**Red:** 162.7604	**Blue:** 122.4993	**Green:** 152.5808

After comparing the mean values of the data, the distribution of the data, and the correlation plots, it seems reasonable to conclude that Red corresponds with the B1 value, and Green with B2.

After coming to this conclusion, we rename the columns in the holdout data to match the training data, and continue with some exploratory data analysis of the holdout data.

Before moving too deep into the exploratory data analysis, we can first make some observations based on the correlation matrix. Like the training data, for images classified as having blue tarps, and the image begins to have red pixel values over about 200, they are less likely to have blue tarps. There are almost no images with blue tarps that have green or blue pixel values below 100, and images containing a blue value of 200 or more seem to be exclusively classified as containing a blue tarp.

Finally, like with the training data, we use a 3D model to observe the separation of color data within each pixel.



Like the training data, there is a clear delineation between the pixels identified as blue tarps and those labeled as non-tarp in the 3D representation of the pixel color data.

Applying the Holdout Data to the Models

Now that we have organized and briefly examined the holdout data it is time to test the performance of each previously constructed model against the holdout data. Once applying the model to the holdout data, I compile the results in a table similar to the performance table given for the training data.[6]

Performance Table (Holdout Data)

The performance table shows the results of each model as it has been applied to the holdout data, as well as relevant metrics such as accuracy, false positive rate, false negative rate, and any tuning metrics applied.

[6] The idea for a model test function which calculates the predictive ability of the selected models against the holdout data: https://www.rpubs.com/christianaaronschroeder/799196.

Model	AUROC	Threshold	Accuracy	FalseNeg	FalsePos	Precision	Tuning	Tuning2
LOG	0.9994	0.7	0.9947	0.0052	0.0171	0.9999		
LDA	0.9980	0.7	0.9976	0.0017	0.0959	0.9993		
QDA	0.9915	0.5	0.9960	0.0018	0.3054	0.9978		
kNN	0.9627	0.4	0.9916	0.0077	0.1032	0.9992	7.00	
PLR-Ridge	0.9900	0.8	0.9937	0.0000	0.8758	0.9937	0.00	0.004
SVM	0.9991	0.5	0.9802	0.0198	0.0112	0.999	1.25	

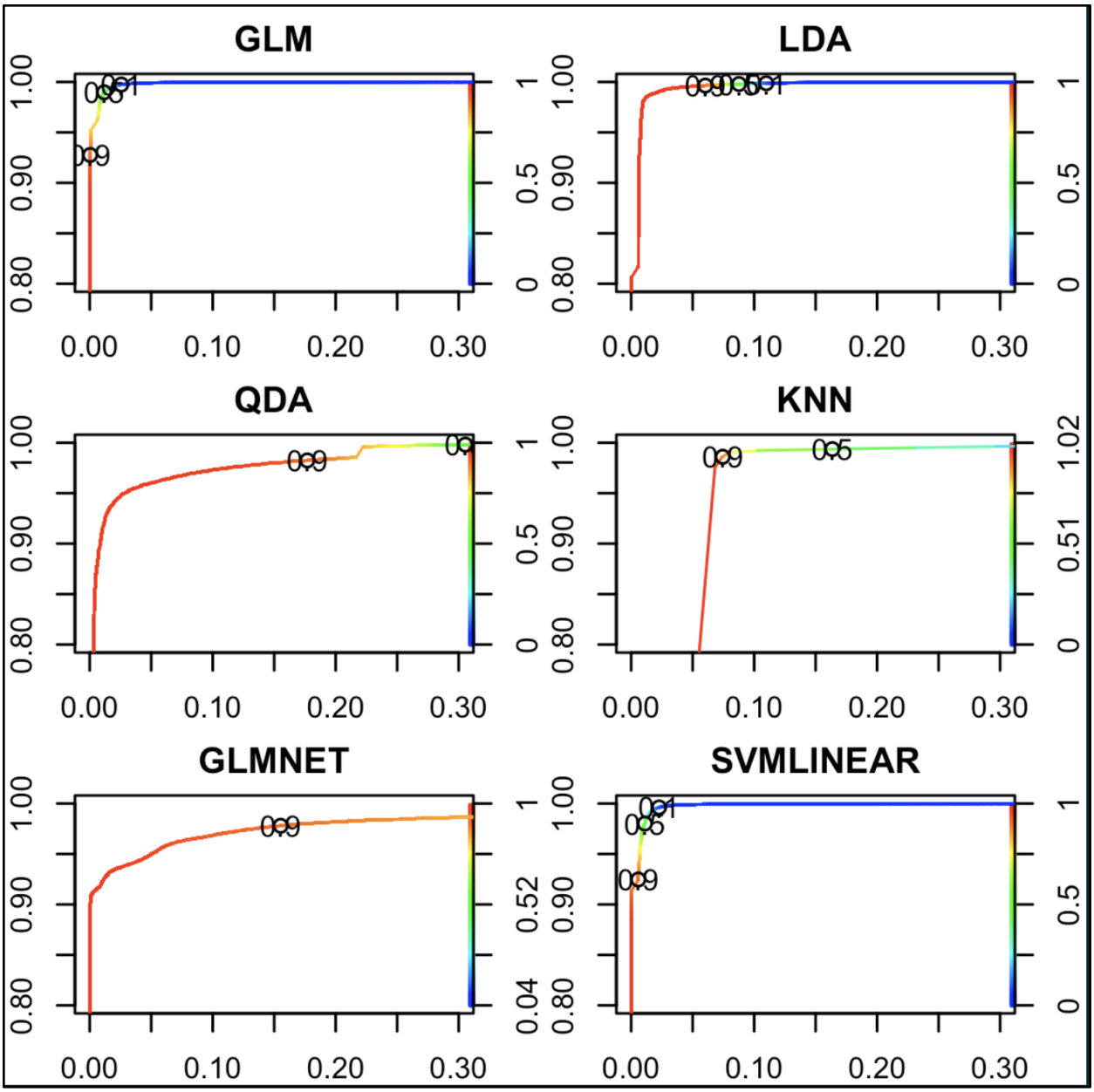
Here, we see the results of each model fitted with the holdout data. Tuning 1 refers to alpha, k, C, or mtry, while Tuning 2 refers to lambda.

