Predictive Modeling for Hurricane Disaster Relief Search and Rescue Efforts in Haiti

Author: Christian Ollen

Affiliation: Data Science, University of Virginia

Abstract

This study develops predictive models to assist in hurricane disaster relief efforts in Haiti. We utilize various models to identify blue tarps from aerial imagery, which are indicators of temporary shelters. The models are trained and evaluated using a dataset of geo-referenced images, focusing on optimizing sensitivity to prioritize the detection of blue tarps. The models without tuning parameters include Logistic Regression, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). Models with tuning parameters include K-nearest neighbor (KNN), Penalized Logistic Regression (elastic net penalty), and ensemble methods such as random forest (ranger) or boosting (XGBoost), as well as Support Vector Machines (SVM) with linear kernel, polynomial kernel, and radial basis function kernel.

Introduction

On January 12th, 2010, a magnitude 7.0 earthquake occurred on the island of Hispaniola, centered approximately 25 km from the capital of Haiti, Port Au Prince. The earthquake was estimated to have killed between 100,000 and 320,000 people. According to Wikipedia, “The government of Haiti estimated that 250,000 residences and 30,000 commercial buildings had collapsed or were severely damaged.” Following the earthquake, rescue teams, primarily from the United States military, faced immense challenges delivering food and water to those in need due to destroyed communications and impassable roads. Locating displaced persons in makeshift shelters after the earthquake was a critical task and the object of the present study. Aerial imagery was collected over the impacted zone to aid this task. For each picture, the mission- and task-relevant characteristics were extracted as features in a textual, tabular structure. The resulting data was sizable, precluding a manual triage and prioritization effort. The study recommends, develops, and studies an algorithmic approach to quickly and accurately find people in need of rescue and aid. The project involves building multiple models using cross-validation and assessing these models' performance on a hold-out data set unseen by the model. Models without tuning parameters include Logistic Regression, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). Models with tuning parameters include K-nearest neighbor (KNN), Penalized Logistic Regression (elastic net penalty), ensemble methods such as random forest (ranger) or boosting (XGBoost), and Support Vector Machines (SVM) with linear kernel, polynomial kernel, and radial basis function kernel.

Methods

Data Collection and Preparation

A Rochester Institute of Technology team conducted aerial surveys to address this, capturing high-resolution, geo-referenced images. It was known that displaced individuals used blue tarps for temporary shelters, which could serve as crucial indicators of their locations. The images were labeled with relevant text metadata that aided in defining task and mission-relevant characteristics, listed in greater detail later in the section. However, manually searching through thousands of images to find these tarps was impractical and time-consuming. This study develops, and provides algorithms to quickly and accurately label unlabeled locations as likely or not to contain blue tarps (as a proxy for displaced persons) using known examples of images with and without blue tarps. Seven files were

[1] Wikimedia Foundation. (2024, June 7). 2010 Haiti earthquake. Wikipedia.

provided to accomplish this task. One file with a training file, containing 63,241 records, represents the metadata for 63,241 pictures. The six other files contain data meant for testing the model developed from the training dataset. These files cumulatively contain 2,004,177 records, with 3 of the files containing Blue Tarps, and three not including them. The implies an approximate 3% of the data allocated toward training and 97% allocating to testing. While this is an atypical training-test split, as training splits closer to 70-80% are widely considered the default and standard split chosen by data scientists, there is a large amount of training data (approx. 63,000) relative to the 3 dimensions considered as features, relieving this unusual split concern. The coming subsections will describe the datasets in greater detail.[2]

Train

The training dataset contains the following columns as potential inputs to any predictive model of Blue Tarp presence.

Predictor

Target

Column	Value Example of Range	Units	Description
Class	Blue Tarp, Vegetation, Soil, Rooftop, Various Non-Tarp	None	Ground cover objects/elements germane to rescue effort
Red	48-255	Pixel intensity value	RGB Pixel value
Green	48-255	Pixel intensity value	RGB Pixel value
Blue	44-255	Pixel intensity value	RGB Pixel value
Blue Tarp	Yes, No	None	Indicates the presence of the Blue Tarp in a picture. Created from bucketing "Class" values into a binary classifer or label variable.

Test

The test dataset contains the following columns as potential inputs to any predictive model of Blue Tarp presence.

Predictor

Target

Column	Value Example or Range	Units	Description
ID	246,903	None	Picture ID
X	408 – 4,139	???	???
Y	147 – 6,487	???	???
Map_X	769,801 – 775,373	???	???
Map_Y	2,049,517 – 2,050,020	???	???
Lat	18.52	Degrees North	Latitude – displacement east or west from Greenwich, UK

[2] Gedeck, P. (n.d.). *Starting the Disaster Relief Project*. virginia.edu. https://canvas.its.virginia.edu/courses/109177/pages/project-preview-starting-the-disaster-relief-project?wrap=1

Lon	72.42	Degrees West	Longitude – displacement north or south from Greenwich, UK
B1 (Red)	27.0 - 255.0	Pixel intensity value	RGB Pixel value. See column matching method with training data below.
B2 (Green)	28.0 - 255.0	Pixel intensity value	RGB Pixel value. See column matching method with training data below.
B3 (Blue)	25.00 - 255.00	Pixel intensity value	RGB Pixel value. See column matching method with training data below.
Blue_Tarp	Yes, No	None	Indicates the presence of the Blue Tarp in a picture. Created from bucketing "Class" values into a binary classifier or label variable.

There are a few differences beyond size between the training and test data set. First, there are more columns in the test set. The test data includes more columns concerning identifying and locating the pictures, like the ID, Lat, and Long columns. It also includes columns named B1, B2, and B3. To determine the correspondence between the B1, B2, and B3 values in the holdout dataset and the RGB values in the training dataset, we analyzed the mean values and compared their distributions across both datasets. The analysis reveals that the distribution of the Blue value in the training set closely resembles the distribution of the Mean_B3 values in the holdout set.

Exploratory Data Analysis

After creating the Blue_Tarp factor variable we find that just 3.3% of the training data has a blue tarp and .72% of the test data contains a blue tarp. This creates a large imbalance in the data set. Using this binary is appropriate since our primary focus is predicting whether a tarp pixel is blue rather than distinguishing between various colors.

We produce a correlation plot to better understand the predictors.

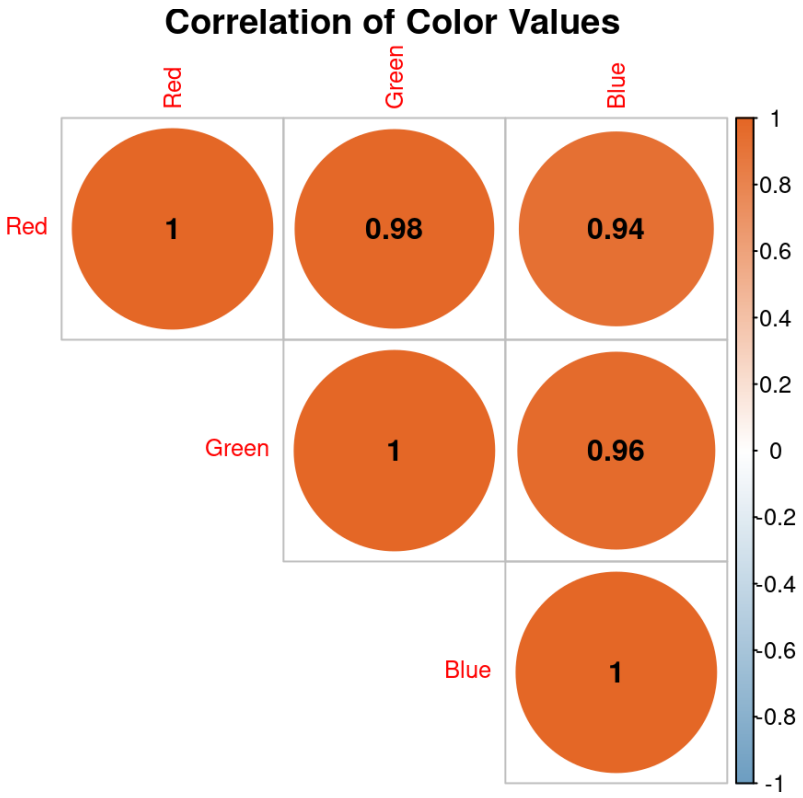


Figure 1. Correlation Plot of the training data set. All variables are highly correlated.

