Disaster Relief Project: Part 2

DS-6030
Sam Knisely (sck4jh), Alec Pixton (etk3pu), Darreion Bailey (rzu5vw)

Introduction

The objective of this study is to develop a methodology for identifying displaced persons following a natural disaster, exemplified by the 2010 earthquake in Haiti. The dataset utilized, named HaitiPixels.csv, was collected shortly after the earthquake and comprises aerial imagery of Haiti segmented into Red, Green, and Blue spectral bands. Each pixel group within the dataset is classified into one of five categories: Vegetation, Rooftop, Blue Tarp, Soil, or Various Non-Tarp. Given the prevalent use of blue tarps as temporary shelters by the displaced population, this research aims to leverage image data to accurately predict the locations of blue tarps, thereby aiding in the identification of areas with high concentrations of displaced individuals.

Data

The training data, HaitiPixels.csv, contains aerial pictures of Haiti after the earthquake in 2010. It contains Red, Green, and Blue pixel variables along with the Class variable that classifies the pictures into vegetation, soil, rooftop, various non-blue tarp, and blue tarp classes.

Figure 1: Correlation of Color Values

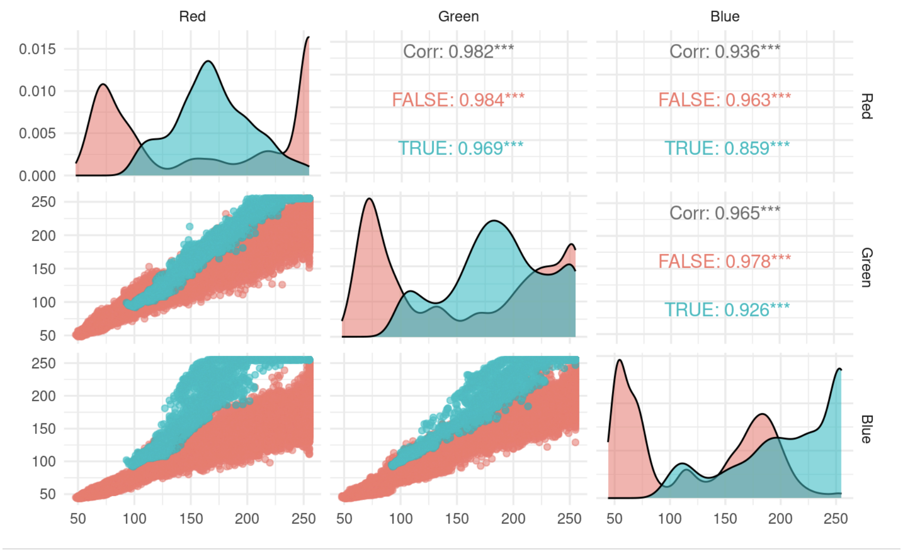
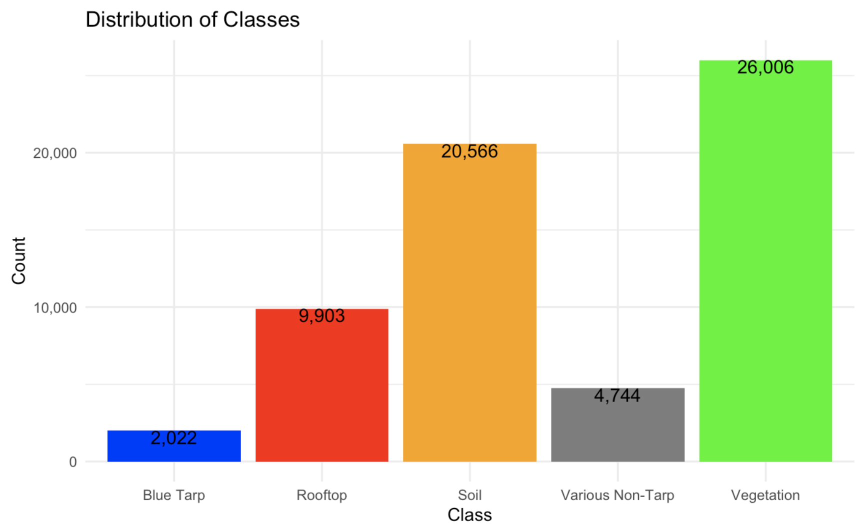


Figure 1 indicates that while there are high degrees of correlation between the spectral bands, there are still discernible differences between the TRUE and FALSE classes. These distinct distributions and high correlations will guide us as we leverage these spectral differences to classify pixels into various categories.

Figure 2: Distribution of the Class Variable



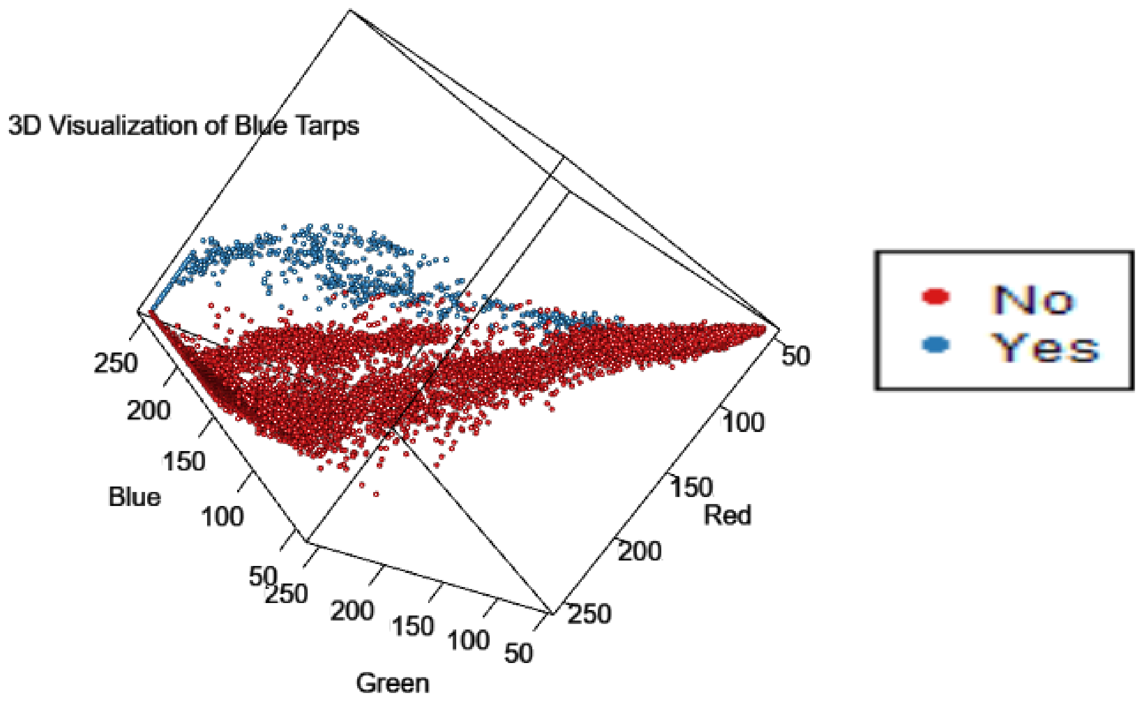
As can be seen from Figure 2, the aerial photographs mostly capture vegetation and soil, and blue tarp observations are the least common with only 2,022 observations. Since our goal is to identify Blue Tarp observations, we create a binary factor variable called BT_Class for use in our modeling that is true if the class is blue tarp and false otherwise. A numerical five-number summary of the training dataset variables is provided in Figure 3 below.

Figure 3: Training Data Summary

```
   Class                Red             Green             Blue        BT_class    
 Length:63241       Min.   : 48    Min.   : 48.0   Min.   : 44.0   FALSE:61219  
 Class :character   1st Qu.: 80    1st Qu.: 78.0   1st Qu.: 63.0   TRUE : 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                
```

We observe from Figure 3 that on average, red pixels are the most prevalent in the training data photographs followed by green and then blue pixels.

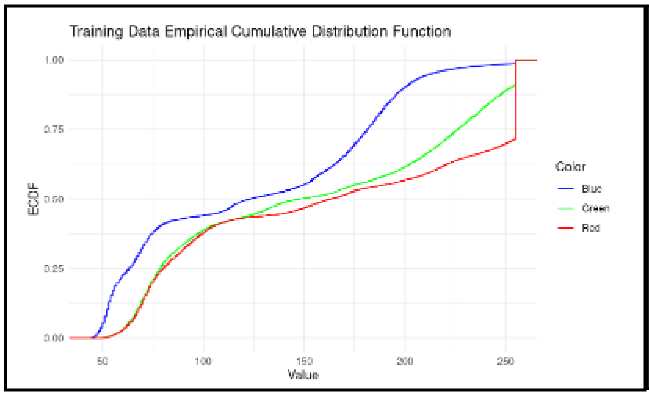
Figure 4: 3D Visualization of Blue Tarps

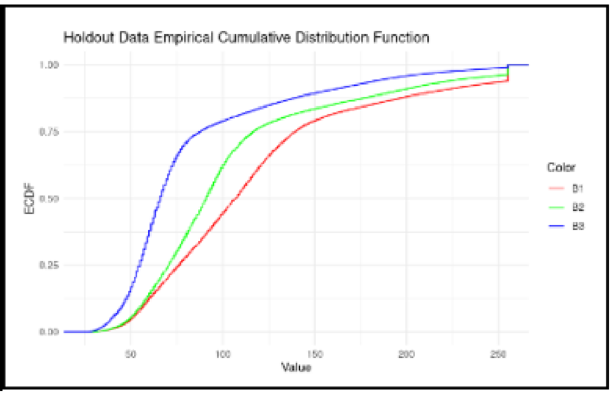


Blue spheres represent 'Yes' to the response variable 'Blue Tarp', and Red spheres represent 'No.' Here we can see that the Blue spheres tend to cluster towards higher blue values, and the Red spheres are more dispersed. Given the clear distinction between the clusters, we can infer that leveraging a thresholding approach would be useful in identifying blue tarps.