Notably, both the LOG and SVM models have an accuracy of over 0.999. PLR Ridge Regression has a false negative rate of 0, followed closely by QDA and LDA. If we are interested in minimizing false negative rate while also maximizing accuracy, then LDA appears to be the strongest model.

Some interesting observations can be made when comparing the results using the holdout data with the initial results from the training data. The LDA model's accuracy actually increased from the training data to the holdout data, with a slight increase in false negative rate, but overall increase in accuracy. QDA and PLR-Ridge Regression also show a slight increase in accuracy on the holdout data in comparison to the training data. This is somewhat to be expected, considering the increased imbalance in the holdout data over the training data.

ROC Curves

Before making our conclusions, it is worth taking a look at the ROC curves for each model once applied to the holdout data. The performance chart above lists the AUROC (area under the ROC Curve) as a means of simple comparison, but seeing the charts themselves can help us to visualize the data.



Here, if we were to compare these ROC curves with those of the models applied to the training data, it is evident that most of the models had a drop in performance. Notably, SVM, LDA, and GLM are still almost completely aligned with the left and upper edges of the graphic, while the other models evidence much more of a slope. One somewhat interesting exception is the kNN model, which seems to indicate a drop in accuracy in comparison to the other models. This observation is reflected in the performance table, where we can note that the AUROC for all six models is over 0.95, which confirms our earlier conclusions regarding the seemingly-impressive accuracy of the models (accuracy that would be indeed impressive if it were not for the imbalanced nature of the data which would allow even a

model that assigns ALL pixels to a single class to be accurate approximately 98% of the time).

Conclusions

Conclusion 1: Model Selection

Based on the above analysis, it is worthwhile to note that all five models are effective in correctly predicting the presence of blue tarps. That being said, with a focus on reducing false negatives in order to ensure that as many people are aided as possible, we recommend the following models (in this order) for use in classifying image files as those containing Blue Tarps and those without Blue Tarps with the purpose of providing aid to those harmed in the disaster in Haiti.

1) LDA Regression

The LDA Regression model is the first choice of predictive model for this data. To tune this model, we chose a threshold of 0.7. It was fit using interaction terms. When tested against the holdout data, accuracy was maximized while also minimizing the false negative rate, with a false negative rate of just 0.0017, and an accuracy of 99.8%.

2) QDA Regression

The second choice of model is the QDA regression model without interaction terms with a tuning threshold of 0.5. This model minimizes false negatives (0.0018 on the holdout data), and has an only slightly-reduced accuracy rate from the QDA Regression model, still over 99%.

3) PLR Ridge Regression

The third choice of model is the PLR Ridge Regression model. To tune this model, we chose a threshold of 0.8, alpha of 0, and lambda of 0.000398. This model was fit using interaction terms. Impressively, the PLR Ridge Regression model has a false negative rate of 0. This does slightly decrease accuracy, but only barely, as the accuracy is still 99%. In the event of unlimited resources, a 0% false negative rate would be optimal for ensuring the most possible rescues. However, due to the very slightly reduced accuracy, this is the third choice of model for predicting the presence of blue tarps.

Based on the results of testing against the holdout data, I am confident that any of these three models would be effective in identifying blue tarps from image data.