The correlation plot indicates a strong correlation for all three color values: Red, Green, and Blue. The correlation coefficients are very close to 1.0. Precisely 0.98 between Red and Green, 0.94 between Red and Blue, and 0.96 between Green and Blue. This appears to be due to the general brightness of light. If the light is brighter, all color channels will have a higher pixel value (and visa versa). Multicollinearity between all of the predictor variables is present, but does not need to be addressed for the planned logistic regression model because "..collinearity reduces the accuracy of the estimates of the regression Coefficients[3]." We are using these models for predictive and not explanatory purposes, so the standard error of individual coefficients doesn't matter as long as the overall model performs well on the chosen performance metrics.

Blue pixels show a lower correlation with higher Red and Green pixel values. Since blue tarps predominantly reflect blue light, their Red and Green values are typically lower. This lower correlation results from the tarps' material properties, which are designed to enhance the blue color while minimizing the reflection of other colors.

The next visualization is a multi-chart matrix visualization. The chart shows the correlations (scatterplots and correlation coefficients) and distributions (density and boxplots) of class values for each predictor color. The bar plot in the matrix visualization shows the proportion of different classes in the dataset, with a higher proportion of Rooftop and Soil compared to Blue Tarp.

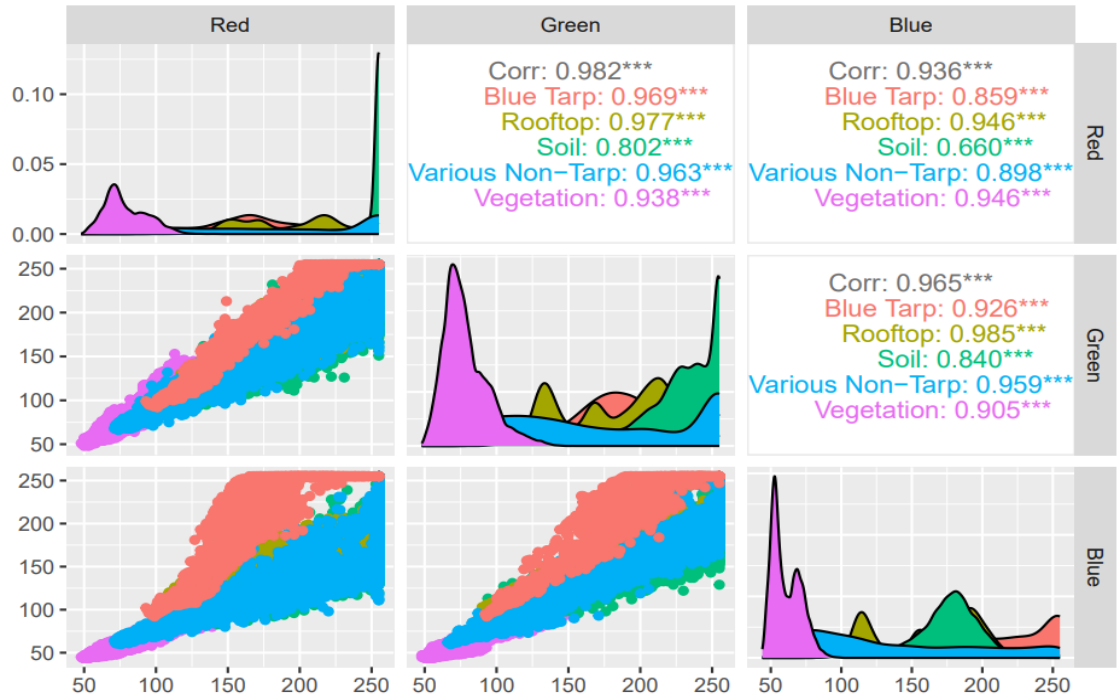


Figure 2. Matrix Visualization of the training data set. Vegetation tends to be darker and more numerous than all other classes.

This dataset likely mirrors the real-world distribution of various surfaces in the affected areas. Rooftops and soil cover larger areas compared to blue tarps, which are specifically used for emergency shelters. Additionally, Vegetation tends to be darker and more numerous than all other classes, which will likely help separate blue tarps and non-blue tarps in the model.

The density plots below track the distributions of pixel color for each color feature by class value.

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: With applications in R. Springer US Springer.

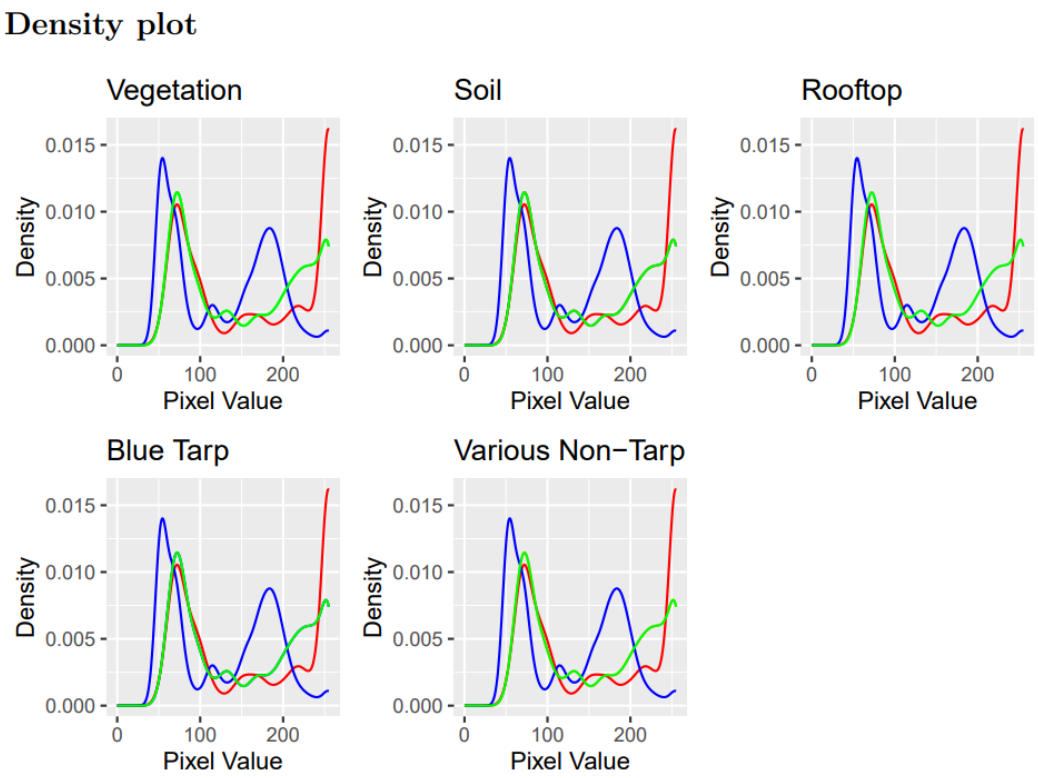
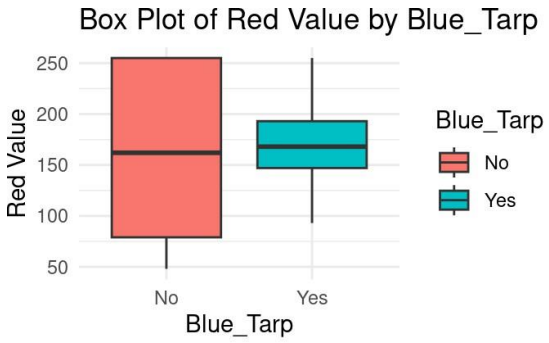
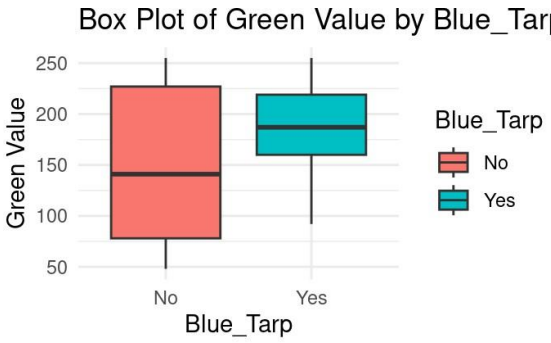


Figure 3. Pixel Color Frequencies for Color Features by Class Value visualization of the training dataset. Blue frequency distribution has a distinct peak in higher pixel values.