Next, we evaluate the holdout datasets. The holdout data is provided in a zip file with 11 data files, 8 text files and 3 jpeg files. We only use the text files in our analysis. All of these text files, with the exception of one file, contain the variables ID, X, Y, Map X, Map Y, Latitude, Longitude, B1, B2, and B3. There is dataset, orthovnir067_ROI_Blue_Tarps_data.txt, that is just the variables B1, B2, and B3 but we exclude this dataset from our holdout analysis because we identified that it is the same observations as orthovnir067_ROI_Blue_Tarps.txt. We append the rest of the text files and create the blue tarp binary classification variable, BT_class, based on the file names indicating whether each dataset contains blue tarp or non-blue tarp observations. We then did a test to determine which variables in the holdout dataset correspond to the Red, Green, and Blue variables similar to the training dataset. We tested this by plotting the overall empirical cumulative distribution functions for the Red, Green, and Blue variables in the training dataset and compared them to the overall empirical cumulative distribution functions for the B1, B2, and B3 variables in the holdout dataset. Please see Figure 6 below.

Figure 5: Empirical Cumulative Distribution Functions





We observe in the training data that the Blue ECDF is the highest, followed by the Green and then the Red. The Blue ECDF looks to clearly be the highest, with the Green ECDF slightly above the Red, and lastly a jump in the Red ECDF at higher values. We observe a similar pattern in the holdout data ECDFs. We see B3 clearly has the highest ECDF, followed by B2 being in the middle and B1 being the lowest. There is also a slight jump in the B1 ECDF at higher values. Since the holdout ECDFs for B1, B2, and B3 follow similar patterns as the Red, Green, and Blue ECDFS, respectively, we feel comfortable mapping the B1 variable to Red, the B2 variable to Green, and the B3 variable to Blue.

Methodology

In our analysis, we used R, leveraging several packages to streamline the process. The tidyverse package, which includes dplyr and ggplot2, facilitated data manipulation and visualization. We also utilized the tidymodels package suite, offering tools for various stages of model creation, from preprocessing to tuning. For performance metrics calculation, we used the yardstick package. Additionally, the discrim and MASS packages, in conjunction with parsnip, provided models for Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). Visualization tools such as ggcorrplot, GGally, and patchwork were also employed.

To develop our models, we employed 10-fold cross-validation. This technique divides the data into 10 groups, each serving as a validation set while the other nine groups are used to train the model. This process iterates until each group has served as a validation set once. Performance metrics, J-index, specificity, sensitivity, accuracy, and precision, are then calculated using yardstick and threshold_perf from the probably package. This also provides the optimal threshold for the metrics.

We created models for logistic regression regularization, KNN, Random Forest, and SVM (linear, polynomial, and radial basis). Models that have tunable parameters were tuned using 10-fold cross-validation as the resamples. Once parameters have been set, the workflow is finalized and the cross-validation is fit to the model. Predictions from the model fitted with the cross-validation data is used to produce an ROC curve.

For holdout data, we aggregated several files, added them to the fitted model using augment, and calculated metrics using yardstick and probably. Sensitivity, or the true positive rate, was particularly important for detecting the presence of a small Blue Tarp in a vast area with varied terrain. Specificity, the true negative rate, complements this by showing the false positive rate when represented as 1-specificity in ROC curves. ROC curves provided a crucial visual representation of the tradeoff between thresholds. In disaster relief scenarios, minimizing false positives is vital to allocating resources effectively, making precision (the proportion of true positives among all positives) a crucial metric. High precision indicates a low false positive rate.

We determined the optimal threshold for our models by maximizing the J-index, which balances sensitivity and specificity by adding them together and subtracting one. To limit false positives, we set the minimum possible threshold for evaluation at 0.05. Although we evaluated accuracy (the ratio of true predictions to total observations), it was less informative due to the imbalance in true positives and negatives. Therefore, accuracy could potentially mislead the interpretation of performance.

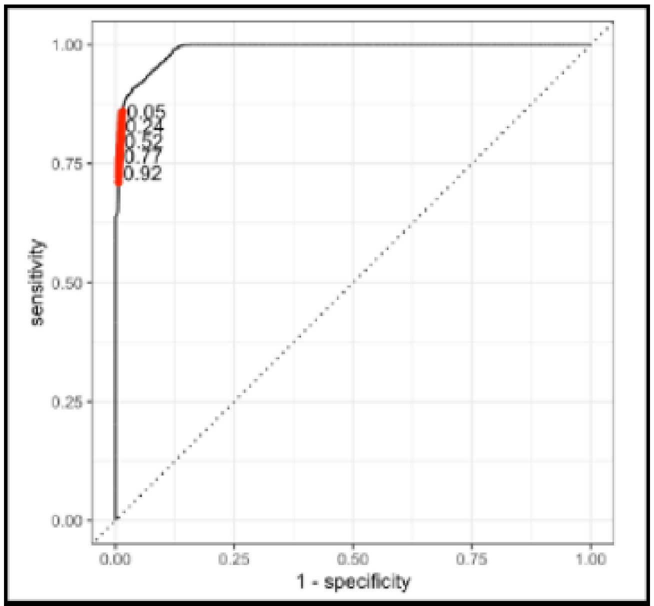
Results

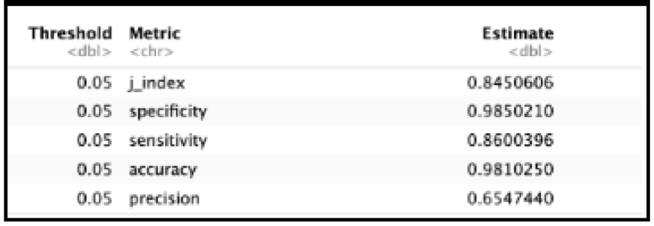
We trained our models using the random seed of '4896' for reproducibility. We performed 10-fold cross-validation for each model, found the optimal threshold, and found the performance metrics on both the training and holdout data.

Linear-Discriminant Analysis (LDA)

The 10-fold cross-validation for the LDA model yielded an accuracy of 98.39% and a roc_auc of 98.88%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below. The red dots represent various thresholds along the ROC curve.

Figure 6: LDA Cross-Validation Predictions ROC Curve and Performance Metrics



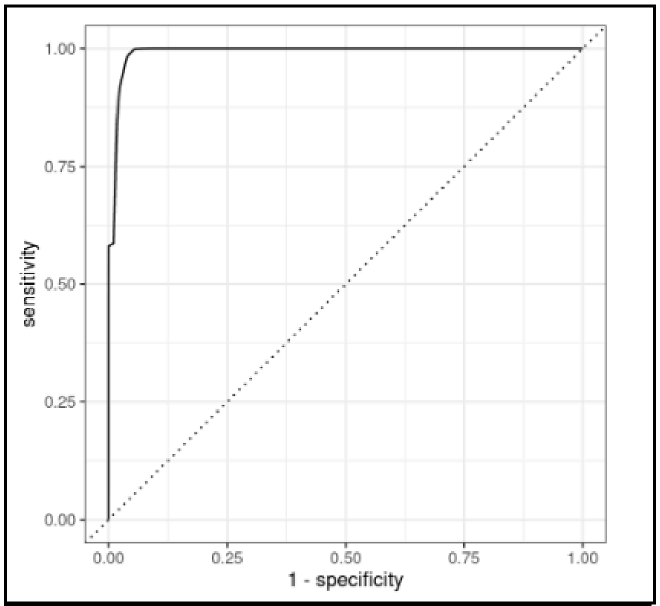


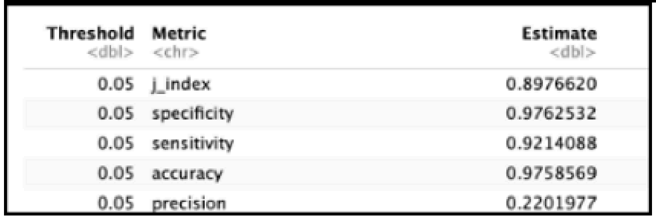
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.8450606
0.05	specificity	0.9850210
0.05	sensitivity	0.8600396
0.05	accuracy	0.9810250
0.05	precision	0.6547440

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.05. This lower threshold helps pick up observations with lower predicted blue tarp probabilities, minimizing missing displaced people. The roc_auc of the LDA model on the training data was 98.89%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found on the training data and are presented in the table above. From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.05 is 86%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 1.48%. Next, we fit the LDA model to the holdout set and evaluated the performance.

First, we plot the ROC curve of the model on the holdout data.