Conclusion 2: Imbalanced Data

However, we must remember what was noted in the original data analysis: that this data is extremely unbalanced. There are far more image files without blue tarps than there are with blue tarps, so it is possible that a model could always predict either "yes" or "no" and still have an accuracy of over 95%. Because of this, though the accuracy of these models appears to be phenomenal, that should be "taken with a grain of salt."

Due to the imbalanced nature of the data, I am curious to see how the models perform on the test data set. My inclination is that we will see increased error rates on the testing data due at least in part to the imbalance in the data.

This article by Saito & Rehmsmeier details how, for imbalanced data sets, ROC plots and AUC scores may not be the best metrics for evaluating model performance.[7] It was outside of my current understanding of the metrics discussed in the article to incorporate them into this project, but I do believe they would be applicable to this data set and may provide additional insights into the performance of the various model.

Conclusion 3: Project Assumptions

The other concern with the data (and really, the premise of the data) is that an assumption is being made that anyone who needs aid has access to a blue tarp (and that the only blue tarps are in locations where people are in need of aid). I would suspect that there are many people in need of aid who do not have a tarp–and that there are many blue tarps in Haiti that are not being used by people in need of assistance.

This assumption is in many ways the premise of the project and has severe implications for the application of these models to the actual real-world work of providing aid to people in need. If this assumption turns out to be incorrect, the aid workers may find themselves chasing blue tarps that are in fact just serving as roofing or construction material, while passing by families in need who happen to be huddling under a palm tree rather than a blue tarp.

---

[7] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4349800/

Conclusion 4: Model Tuning

Additionally, the tuning of the models may need to be refined based on further information from the administration or organization overseeing the rescue operation. As it is, I have tuned the models to minimize false negatives, which inevitably means that there will be at least a minimal increase in false positives. In the context of this project, such false positives may mean wasting time, personnel, and financial resources accessing a location that was wrongly classified as having a blue tarp–and there being no one there to rescue (and no blue tarp). With no data on the budget or time constraints for the rescue, I opted to prioritize the need to help as many people as possible, thus risking the loss of resources on a few false positives, by minimizing the false negatives. This is something that could be refined if there is further information indicating that such resources are of limited supply.

Conclusion 5: Application to "Noisier" Environments

While a blue tarp in a sea of vegetation and soil (the largest class in the data set by far!) seems as if it could potentially be obvious even to an aerial observer, I am curious as to the potential of similar technology applied to more dense suburban and urban areas where a tarp might not be as evident. An image of a sparsely populated area has far less noise than does a busy urban center, especially in many developing countries where blue tarps are routinely used to house homes and cover outdoor markets. In these contexts, I would expect the ability of the models to accurately predict the presence of blue tarps to decline dramatically.

Along with this, it is worth considering the addition of geospatial influence / data to improve the accuracy of these models in order to reduce the risk of error from noise such as roofs, water features, etc. Perhaps some kind of filtering to identify hue and density could also be effective in improving accuracy.

Conclusion 6: Considerations for Real-World Application

Aside from the already-mentioned influence of limited resources such as finances, time, and personnel, the real-world application of this project is likely to be limited due to other factors. In a rescue situation, such as this one, factors such as access and safety of the rescuers are also important. This data on tarp location may be the first step in carrying out an effective rescue operation, but it would be necessary for rescue teams to evaluate on a much more case-by-case basis whether the operation is actually feasible.

It is also worth considering other, potentially more effective means of locating displaced persons, such as cell phone location data. During my time living in eastern Africa, the use of cell phones was prolific, with even the most impoverished villagers often owning at least a small flip phone. Unless a storm like the one affecting Haiti had also damaged communications networks, one would think that pinging cell locations might be a more accurate means of locating the presence of displaced people–perhaps even using automated text procedures to get a response from those in need of assistance.

Conclusion 7: Further Application

Though my mind is somewhat "stuck" in the application of this project to aid-driven scenarios, such as automating the process of finding boats of refugees lost as sea through continuous geo-scanning (which, if possible, I would hope would be used to provide aid to said refugees rather than imprison or deport them, but that is only my personal view of socioeconomic & global policy), I can only imagine that the applications of such models are somewhat endless with appropriate refinement. I would be curious as to whether a similar concept, applied to temperature scans of the earth, could be used to predict and prevent forest fires.

After beginning to think outside of just aid work on this topic, I began researching and found the Microsoft Geospatial Research Machine Learning page, which details a number of related projects using machine learning and geospatial imagery to do everything from identify building damage post-earthquake and natural disaster to monitor the melting of the polar ice caps.[8] I won't reproduce those ideas here, but found them extremely interesting.

[8] https://www.microsoft.com/en-us/research/project/geospatial-machine-learning/