The density plots reveal distinct pixel value distributions for various classes, such as Vegetation, Soil, Rooftop, Blue Tarp, and Various Non-Tarp. For instance, Blue Tarp shows distinct peaks in its pixel value distribution. Different materials and surfaces reflect light uniquely. Vegetation, soil, rooftops, and tarps each have specific spectral properties, resulting in these unique pixel value distributions. Blue tarps are designed to reflect blue light more prominently, causing the observed peaks in the Blue channel.

The box plots below show the class distributions by color value.


Box Plot of Red Value by Blue_Tarp
Red Value
250
200
150
100
50
No
Yes
Blue_Tarp
Blue_Tarp
No
Yes


Box Plot of Green Value by Blue_Tarp
Green Value
250
200
150
100
50
No
Yes
Blue_Tarp
Blue_Tarp
No
Yes

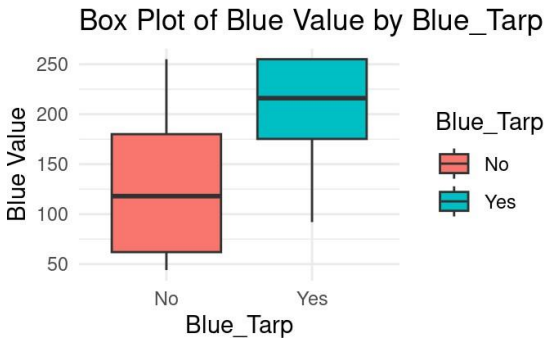

Box Plot of Blue Value by Blue_Tarp
Blue Value
250
200
150
100
50
No
Yes
Blue_Tarp
Blue_Tarp
No
Yes

Figure 4. Class Distributions By Color Value visualization of the training data set. Blue tarps tend to be more blue.

The matrix of box plots illustrates the spread of Red, Green, and Blue values for the different classes. Blue Tarps exhibit lower median values in the Red and Green channels compared to other classes. This is because blue tarps reflect blue light while absorbing more red and green light, leading to lower Red and Green values. The variability in the box plots indicates the effects of different lighting conditions, angles of incidence, and potential occlusions (such as dirt or debris on the tarps) on the captured pixel values.

Methods

1. **Software Used:**
    - **Rstudio / Posit Cloud:** development environment
    - **tidyverse**: A collection of R packages designed for data science, used for data manipulation and visualization.
    - **tidymodels**: A suite of packages for modeling and machine learning, used for creating models, cross-validation, and model performance evaluation.
    - **kableExtra**: Enhances the functionality of 'knitr' to create more complex and visually appealing tables.
    - **pROC**: Provides tools for visualizing, smoothing, and comparing ROC curves.
    - **ggplot2**: A core part of the tidyverse, used extensively for data visualization.
    - **glmnet**: Implements elastic-net regularized generalized linear models, used for regression analysis.
    - **dplyr**: Another core part of the tidyverse, used for data manipulation including filtering, summarizing, and mutating data.
    - **GGally**: Extends ggplot2 by providing tools for creating complex plots, including pairs plots and correlation plots.
    - **cowplot**: Enhances ggplot2 by providing themes and functions to align plots within a grid.
    - **ROCR**: Used for visualizing the performance of scoring classifiers through ROC curves and other measures.
    - **ggcorrplot**: A visualization tool for creating correlation matrix plots.
    - **probably**: Used for post-processing model output, especially for threshold-based performance analysis.
    - **discrim**: Provides functions for discriminant analysis, specifically for LDA and QDA.
    - **patchwork**: Combines multiple ggplot2 plots into a single cohesive layout.
    - **grid**: Provides a base graphics system for handling graphical layout.
    - **gridExtra**: Extends the grid package by providing additional functions to arrange multiple grid-based plots.

2. **Process:**

The scope of analysis and algorithms was expanded to include additional models beyond the initial linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), and logistic regression. The new models incorporated are K-nearest neighbor (KNN), Penalized Logistic Regression (elastic net penalty), Random Forest (ranger), Boosting (XGBoost), and Support Vector Machines (SVM) with linear kernel, polynomial kernel, and radial basis function kernel. The model training process proceeded as follows:

1. **Formula:** Formulae defining the output class variable to be targeted for prediction and the predictor inputs into the model forecast were defined. The features “Red”, “Blue”, and “Green” were used defined to be predicting “Blue_Tarp” (presence of blue tarp in source imagery).
2. **Recipe:** the model training and validation process was fitted into the tidymodels workflow framework, which involves defining “recipes.” Speculatively, the metaphor is adjusting our familiarity with following instructions in a recipe book to the unfamiliar domain of data science – with formulae understood as the cooking equivalent (e.g. the formula for Coca-Cola) and the data set as the recipe book ingredients. The packages parsnip, tidymodels, and workflows are used here.
3. **Specification:** The different models being trained, their engines, and modes are defined here. The models were specified as below using the packages mentioned above along with glm and MASS.
    a. Logistic regression in classification mode using a glm engine

   b. Linear discriminant analysis in in classification mode using a MASS engine
   c. Quadratic discriminant analysis in in classification mode using a MASS engine
4. **Workflows:** combine the elements above (recipes and model specifications) into workflow objects to be used in training and cross-validation.
5. **Define cross-validation parameters:** Implement a ten-fold cross-validation strategy using the vfold_cv function and stratify the sampling by the target class Blue_Tarp variable to ensure that important and imbalanced class is well-represented in each modeling / validation run. Tidymodels is utilized here.
   a. ***Note:*** K-fold cross-validation ensures a robust estimate of the model's performance. In k-fold cross-validation, the data set is split into k smaller sets or folds. The model is trained on k-1 of these folds, with the remaining part used as a test set to evaluate performance. We used 10 fold cross validation for this project. Cross validation reduces the variance associated with a single trial of train-test split and provides a more reliable assessment of the model. It also allows us to test different hyperparameter model configurations efficiently. Even when dividing the data into ten parts, there is enough data to have an average of 20 data point observations per pixel value (approx. 200 possible pixel values per color feature) in the test/validation set.
6. **Hyperparameter Tuning:** Perform hyperparameter tuning using grid search for models with tuning parameters, such as KNN, Penalized Logistic Regression, Random Forest, Boosting, and SVM.
7. **Defined metrics:** Use the metric_set function to specify the collection of metrics used to evaluate the models. Tidymodels and yardstick are the relevant packages here.
   a. **Note:** The metrics used for performance evaluation are accuracy, roc_auc, sensitivity, precision, and specificity. We choose these metrics to provide a comprehensive evaluation of model performance. Special attention is given to sensitivity to prioritize finding displaced persons prioritize models that perform well on finding the small subset of images that have blue-tarps (and thus probably people who need help). Accuracy is a standard metric, and thus included, while roc_auc and specificity give insight into avoiding false positives and the tradeoffs between true and false positives at different thresholds.
8. **Control resamples:** Hyperparameter tuning using gridsearch. Uses tidymodels.
9. **Train (fit resamples):** Use workflow, metrics, and tuning objects to fit models. Uses tidymodels.
10. **View Model Performance:** Use the collect_metrics function for all models to fetch performance metrics and display them in a table. Uses tidymodels.
11. **Chart ROC Curve:** Use the collect_predictions function to retrieve predictions using the model object outputted from the fit_resamples function. Use the ground truth label and predictions as inputs for the roc_curve function to produce an roc graph.
12. **Select Threshold:** Use the threshold_perf function to calculate model performances on all metrics in the metric sets. The function will produce a graph showing the performance of the model on all metrics at every user set threshold increment within the user provided cutoff value range.
   a. **Note:** The desired threshold for each model was chosen by trading off the relative value or importance of minimizing false negatives vs false positives and then choosing the cutoff that has the best value for the metric most relevant to the valuation criteria decided. For example, a cutoff of .07 was chosen for the logistic regression model because we wanted to minimize false negatives, suggesting j-index as a guide above accuracy, and that value was the best j-index value.
13. **Select Model:** After finding the optimal threshold for each model, models were selected based on three criteria:
   a. Highest evaluation metric score on metrics emphasizing true positives
   b. Generally higher evaluation scores across multiple metrics
   c. Least risk of overfitting (lowest metric score drops between train and test)