Figure 7: LDA Holdout Data ROC Curve and Performance Metrics





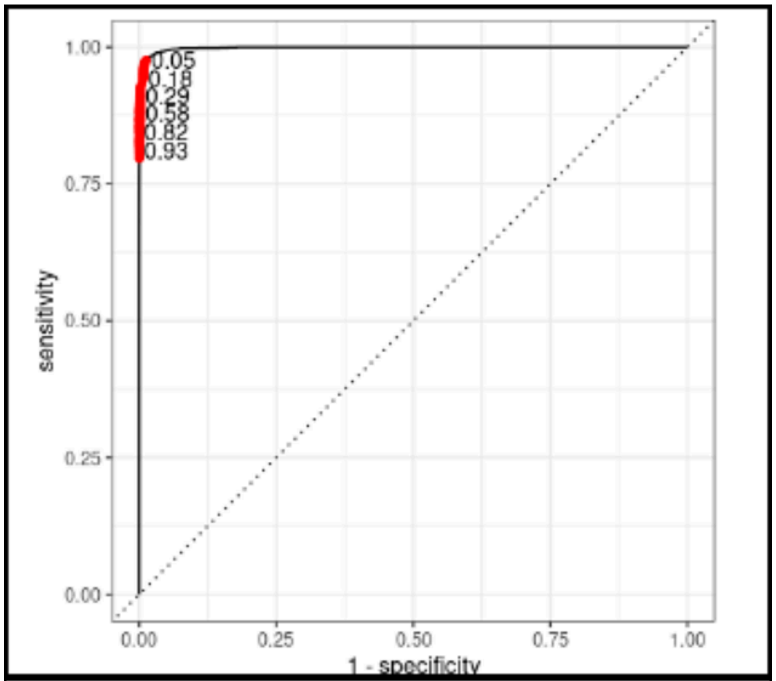
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.8976620
0.05	specificity	0.9762532
0.05	sensitivity	0.9214088
0.05	accuracy	0.9758569
0.05	precision	0.2201977

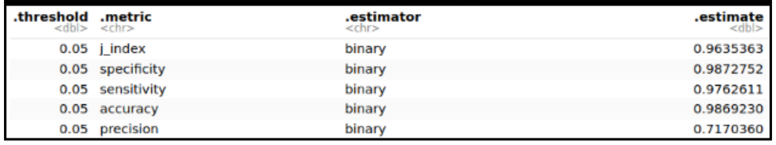
The roc_auc of the LDA model on the holdout data was 99.21%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the LDA model on the holdout data at the optimal threshold of 0.05 is 92.14%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 2.37%.

Logistic Regression

Using logistic regression with 10-fold cross validation produced an accuracy of 99.53%. The ROC_AUC is 99.84% and the ROC curve is shown below. The optimal threshold of 0.05 was determined by maximizing the J-index. This was the lower bound of thresholds searched, any lower would lead to a significant rise in false positives. The ROC curve is shown with several threshold values indicated.

Figure 8: Logistic Cross-Validation ROC Curve and Performance Metrics

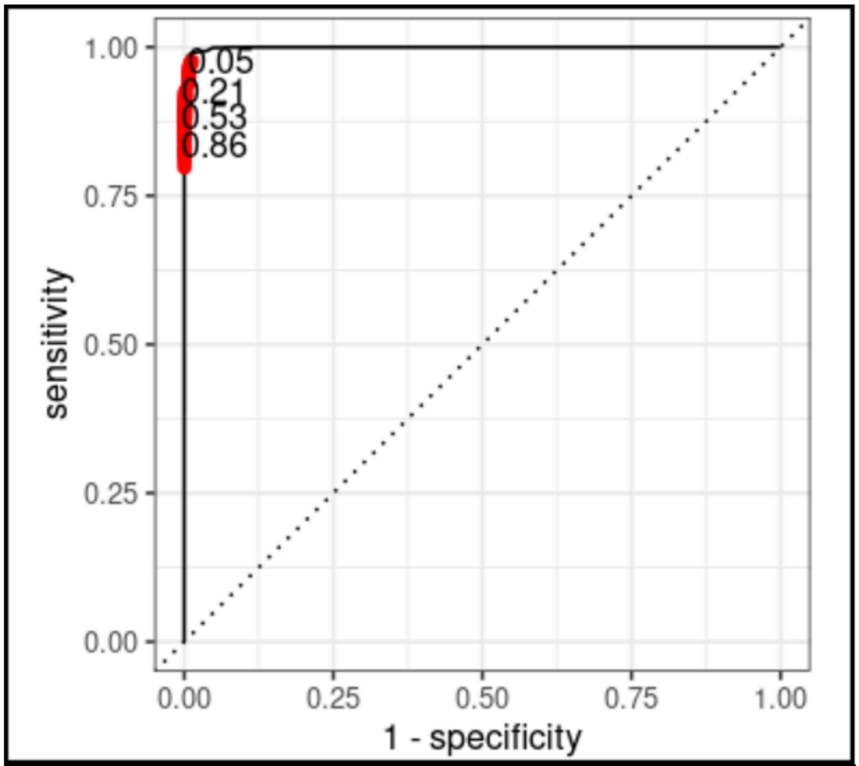


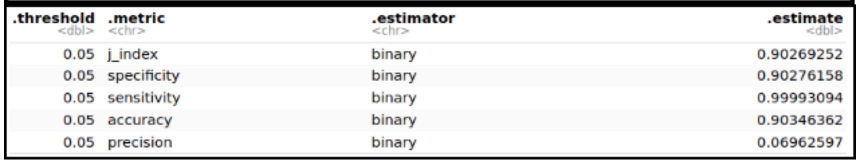


.threshold <dbl>	.metric <chr>	.estimator <chr>	.estimate <dbl>
0.05	j_index	binary	0.9635363
0.05	specificity	binary	0.9872752
0.05	sensitivity	binary	0.9762611
0.05	accuracy	binary	0.9869230
0.05	precision	binary	0.7170360

The j-index, specificity, sensitivity, accuracy, and precision of the logistic regression model are shown above using 0.05 as the threshold. Each metric is high except for precision. This provides evidence that precision may be particularly useful in determining which model performs best. The holdout data was predicted and evaluated resulting in the below ROC curve.

Figure 9: Logistic Holdout Data ROC Curve and Performance Metrics





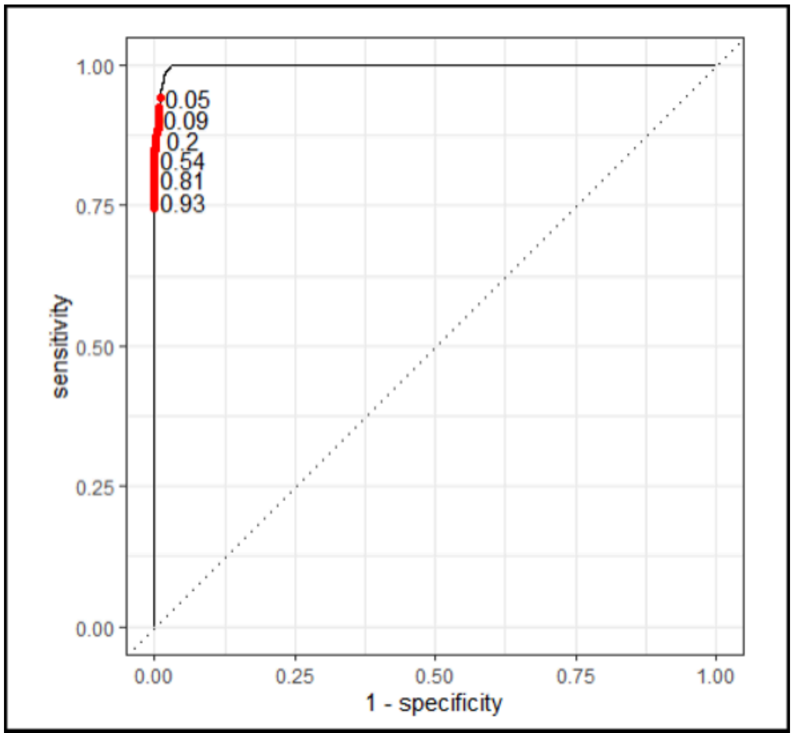
.threshold <dbl>	.metric <chr>	.estimator <chr>	.estimate <dbl>
0.05	j_index	binary	0.90269252
0.05	specificity	binary	0.90276158
0.05	sensitivity	binary	0.99993094
0.05	accuracy	binary	0.90346362
0.05	precision	binary	0.06962597

Once again, the j-index, specificity, sensitivity, accuracy, and precision of the model were calculated and are shown above. The false positive rate at this threshold is computed as 1 - specificity, resulting in a false positive rate of 9.73%.

Quadratic Discriminant Analysis (QDA)

The Quadratic Discriminant Analysis (QDA) with 10-fold cross-validation resulted in an accuracy of 99.46% and a ROC AUC of 99.82%. Using the predictions, the below ROC curve has been produced.

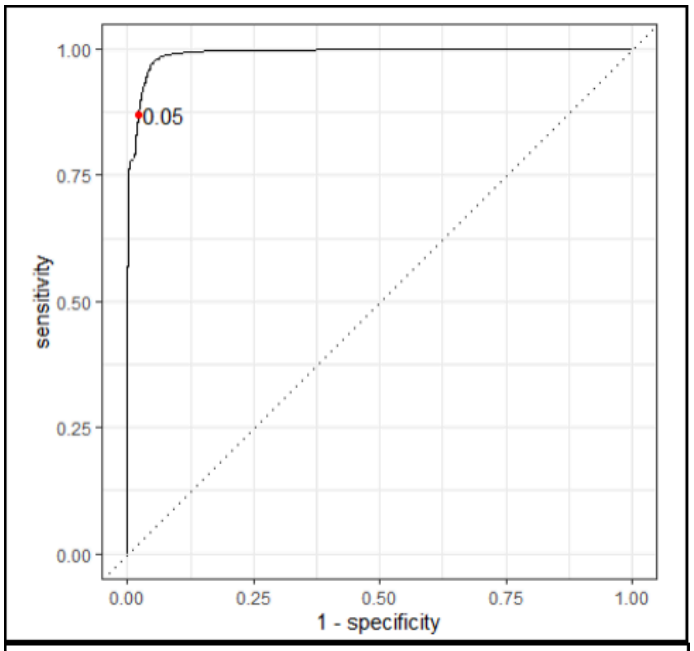
Figure 10: QDA Cross-Validation Predictions ROC Curve and Performance Metrics

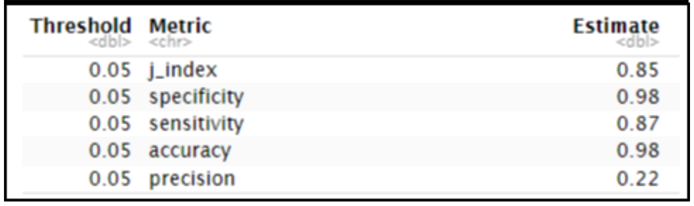


Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.93
0.05	specificity	0.99
0.05	sensitivity	0.94
0.05	accuracy	0.99
0.05	precision	0.75

This led to an optimal threshold, maximizing the J-index on the training dataset, of 0.05. This lower threshold increases model sensitivity so it picks up observations with lower predicted probabilities of blue tarps and decreases the chances of missing displaced people. Below is the ROC curve with the model fitted to the training data. It returned a QDA model with an ROC AUC of 99.82% on training data. Then, it computed the J-index, specificity, sensitivity, accuracy, and precision measures of the model at the optimal threshold of 0.05 and are shown in the table above. The output indicates that the true positive rate (sensitivity) for the QDA model on the training data at the optimal threshold of 0.05 is 94.21%. The false positive rate at this threshold is calculated as 1 - specificity, resulting in a false positive rate of 1.01%. Subsequently, the QDA model was fitted to the holdout dataset to evaluate its performance. First, we plot the ROC curve of the model on the holdout data.