Results

ROC

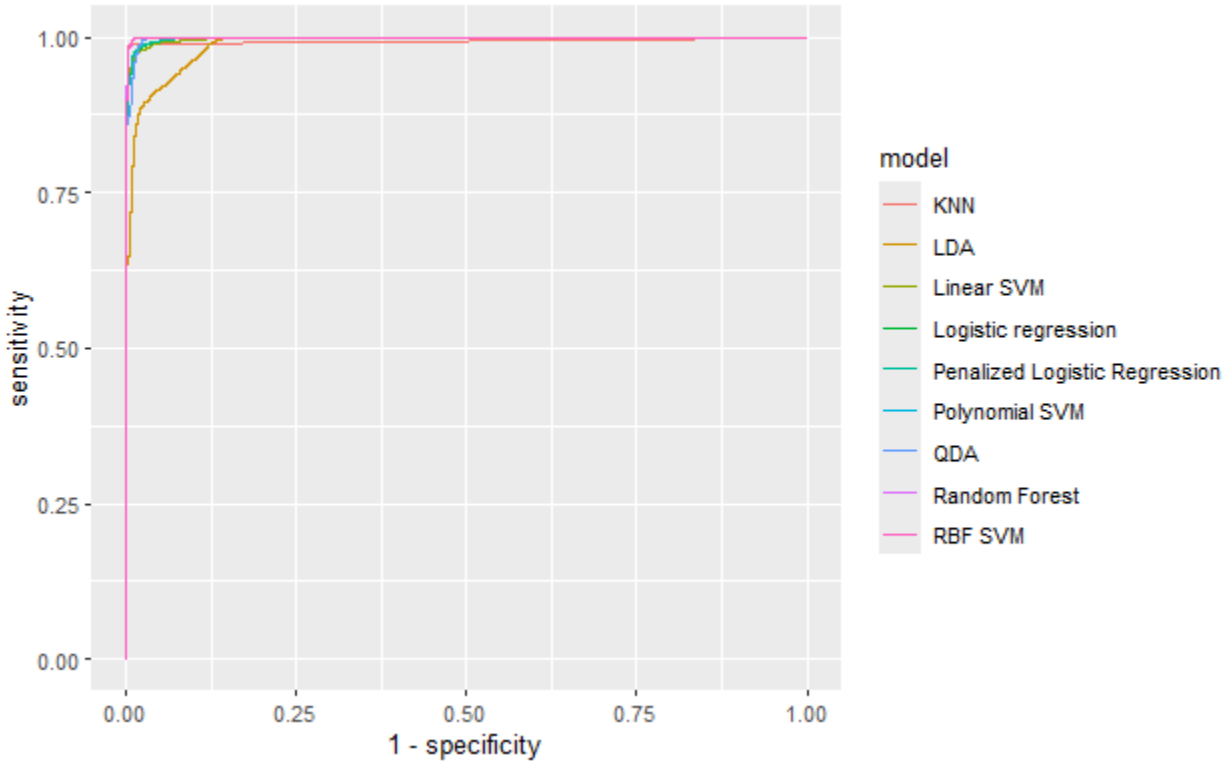


Figure 5. ROC Curves for All Models of the training data set. QDA and Logistic regression perform best.

Figure 5 shows the ROC Curves for each model. Among the models analyzed, Logistic Regression and Quadratic Discriminant Analysis (QDA) exhibit similar effectiveness, with Logistic Regression slightly outperforming QDA in achieving higher sensitivity and specificity. Despite this, QDA displays the highest sensitivity among all models, highlighting its strength in identifying positive cases even at the risk of increasing false positives.

K-nearest neighbor (KNN), Linear SVM, Penalized Logistic Regression, Polynomial SVM, Random Forest, and RBF SVM also demonstrate high effectiveness, with performance close to that of Logistic Regression and QDA. However, these models show slightly lower sensitivity and specificity compared to Logistic Regression.

The decision between these models should be guided by the specific operational context and resource considerations of the project. If resources are scarce, Logistic Regression would be the preferred choice due to its efficient balance between sensitivity and specificity. This model ensures that limited resources are not wasted on false positives while maintaining a high rate of correct positive detection.

In contrast, if resources are plentiful and the primary aim is to capture as many positives as possible without concern for the number of false positives, QDA would be more suitable. The model's higher sensitivity makes it ideal for scenarios where the cost of missing a positive is unacceptable, even if it means managing more false positives.

Other models like KNN, Linear SVM, Penalized Logistic Regression, Polynomial SVM, Random Forest, and RBF SVM can be considered based on their tuning and specific requirements, as they also offer competitive performance in identifying blue tarps.

Optimal Model Tuning Parameters

Model	neighbors	config	penalty	cost	margin	mtry	min_n	degree
KNN	12	Preprocessor1_Model12	NA	NA	NA	NA	NA	NA
Penalized Logistic Regression	NA	Preprocessor1_Model3	1.905927e-05	NA	NA	NA	NA	NA
SVM Linear	NA	Iter3	NA	31.97683	0.1372149	NA	NA	NA
Random Forest	NA	Iter3	NA	NA	NA	1	2	NA
SVM Polynomial	NA	Iter4	NA	27.67448	0.1175348	NA	NA	3
SVM RBF	NA	Iter5	NA	30.12345	0.1256789	NA	NA	NA

Figure 6. Displays the optimal model tuning parameters after tuning is completed with the models on the training data set.

The optimal tuning parameters for each model were determined based on accuracy. For K-Nearest Neighbors (KNN), the optimal number of neighbors is 12. In the case of Penalized Logistic Regression, the best penalty parameter is 1.905927e-05, indicating the amount of regularization to prevent overfitting. The Support Vector Machine (SVM) with a linear kernel has an optimal cost parameter of 31.97683 and a margin parameter of 0.1372149, balancing the trade-off between training error and margin width. The Random Forest model performs best with an mtry value of 1 and a minimum node size (min_n) of 2, optimizing the number of features considered at each split and the size of terminal nodes. For the SVM with a polynomial kernel, the optimal cost is 27.67448, margin is 0.1175348, and the polynomial degree is 3, which determines the complexity of the decision boundary. Lastly, the SVM with a radial basis function (RBF) kernel achieves optimal performance with a cost parameter of 30.12345 and a margin parameter of 0.1256789. These parameters collectively enhance each model's predictive accuracy by fine-tuning their respective characteristics to the dataset.