Figure 11: QDA Holdout Data ROC Curve and Performance Metrics





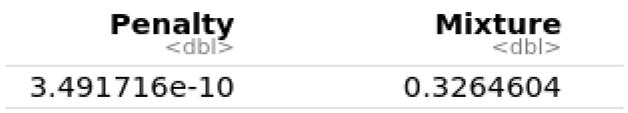
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.85
0.05	specificity	0.98
0.05	sensitivity	0.87
0.05	accuracy	0.98
0.05	precision	0.22

It returned a ROC AUC of 99.15% on the holdout data for the QDA model. Then, calculations for the J-index, specificity, sensitivity, accuracy, and precision at the optimal threshold of 0.05 were obtained for the model, as shown in the table above. The output indicates that the true positive rate (sensitivity) for the QDA model on the holdout data at the optimal threshold of 0.05 is 87.00%. The false positive rate at this threshold is computed as 1 - specificity, resulting in a false positive rate of 2.22%.

Penalized Logistic Regression

An elastic net logistic regression combines two penalties called lasso (L1) and ridge (L2). The model contains two tunable parameters, penalty and mixture. Penalty is the strength of the penalty applied to the model. Mixture is the ratio between lasso and ridge methods of penalty. Lasso regression can be used to reduce the coefficients of the model, even reducing some to zero. Ridge regression is used to bring coefficients closer to one another.

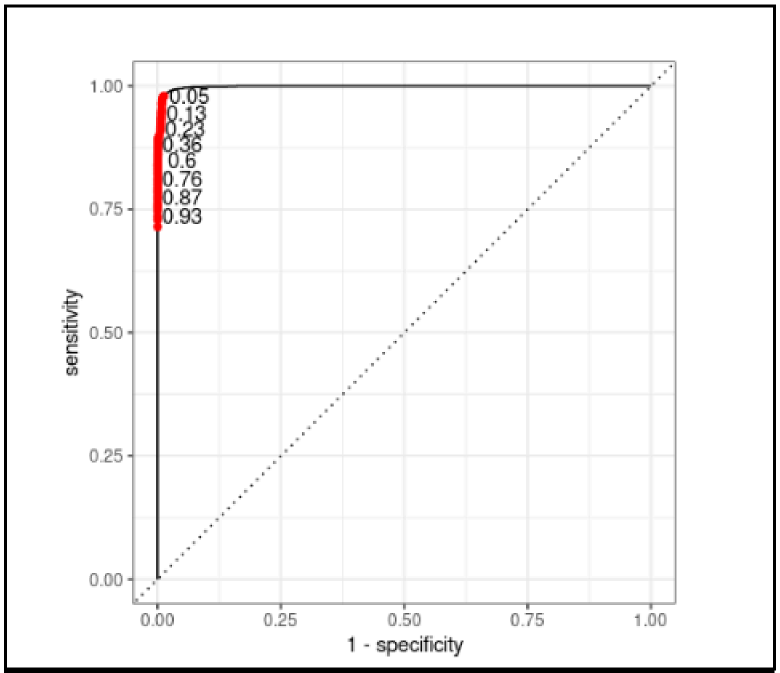
Figure 12: Penalized Logistic Regression Tuning Parameters

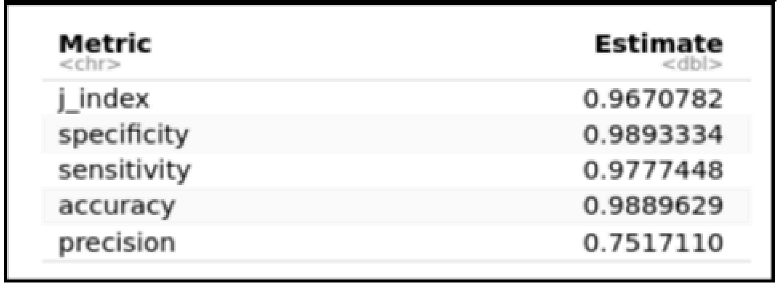


Penalty <dbl>	Mixture <dbl>
3.491716e-10	0.3264604

Optimization was performed using a tune grid on 10-fold cross validation. The optimal mixture, 0.33, indicates more ridge regression than lasso regression. However, the penalty is very close to 0, which means the penalties have no effect on the model. The accuracy of the model is 99.50% and the ROC is 99.86%. The resulting ROC curve is shown below with several tested thresholds shown in red.

Figure 13: Penalized Logistic Regression Cross-Validation ROC Curve and Performance Metrics

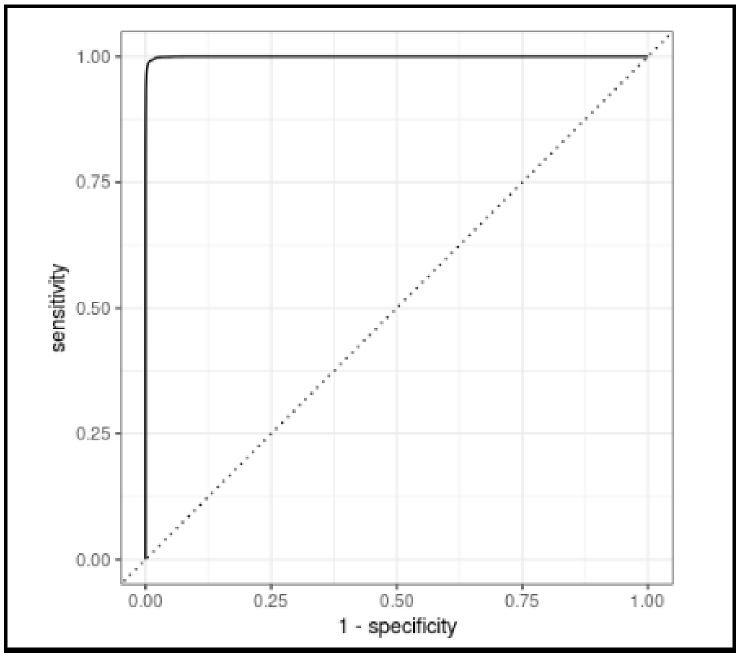


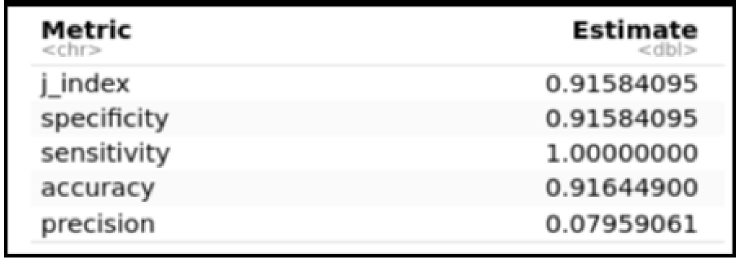


Metric <chr>	Estimate <dbl>
j_index	0.9670782
specificity	0.9893334
sensitivity	0.9777448
accuracy	0.9889629
precision	0.7517110

The best threshold to optimize J-index was found to be 0.06. Metrics at this threshold are shown in the table above. The logistic regression without any tuning had a cross-validation precision of 0.71, this suggests that the tuned model performs slightly better. To evaluate the final performance of the model the holdout dataset was fit and an ROC curve produced.

Figure 14: Penalized Logistic Regression Holdout Data ROC Curve and Performance Metrics





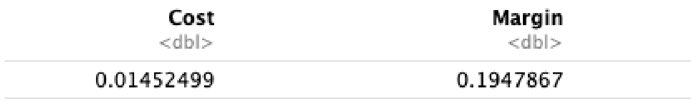
Metric <chr>	Estimate <dbl>
j_index	0.91584095
specificity	0.91584095
sensitivity	1.00000000
accuracy	0.91644900
precision	0.07959061

There is a large drop in precision for the holdout set. This drop indicates overfitting making this model a poor candidate. However, the holdout precision for the tuned model is 0.0796 and for the untuned model 0.0696. This supports that the tuned logistic regression performs better than the untuned logistic regression.

Linear Support Vector Machine (SVM)

A linear support vector machine model has two tunable parameters, cost and margin. First, bayesian optimization is used to identify these model parameters by optimizing the roc_auc on the 10-fold resamples object from the training data using 25 search iterations. These model parameters are output below.

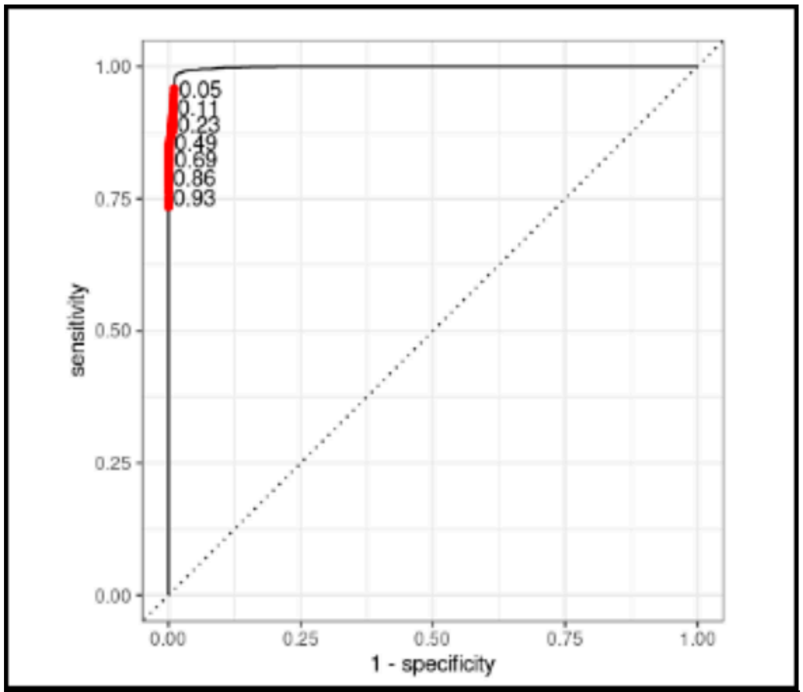
Figure 15: Linear SVM Tuning Parameters

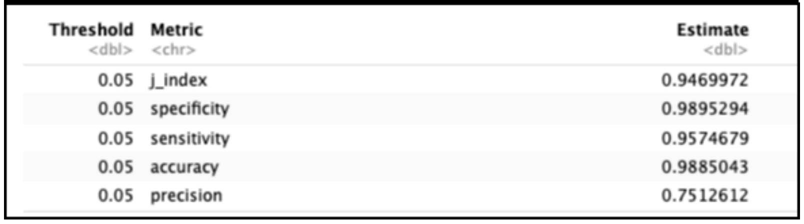


Cost <dbl>	Margin <dbl>
0.01452499	0.1947867

The cost parameter, 0.01452499, indicates the weight of the penalty applied to misclassified observations. The margin parameter, 0.1947867, indicates the distance from the decision boundary to the closest data points from any class.
Next, 10-fold cross-validation for the tuned linear SVM model yielded an accuracy of 99.17% and a roc_auc of 99.81%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below. The red dots represent various thresholds along the ROC curve.

Figure 16: Linear SVM Cross-Validation Predictions ROC Curve and Performance Metrics



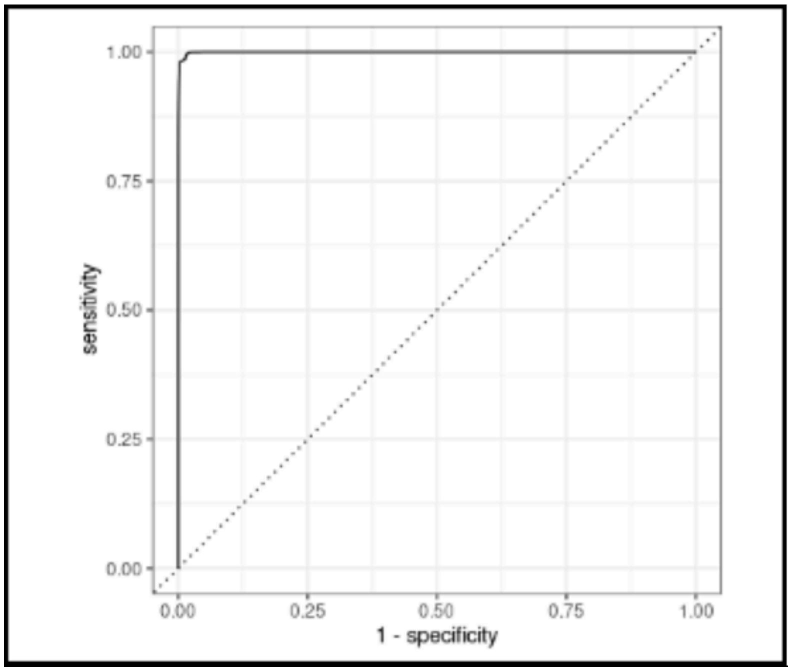


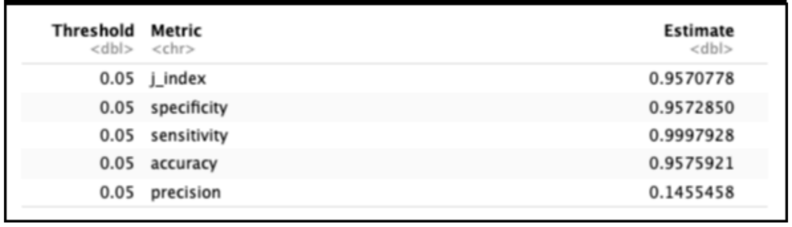
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.9469972
0.05	specificity	0.9895294
0.05	sensitivity	0.9574679
0.05	accuracy	0.9885043
0.05	precision	0.7512612

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.05. The roc_auc of the linear SVM model on the training data was 99.80%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found on the training data and are presented in the table above. From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.05 is 95.75%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 1.05%.

Next, we fit the linear SVM model to the holdout set and evaluated the performance. First, we plot the ROC curve of the model on the holdout data.

Figure 17: Linear SVM Holdout Data ROC Curve and Performance Metrics





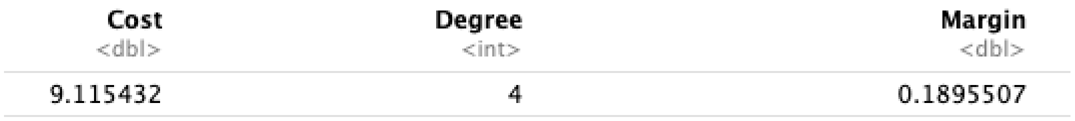
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.9570778
0.05	specificity	0.9572850
0.05	sensitivity	0.9997928
0.05	accuracy	0.9575921
0.05	precision	0.1455458

The roc_auc of the linear SVM model on the holdout data was 99.96%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the linear SVM model on the holdout data at the optimal threshold of 0.05 is 99.98%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 4.27%.

Polynomial Support Vector Machine (SVM)

A polynomial support vector machine model has four tunable parameters, cost, margin, degree and scale factor. In this model, the cost, degree and margin parameters are tuned. First, bayesian optimization is used to identify these three model parameters by optimizing the roc_auc on the 10-fold resamples object from the training data using 25 search iterations. These model parameters are output below.

Figure 18: Polynomial SVM Tuning Parameters

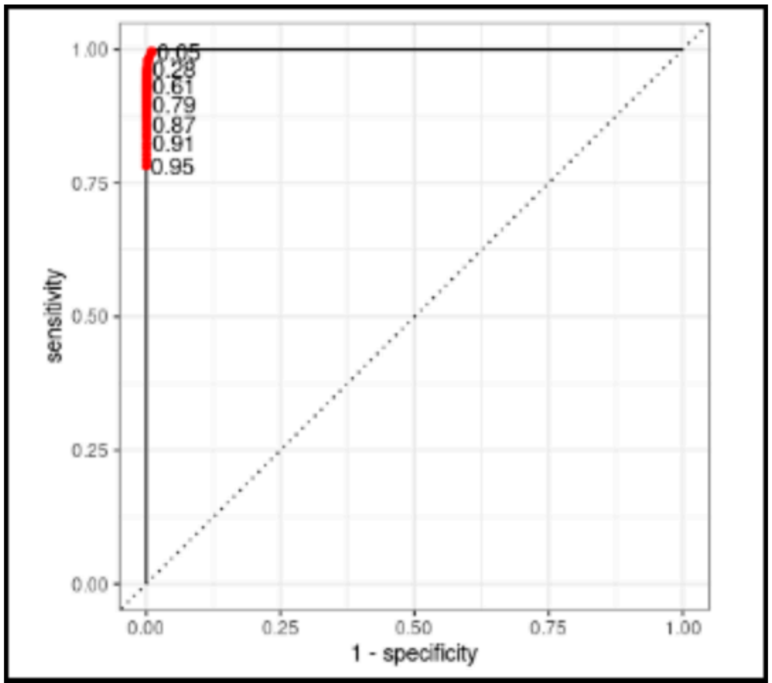


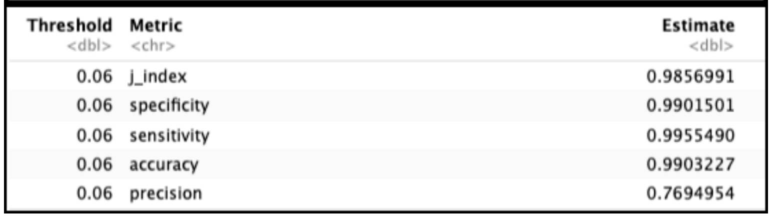
Cost <dbl>	Degree <int>	Margin <dbl>
9.115432	4	0.1895507

The cost parameter, 9.115432, indicates the weight of the penalty applied to misclassified observations. We observe that this cost parameter is

much larger than the cost parameter from the linear SVM model. This larger cost parameter indicates the polynomial SVM is heavily penalizing misclassified observations and may be overfitting. The margin parameter, 0.1895507, indicates the distance from the decision boundary to the closest data points from any class. Lastly, the degree, 4, is the power that the features in the polynomial SVM are raised to. A higher degree such as 4 indicates that the decision boundary may be complex and again that the model may be overfitting.
Next, 10-fold cross-validation for the tuned polynomial SVM model yielded an accuracy of 99.69% and a roc_auc of 99.97%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below. The red dots represent various thresholds along the ROC curve.