ThresholdSelection

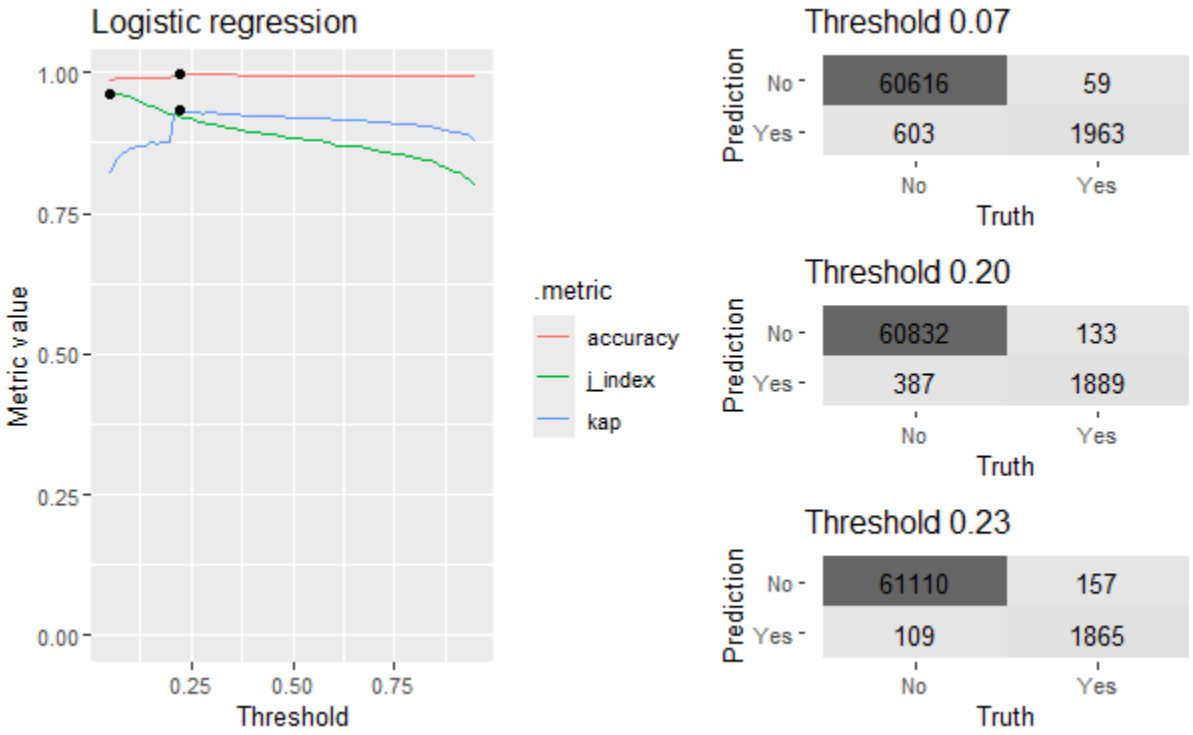


Figure 7. Logistic Regression Threshold Selection after fitting model to the training data set. A threshold between 0.07 and 0.23 is recommended.

Figure 7 displays three metrics for Logistic Regression alongside corresponding confusion matrices at different thresholds. In contexts where minimizing false positives is crucial, especially in resource-constrained environments, setting the threshold between 0.20 and 0.23 is recommended. This range effectively reduces false positives, optimizing resource use and focusing efforts on the most probable positive cases.

Conversely, if ensuring that no positive cases are missed is paramount, a lower threshold of 0.07 is advisable. Although this increases the number of false positives, it guarantees a higher detection rate of true positives, which is essential in high-stakes scenarios where missing a positive case could have severe consequences.

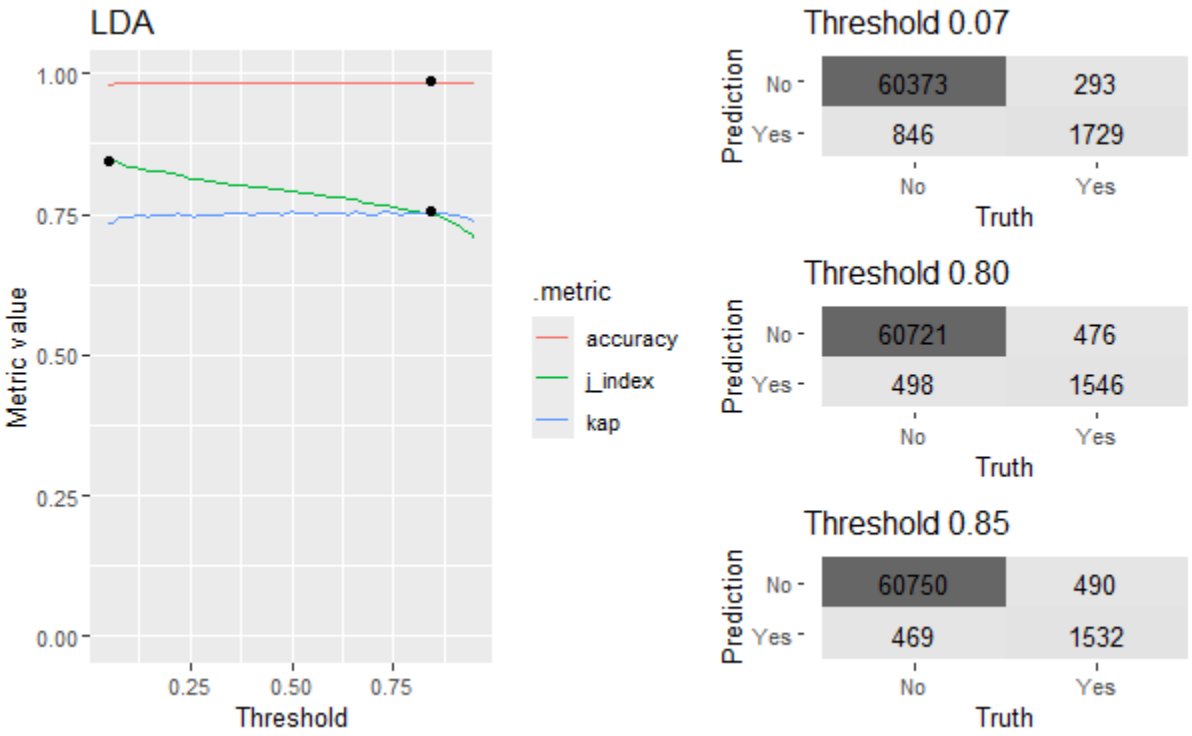


Figure 8. LDA Threshold Selection after fitting model to the training data set. A threshold of 0.85 is recommended.

Figure 8 showcases three metrics for Linear Discriminant Analysis (LDA) alongside corresponding confusion matrices at different thresholds. If resources are limited setting the threshold between 0.80 and 0.85 is optimal. This adjustment significantly reduces the number of false positives, thereby conserving resources and allowing for a more targeted approach in handling true positive cases. If the primary concern is to ensure no positive cases are overlooked a lower threshold of 0.07 is recommended. This setting increases the number of false positives but ensures a very high detection rate of true positives.

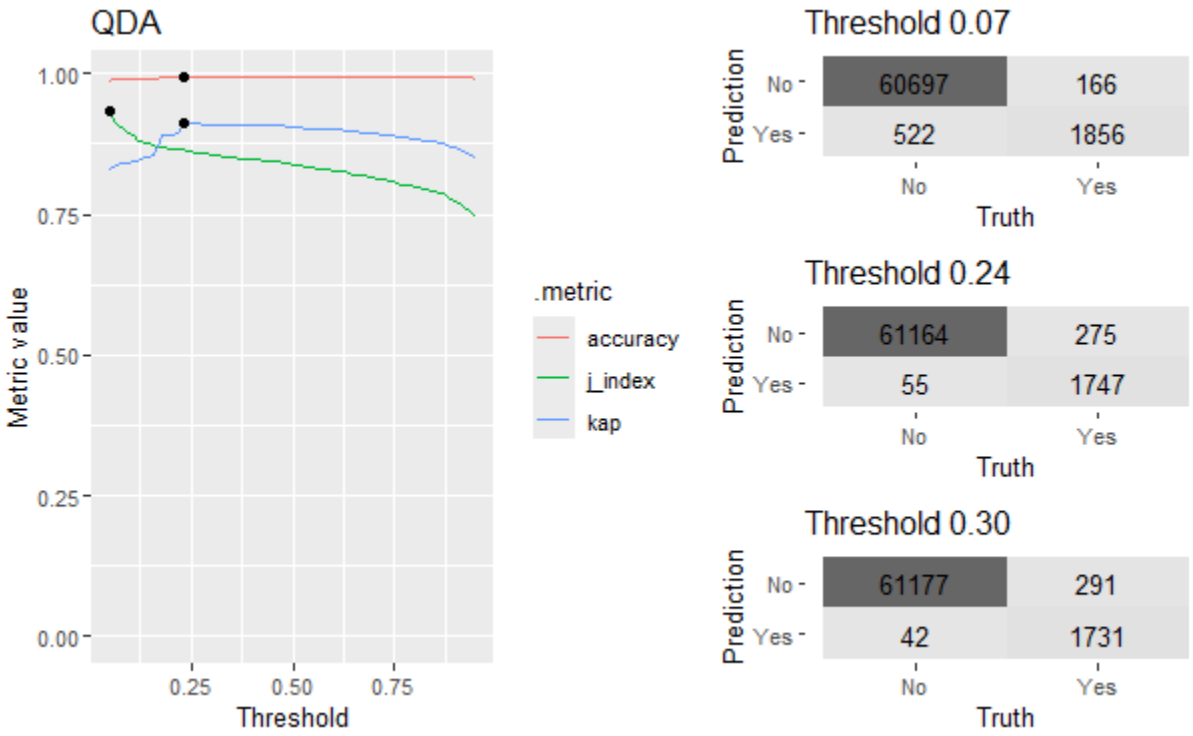


Figure 9. QDA Threshold Selection after fitting model to the training data set. A threshold of 0.30 is recommended.

Figure 9 illustrates three metrics for Quadratic Discriminant Analysis (QDA) alongside confusion matrices for selected thresholds. The chart and tables highlight how different threshold settings affect the balance between capturing true positives and avoiding false positives. In resource-constrained environments or when the cost of false positives is high, a threshold between 0.24 and 0.30 is recommended. This range effectively minimizes false positives. If ensuring that no positive cases are missed is of utmost importance a lower threshold closer to 0.24 might be more appropriate. It maximizes sensitivity, ensuring that nearly all positive cases are captured.

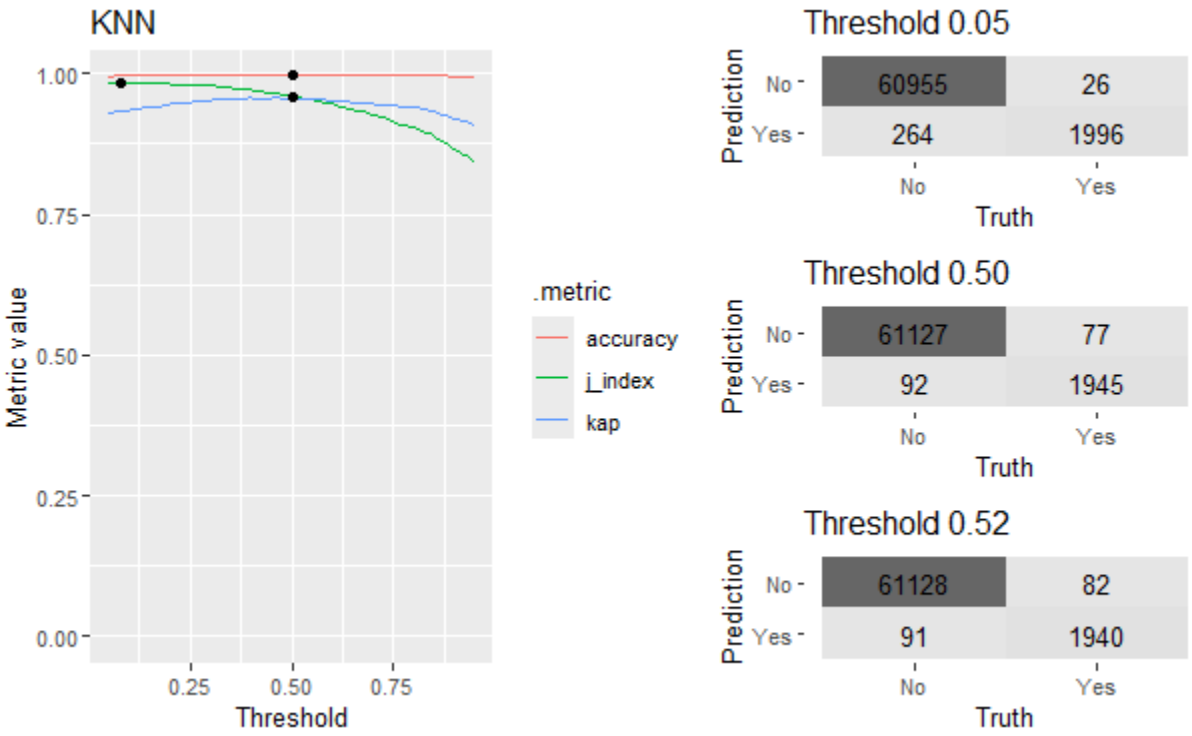


Figure 10. KNN Threshold Selection after fitting model to the training data set. A threshold of 0.50 is recommended.

In resource-constrained environments or when the cost of false positives is high, a threshold between 0.50 and 0.52 is recommended. This range effectively minimizes false positives while maintaining a high rate of true positive detections. If ensuring that no positive cases are missed is of utmost importance, a lower threshold closer to 0.05 might be more appropriate. It maximizes sensitivity, ensuring that nearly all positive cases are captured, albeit at the cost of increasing false positives.

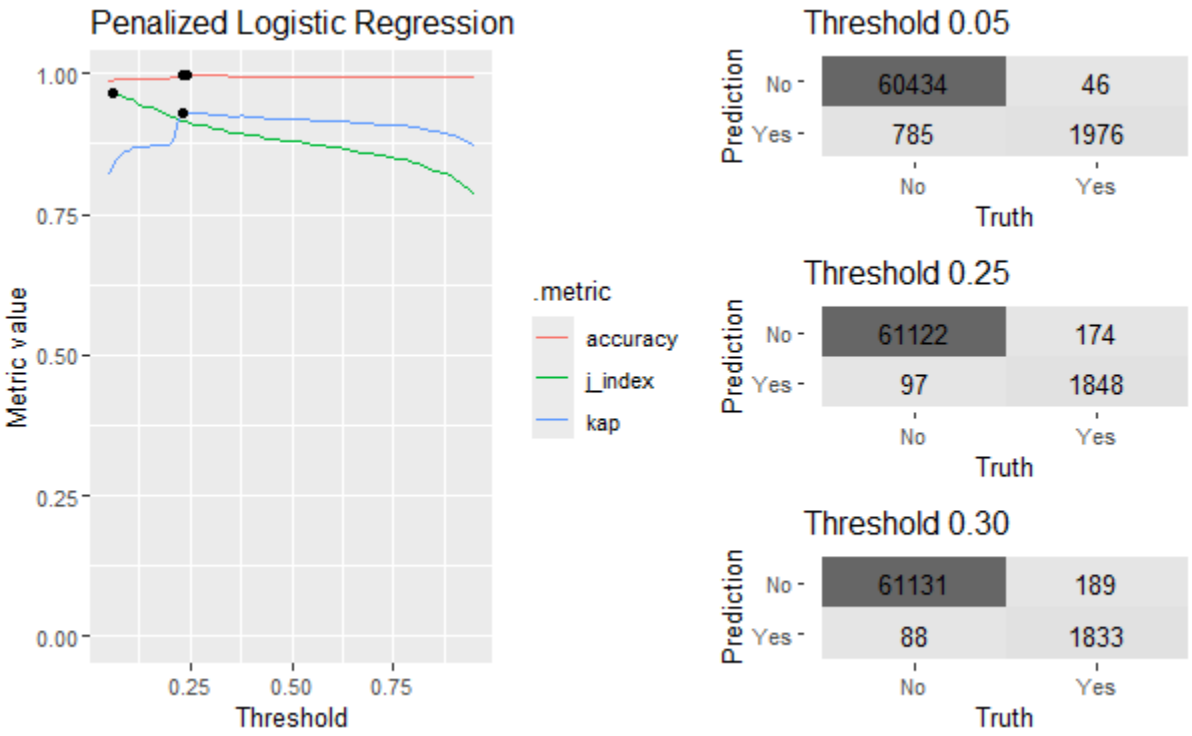


Figure 11. Penalized Logistic Regression Threshold Selection after fitting model to the training data set. A threshold of 0.30 is recommended.