Figure 19: Polynomial SVM Cross-Validation Predictions ROC Curve and Performance Metrics



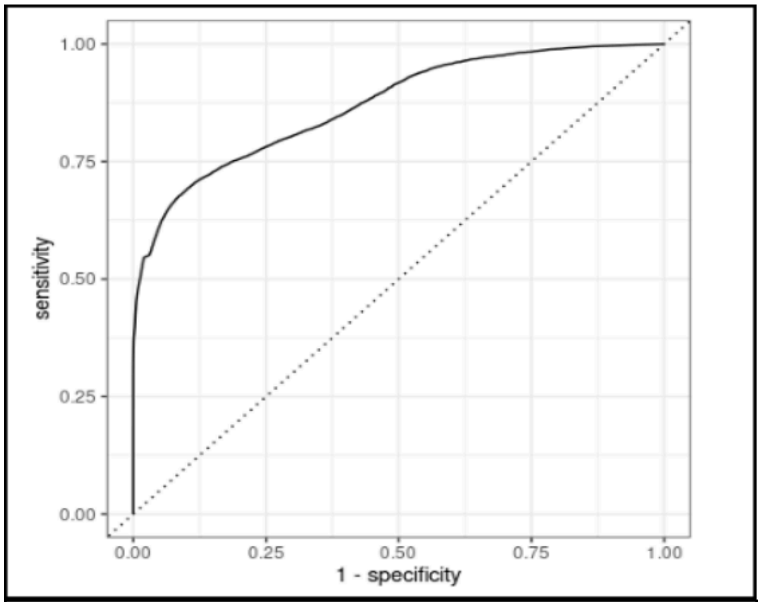


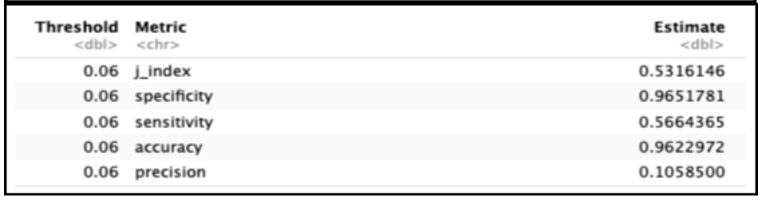
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.06	j_index	0.9856991
0.06	specificity	0.9901501
0.06	sensitivity	0.9955490
0.06	accuracy	0.9903227
0.06	precision	0.7694954

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.06. The roc_auc of the polynomial SVM model on the training data was 99.97%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.06 were found on the training data and are presented in the table above.From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.06 is 99.55%. We find the false positive rate at the optimal threshold of 0.06 by taking 1 - specificity, which yields a false positive rate of 0.98%.

Next, we fit the polynomial SVM model to the holdout set and evaluated the performance. First, we plot the ROC curve of the model on the holdout data.

Figure 20: Polynomial SVM Holdout Data ROC Curve and Performance Metrics





Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.06	j_index	0.5316146
0.06	specificity	0.9651781
0.06	sensitivity	0.5664365
0.06	accuracy	0.9622972
0.06	precision	0.1058500

The roc_auc of the polynomial SVM model on the holdout data was 87.03%. This is a smaller roc_auc than observed on the training data and we also the the ROC curve not performing as well on the holdout data. This reinforces the idea that this polynomial SVM model may be overfitting on the training data.

Next, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.06 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the polynomial SVM model on the holdout data at the optimal threshold of 0.06 is 56.64%. We find the false positive rate at the optimal threshold of 0.06 by taking 1 - specificity, which yields a false positive rate of 3.48%.

Radial Basis Function (RBF) Support Vector Machine (SVM)

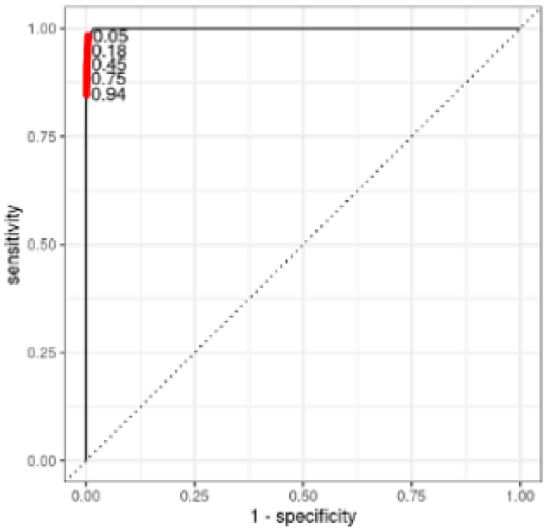
A RBF support vector machine model has three tunable parameters, cost, margin and RBF sigma. First, bayesian optimization is used to identify these three model parameters by optimizing the roc_auc. These model parameters are output below.

Figure 21: RBF SVM Tuning Parameters

Cost <dbl>	RBF_sigma <dbl>	Margin <dbl>
0.03014612	0.9990596	0.08696356

The cost parameter, 0.03014612, indicates the weight of the penalty applied to misclassified observations. This smaller cost parameter indicates the RBF SVM is not penalizing misclassified observations very much. The margin parameter, 0.08696356, indicates the distance from the decision boundary to the closest data points from any class. Lastly, the RBF sigma, 0.9990596, indicates the influence of each training point on the decision boundary.
Next, 10-fold cross-validation for the tuned RBF SVM model yielded an accuracy of 99.39% and a roc_auc of 99.95%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below. The red dots represent various thresholds along the ROC curve.

Figure 22: RBF SVM Cross-Validation Predictions ROC Curve and Performance Metrics

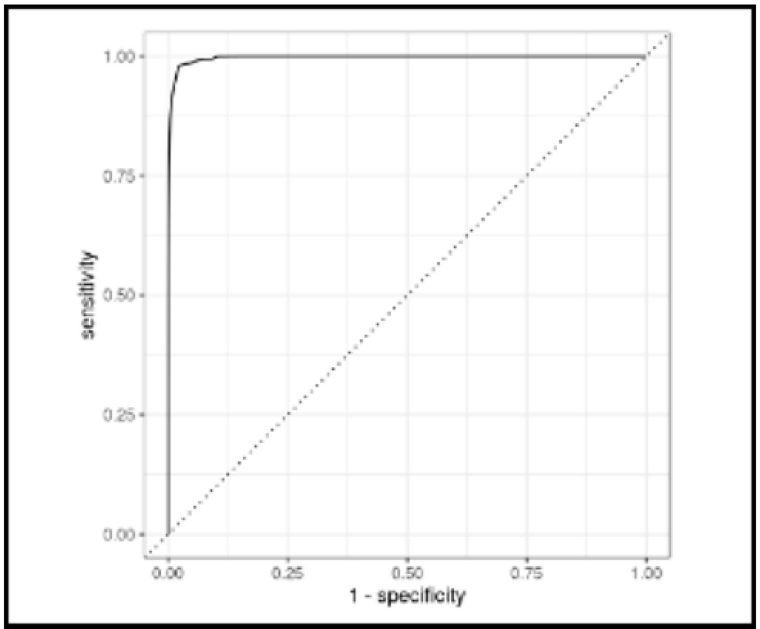


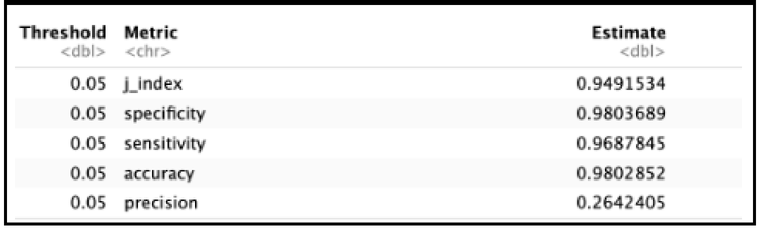
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.9781538
0.05	specificity	0.9949689
0.05	sensitivity	0.9831850
0.05	accuracy	0.9945921
0.05	precision	0.8658537

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.05. The roc_auc of the polynomial SVM model on the training data was 99.95%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found on the training data and are presented in the table above. From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.05 is 98.32%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 0.5%.

Next, we fit the RBF SVM model to the holdout set and evaluated the performance. First, we plot the ROC curve of the model on the holdout data.

Figure 23: RBF SVM Holdout Data ROC Curve and Performance Metrics





Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.05	j_index	0.9491534
0.05	specificity	0.9803689
0.05	sensitivity	0.9687845
0.05	accuracy	0.9802852
0.05	precision	0.2642405

The roc_auc of the RBF SVM model on the holdout data was 99.71%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.05 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the RBF SVM model on the holdout data at the optimal threshold of 0.05 is 96.88%. We find the false positive rate at the optimal threshold of 0.05 by taking 1 - specificity, which yields a false positive rate of 1.96%.

k-Nearest Neighbors (KNN)

A k-Nearest Neighbors model has one primary tuning parameter, 'Neighbors'. First, bayesian optimization is used to identify these model parameters by optimizing the roc_auc on the 10-fold resamples object from the training data using 25 search iterations. These model parameters are output below.