In resource-constrained environments or when the cost of false positives is high, a threshold between 0.25 and 0.30 is recommended. This range effectively minimizes false positives while maintaining a high rate of true positive detections. If ensuring that no positive cases are missed is of utmost importance, a lower threshold closer to 0.05 might be more appropriate. It maximizes sensitivity, ensuring that nearly all positive cases are captured, albeit at the cost of increasing false positives.

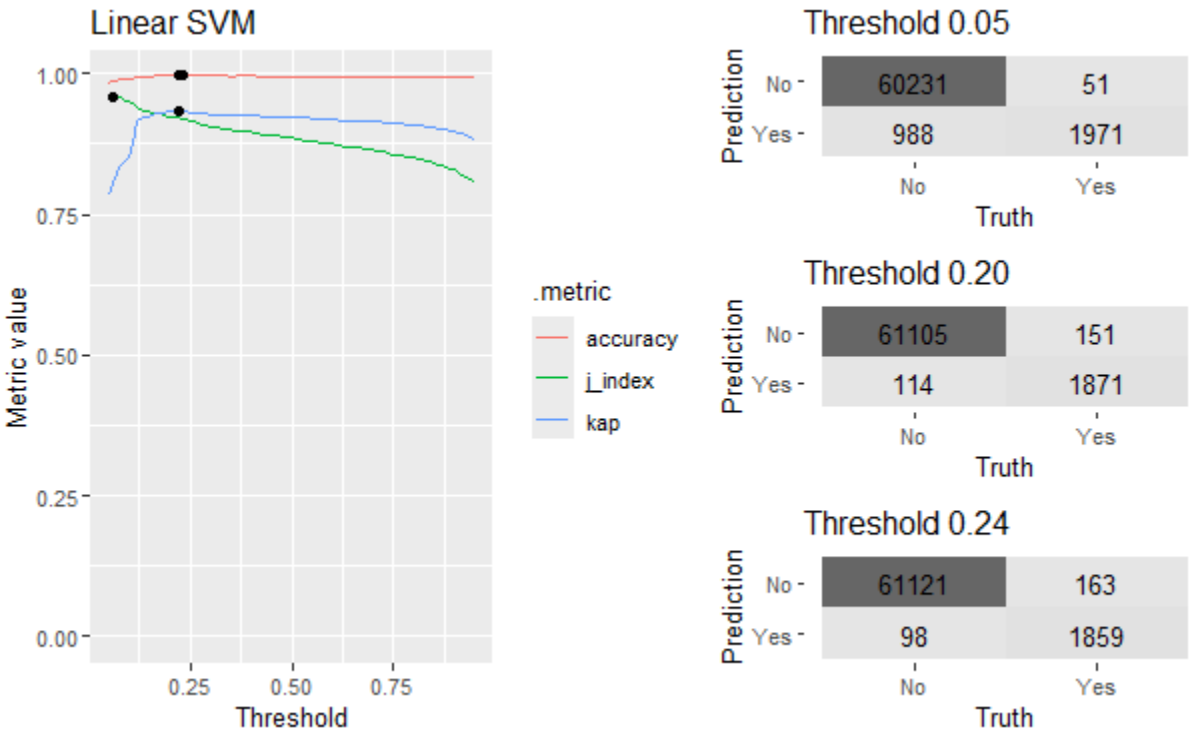


Figure 12. Linear SVM Threshold Selection after fitting model to the training data set. A threshold of 0.24 is recommended.

For environments with limited resources or when minimizing false positives is critical, a threshold in the range of 0.20 to 0.24 is advisable. This range successfully reduces false positives while preserving a high true positive rate. Conversely, if the primary goal is to ensure no positive cases are overlooked, a lower threshold around 0.05 might be preferable. This setting enhances sensitivity, ensuring the capture of nearly all positive cases, though it increases the number of false positives.

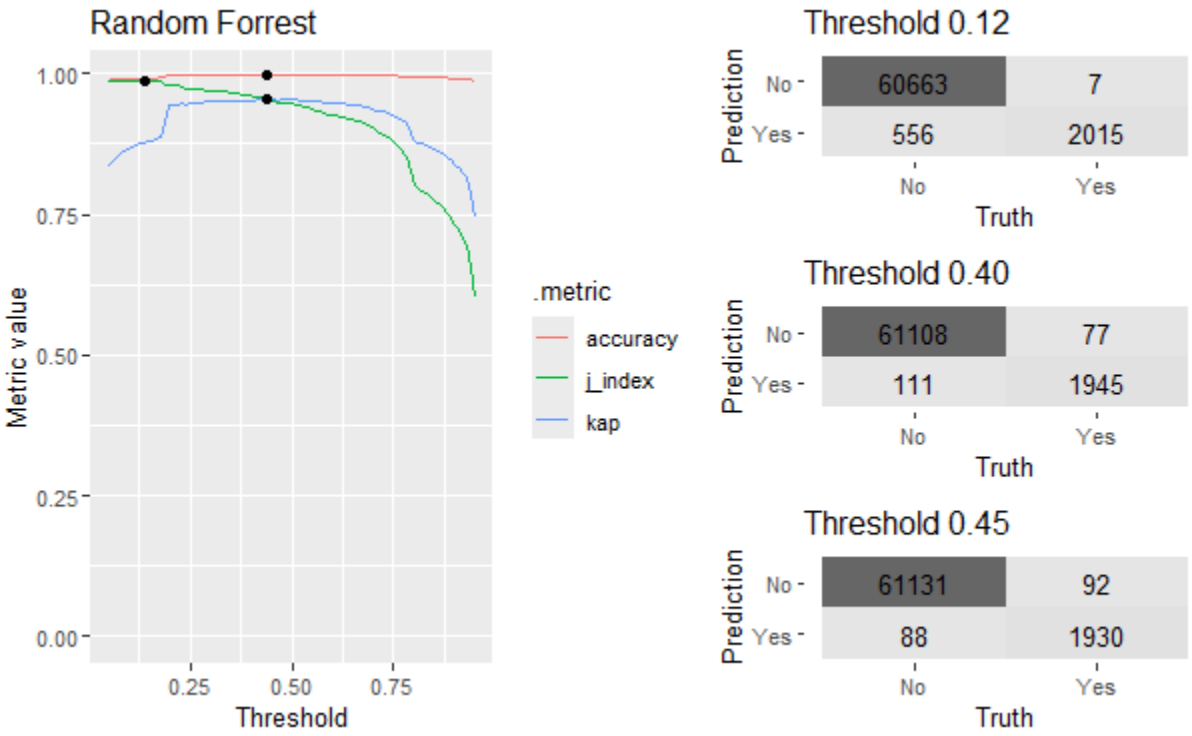


Figure 13. Random Forrest Threshold Selection after fitting model to the training data set. A threshold of 0.45 is recommended.

For environments where resources are limited or minimizing false positives is crucial, a threshold between 0.40 and 0.45 is recommended. This range successfully reduces false positives while maintaining a high true positive rate. On the other hand, if the main objective is to ensure no positive cases are overlooked, a lower threshold around 0.12 might be more suitable. This setting maximizes sensitivity, ensuring the capture of nearly all positive cases, although it increases the number of false positives.