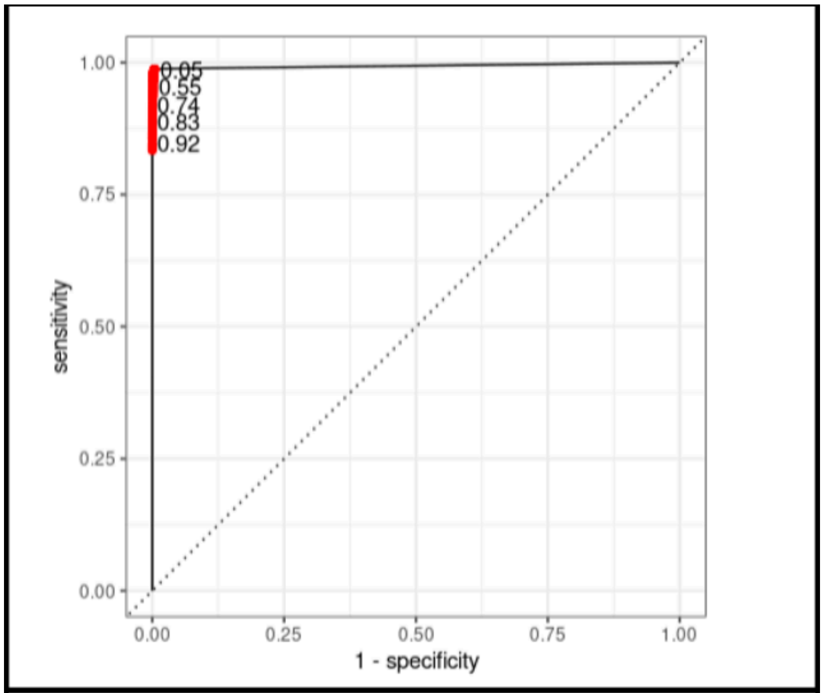
Figure 24: KNN Tuning Parameters

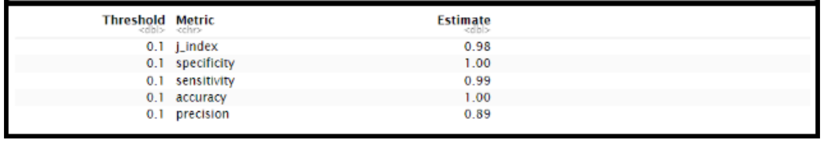
Neighbors <int>
14

The Neighbors parameter being '14' indicates that for each new data point being predicted, the algorithm will consider the 14 closest data points in the training set to determine the class.

Next, 10-fold cross-validation for the tuned KNN model yielded an accuracy of 99.99% and a roc_auc of 99.94%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below. The red dots represent various thresholds along the ROC curve.

Figure 25: KNN Cross-Validation Predictions ROC Curve and Performance Metrics



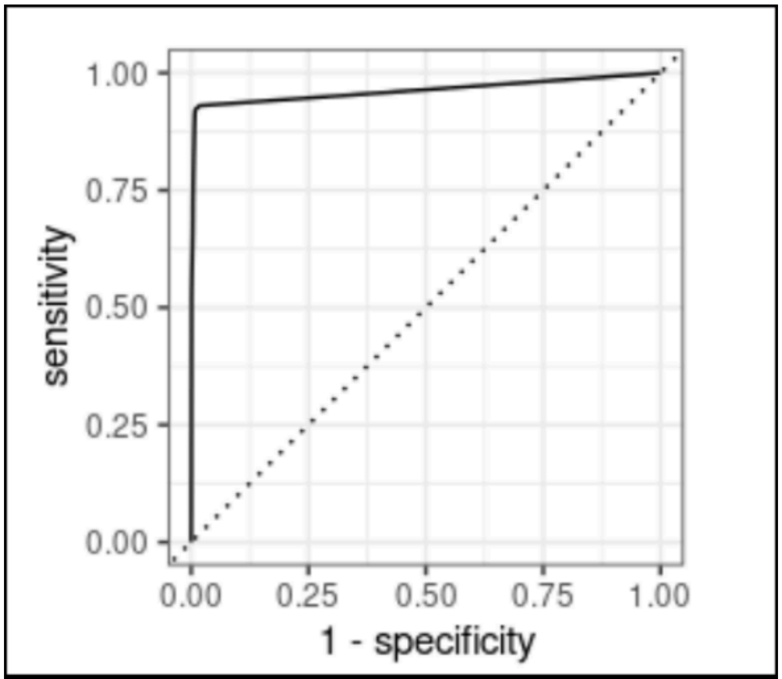


Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.1	j_index	0.98
0.1	specificity	1.00
0.1	sensitivity	0.99
0.1	accuracy	1.00
0.1	precision	0.89

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.01. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.01 were found on the training data and are presented in the table above. From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.01 is 99%. We find the false positive rate at the optimal threshold of 0.01 by taking 1 - specificity, which yields a false positive rate of ~0%

Next, we fit the KNN model to the holdout set and evaluated the performance. First, we plot the ROC curve of the model on the holdout data.

Figure 26: KNN Holdout Data ROC Curve and Performance Metrics



Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.1	j_index	0.91
0.1	specificity	0.99
0.1	sensitivity	0.93
0.1	accuracy	0.99
0.1	precision	0.33

The roc_auc of the KNN model on the holdout data was 99.77%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.01 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the KNN model on the holdout data at the optimal threshold of 0.01 is 93%. We find the false positive rate at the optimal threshold of 0.01 by taking 1 - specificity, which yields a false positive rate of ~1%.

Random Forest Ensemble

A Random Forest model has two primary tuning parameters, 'mtry', and 'min_n'. First, bayesian optimization is used to identify these model parameters by optimizing the roc_auc on the 10-fold resamples object from the training data using 25 search iterations. These model parameters are output below.

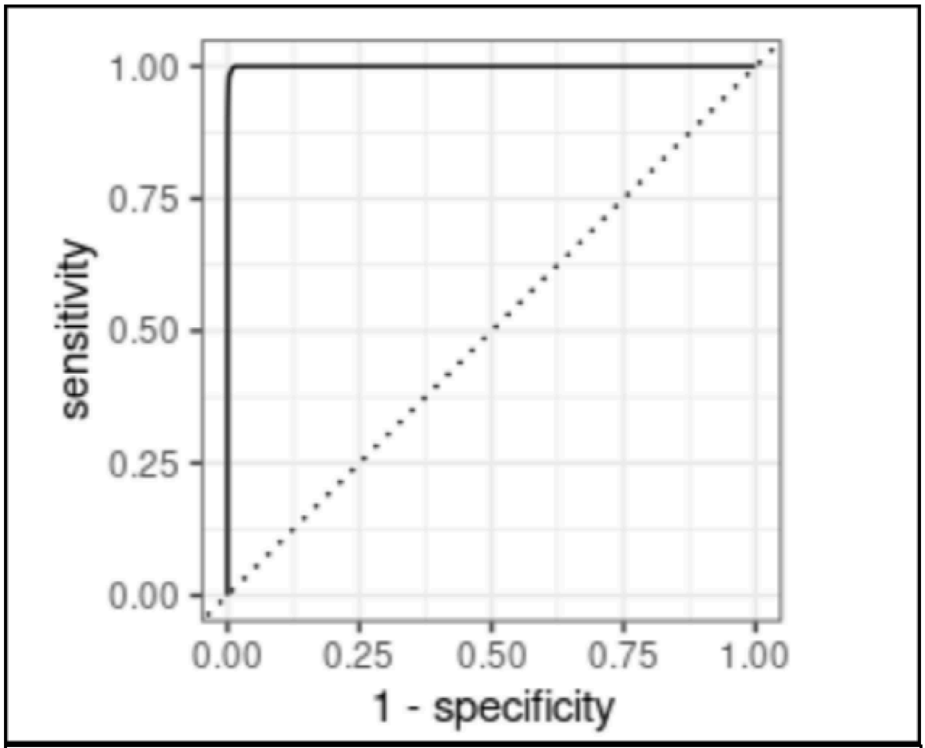
Figure 27: Random Forest Tuning Parameters

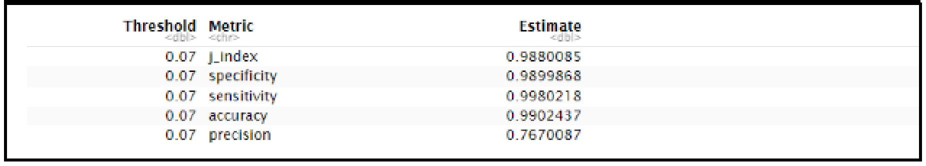
Variables per Split (mtry) <int>	Number of trees (min_n) <int>
2	16

The 'mtry' parameter being 2 indicates that the number of features that will be considered at each split point in the tree will be 2, creating a layer of randomness that helps to make individual trees less correlated. The 'min_n' parameter being 16 indicates that the minimum number of samples required for each split is 16, thereby controlling the complexity of each tree and preventing overfitting.

Next, 10-fold cross-validation for the tuned Random Forest model yielded an accuracy of 99.99% and a roc_auc of 99.93%. The ROC curve of the predictions from the 10-fold cross-validation is displayed below.

Figure 28: Random Forest Cross-Validation Predictions ROC Curve and Performance Metrics



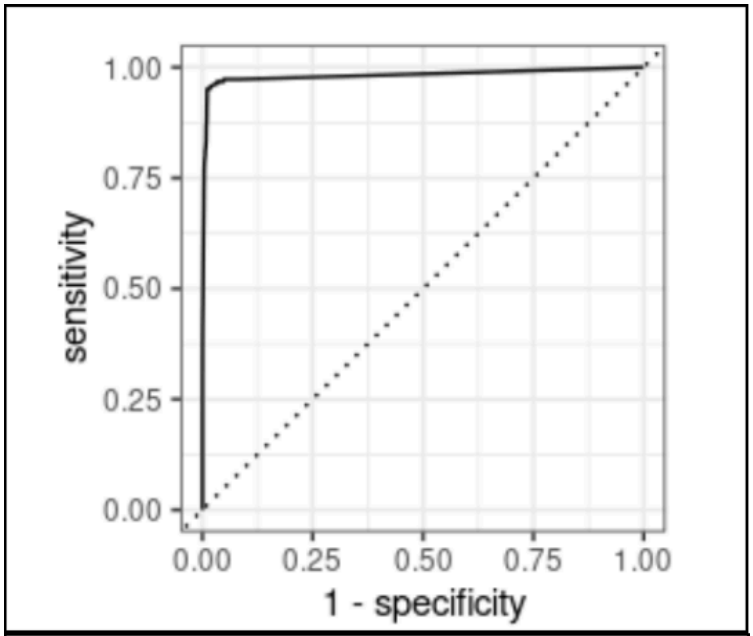


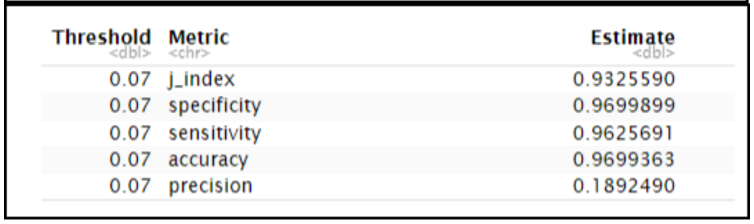
Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.07	j_index	0.9880085
0.07	specificity	0.9899868
0.07	sensitivity	0.9980218
0.07	accuracy	0.9902437
0.07	precision	0.7670087

Then, the optimal threshold was found by finding the threshold that optimized the j-index from this cross-validation. This yielded a threshold of 0.07. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.07 were found on the training data and are presented in the table above. From this output, we find that the true positive rate (sensitivity) at the optimal threshold of 0.01 is 99.8%. We find the false positive rate at the optimal threshold of 0.07 by taking 1 - specificity, which yields a false positive rate of ~1%

Next, we fit the Random Forest model to the holdout set and evaluated the performance. First, we plot the ROC curve of the model on the holdout data.

Figure 29: Random Forest Holdout Data ROC Curve and Performance Metrics





Threshold <dbl>	Metric <chr>	Estimate <dbl>
0.07	j_index	0.9325590
0.07	specificity	0.9699899
0.07	sensitivity	0.9625691
0.07	accuracy	0.9699363
0.07	precision	0.1892490

The roc_auc of the Random Forest model on the holdout data was 97.51%. Then, the j-index, specificity, sensitivity, accuracy, and precision of the model at the optimal threshold of 0.07 were found and are presented in the table above. From this output, we find that the true positive rate (sensitivity) for the Random Forest model on the holdout data at the optimal threshold of 0.07 is 96%. We find the false positive rate at the optimal threshold of 0.01 by taking 1 - specificity, which yields a false positive rate of ~3%.

Conclusions

Conclusion 1 - Limitation of Blue Tarp Assumption

A big concern about the dataset is the assumption that the people in need have access to blue tarps, and that blue tarps exclusively indicate locations where aid is required. This assumption raises a problem, because it is likely that many people in need do not have a blue tarp, and that many blue tarps present in Haiti may not indicate those requiring help.

This assumption significantly impacts the applicability of the models to real-world humanitarian efforts. If this assumption is inaccurate, humanitarian workers might be misdirected, focusing on blue tarps used for farming purposes or as market stall coverings, while overlooking families seeking shelter under makeshift tents or other temporary structures.

It would likely improve the accuracy of the model and decrease the frequency of false negatives and false positives to incorporate additional indicators, such as thermal imaging to detect heat signatures, and cross-referencing those with other types of spectral data, like the RGB data provided in this project.

Conclusion 2 - Importance of Comprehensive Data Collection and Labeling

The resources, data gathering, and human labeling that went into creating these training and holdout datasets are what allowed this predictive modeling exercise to take place. We make an effort to discuss in this section of the report the moral conundrum of where to direct resources and the tradeoffs involved in that decision making; however, without significant effort to first gather the data, label it, and perform predictive modeling on it, we would not even have the ability to make these decisions. To start, the team from Rochester Institute of Technology had to gain access to an aircraft, a pilot, fuel, high resolution cameras, and more to be able to do the data gathering. Then, significant effort went into labeling each photograph as containing a blue tarp or not. With these blue tarp classifications provided, the pixel variables are great quantitative predictors.

The researchers obviously thought through the problem at hand and approached it with a data-centric design. They considered the full data analysis life cycle, from data collection to providing informative predictors for modeling and ultimately identifying people in need. Overall, this kind of effort and thought into data design and its utility is key to solving many real world problems.

Conclusion 3 - Potential and Limitations of Model Application in Disaster Relief

These models have good potential for application in disaster relief. Effectively identifying the presence of blue tarps in varied terrains enables the rapid deployment of resources to affected peoples. A model that quickly identifies blue tarps from images could decrease response time dramatically. Our best model has a 33% precision. Using this model 1 in 3 visits would result in a human contact. This is likely worth it in disaster relief scenarios where survival chances are already low. It is important to recognize that some of the false positives may be adjacent or near an actual blue tarp. Resources could also be redistributed following false positives, however this may become impractical with certain foods and will extend the response time. Importantly this means that less money is going directly to disaster relief.

Conclusion 4 - Determination and Justification of which Algorithm Performs the Best

Non-parameter-tuning models:

The training data precision for logistic regression is 71.7% but is beaten by QDA with a precision of 75.4%. However, when we examine the holdout data, the logistic model's precision drops to 6.96% while QDA drops to 22.1%. LDA is similar to QDA with a holdout precision of 22.0%. The difference between training and holdout performance for the logistic model provides evidence of overfitting. Additionally, the holdout sensitivity is 99.9%. While this percentage should be high given the distribution of classes in the training and holdout datasets, to such an extent indicates that the model is classifying virtually everything as FALSE. Due to this deterioration in performance, we can exclude the logistic regression model.

The QDA model performed better in the training data on every metric compared to the LDA model. While notable, our primary criterion for selecting the best model is holdout performance, therefore we must assess as such. For disaster relief purposes, accuracy and precision are the two most important metrics. Comparing the two models with respect to accuracy and precision, LDA has an accuracy of 97.6% and a precision 22.0%, while the QDA model has an accuracy of 97.7% and precision 22.1%. Given its marginally superior performance, the QDA model is our chosen best classification algorithm for those without parameter tuning.

Parameter-tuning models:

The training data precision for the penalized logistic regression model and linear support vector machine models are both 75%; the polynomial support vector machine model and the random forest ensemble method model both narrowly beat the previous two at 77% precision; the radial basis function support vector machine model has a precision of 87% on the training data; the k-nearest neighbors model has a precision of 89% on the training data. When we examine the precision on the holdout data, the penalized logistic model dropped to 8%, the linear support vector machine model dropped to 15%, the polynomial support vector machine model dropped to 11%, the radial basis function support vector machine model dropped to 26%, the k-nearest neighbors model dropped to 33%, and the random forest ensemble method model dropped to 19%. As with the basic logistic regression, the vast difference between the training and holdout performance provides substantial evidence of overfitting, and again has a holdout sensitivity of approximately 100%. Therefore, we can again exclude the logistic regression model variant. Similarly, the linear support vector machine model, the polynomial support vector machine model, and the random forest model all had low precision on the holdout set, and performed worse than the QDA and LDA on almost all metrics, therefore we can exclude these as well.

As before, we compare the remaining two models (k-nearest neighbors, and radial basis function support vector machine) based on their performance on the holdout set. The KNN model has an accuracy of 99% and a precision of 33%, while the RBFSVM model has an accuracy of 98% and a precision of 26%. While the RBFSVM model showed considerable improvements over the non-parameter-tuning models in QDA and LDA, the KNN model improved even more and is our chosen best classification model overall.

Conclusion 5 - Why We Think KNN Works Well for this Classification Problem

Since this dataset involves spectral data from aerial imagery into categories (vegetation, soil, etc.), KNN works well because the data can be intuitively divided into distinct clusters. Each category – like blue tarps – are likely to have pixel values that are closer to each other in relative space. Additionally, KNN works well with high-dimensionality problems. Because each pixel data point has a lot of features (RGB), KNN can leverage those features to find the nearest neighbors, can identify localized patterns of values, and subsequently exploit those patterns to accurately classify different regions in the imagery. KNN is also a non-parametric algorithm, which is often advantageous when given real-world datasets where the distribution of data points do not follow a common statistical pattern. Furthermore, we think that employing some form of dimensionality reduction (e.g., PCA), and perhaps more proper preprocessing techniques, could further enhance the robustness to noise of the model. In conclusion, we are of the opinion that the KNN method was developed for problems similar to this one, and that it could be improved upon even further in this case.

Conclusion 6 - Relevance of Metrics

The metrics we considered throughout the project include J-index, TPR, FPR, accuracy, precision, and roc_auc. J-index creates a balanced metric by combining the sensitivity and specificity. We used J-index for determining what threshold to use. This allows for optimal identification of actual blue tarps while avoiding false positives. The True Positive Rate is the ratio of how many blue tarps are correctly identified to how many were identified total. Similarly, the False Positive Rate is the ratio of how many areas are marked as blue tarp incorrectly to how many areas are not blue tarps. A high FPR could lead to loss of funding and volunteers making it a critical metric. Accuracy measures the proportion of correctly identified instances out of the total instances. Since most of our data is non-blue tarp, accuracy will be heavily skewed towards that subset of data. As a result, accuracy does not reflect the performance of the models well when identifying blue tarps. Precision measures the proportion of true positives out of the predicted positives. High precision ensures that when the model flags a blue tarp, it is highly likely to be correct. This is another very important metric when evaluating identification of blue tarps. Finally, the roc_auc is the area under the curve when plotting the TPR vs the FPR. This metric is an aggregate measure of performance across all thresholds. It is useful for determining thresholds and informing general performance of model. For disaster relief based on identifying blue tarps, precision is the most useful for evaluating models. Selecting a model with high precision ensures resources are used efficiently.