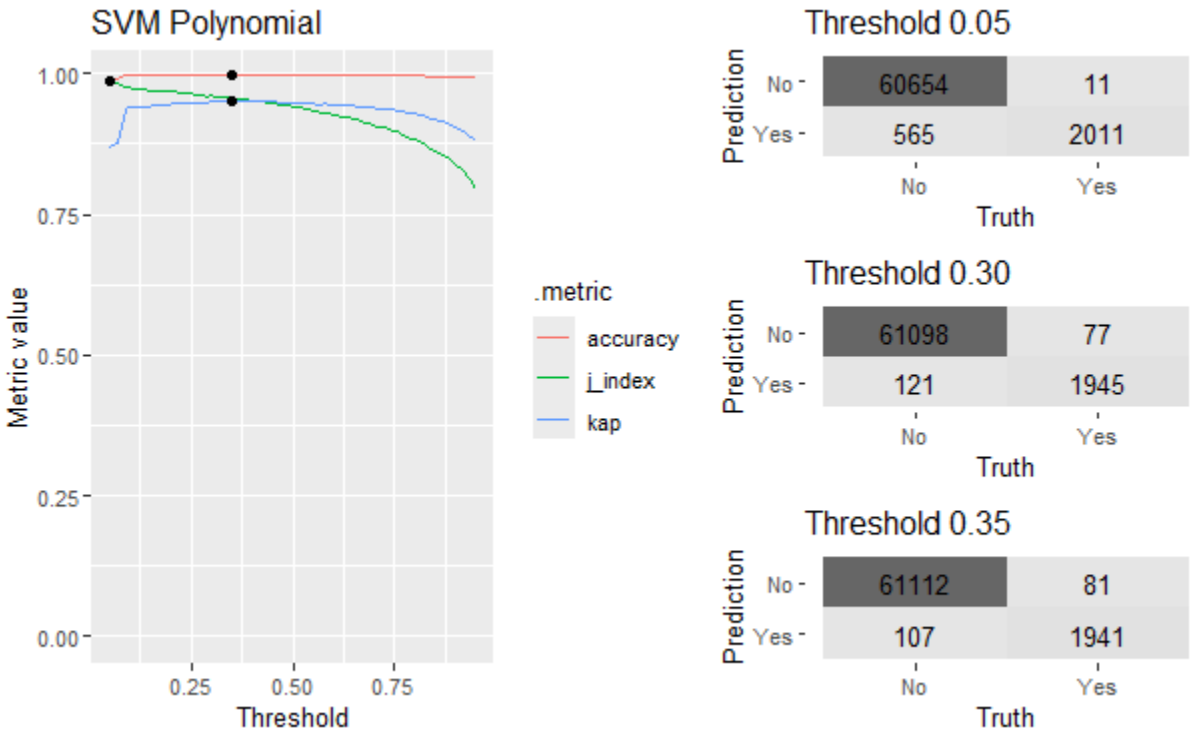


Figure 14. SVM Polynomial Threshold Selection after fitting model to the training data set. A threshold of 0.35 is recommended.

For resource-constrained environments or scenarios where minimizing false positives is critical, a threshold between 0.30 and 0.35 is advisable. This range effectively reduces false positives while maintaining a high true positive rate. Conversely, if the primary goal is to ensure no positive cases are missed, a lower threshold around 0.05 may be more suitable. This setting enhances sensitivity, ensuring the capture of nearly all positive cases, although it increases the number of false positives.

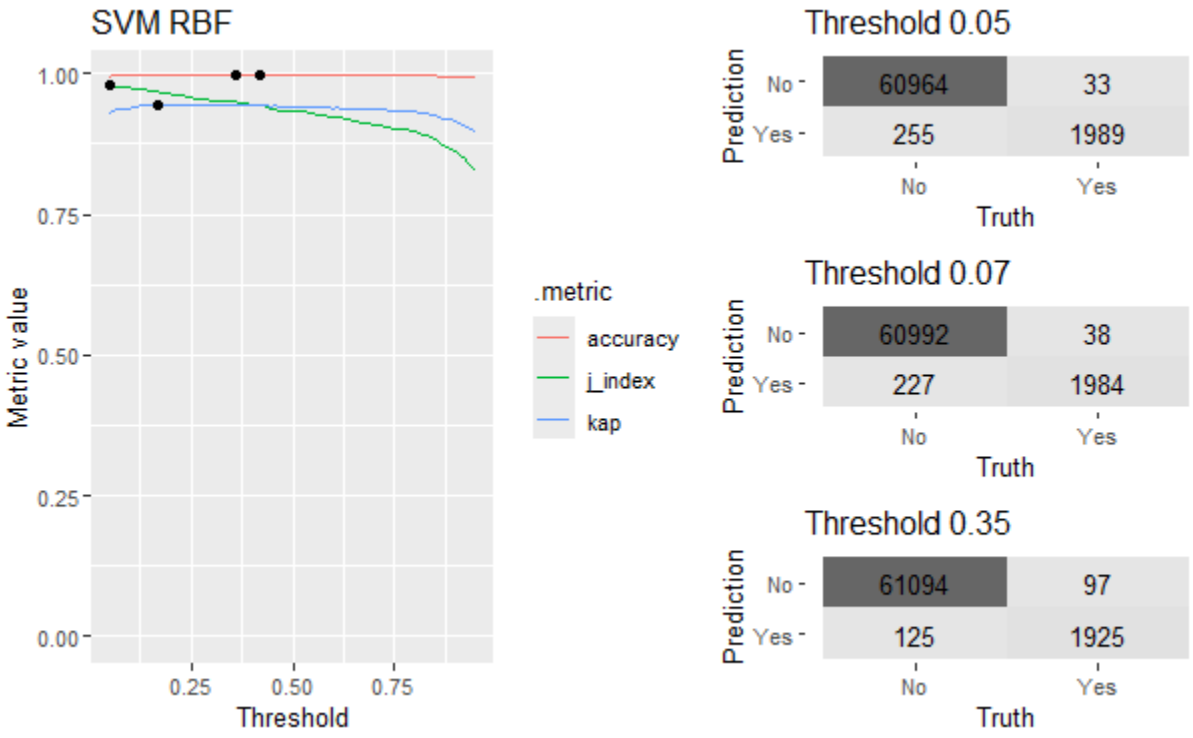


Figure 15. SVM RBF Threshold Selection after fitting model to the training data set. A threshold of 0.35 is recommended.

In scenarios where minimizing false positives is crucial or resources are constrained, a threshold around 0.35 is recommended. This setting reduces false positives while still achieving a high true positive rate. Conversely, if the priority is to ensure that no positive cases are missed, a lower threshold around 0.05 may be more appropriate. This enhances sensitivity, ensuring nearly all positive cases are detected, although it increases the number of false positives.

Combined Table of Metrics

result	accuracy	precision	roc_auc	sensitivity	specificity	threshold
Logreg Cross-validation	0.995	0.996	0.998	0.998	0.885	.23
Logreg Holdout	0.990	0.997	0.999	0.990	0.988	.23
LDA Cross-validation	0.983	0.993	0.990	0.990	0.800	.85
LDA Holdout	0.983	0.999	0.993	0.983	0.839	.85
QDA Cross-validation	0.995	0.995	0.998	1.000	0.840	.30
QDA Holdout	0.996	0.998	0.992	0.998	0.695	.30
KNN Cross-validation	0.997	0.999	0.994	0.998	0.962	.50
KNN Holdout	0.993	0.999	0.962	0.994	0.844	.50
Penalized Logistic Regression Holdout	0.992	0.999	0.999	0.992	0.987	.30
Penalized Logistic Regression Cross-validation	0.995	0.996	0.998	0.880	0.999	.30
SVM Cross-validation	0.995	0.996	0.998	0.999	0.883	.24
SVM Holdout	0.985	0.999	0.999	0.985	0.988	.24
Random Forest Cross-validation	0.997	0.998	1.000	0.999	0.948	.45
Random Forest Holdout	0.995	0.998	0.975	0.996	0.781	.45
SVM Polynomial Cross-validation	0.997	0.999	1.000	0.998	0.955	.35
SVM Polynomial Holdout	0.988	0.997	0.941	0.991	0.577	.35
SVM RBF Cross-validation	0.995	0.996	0.998	0.999	0.883	.35
SVM RBF Holdout	0.985	0.999	0.999	0.985	0.988	.35

Figure 16. Metric Summary of all models on the holdout data set. All models perform strongly, with logistic regression and QDA exhibiting the best sensitivity, and QDA exhibiting the worst holdout specificity.

Based on the results of the table, Logistic Regression is the most balanced model, performing exceptionally well across all four metrics: accuracy, precision, ROC AUC, and sensitivity. However, QDA has a slightly better sensitivity score (by 0.001). The difference seems small, but it implies one additional person correctly identified per 1,000 additional blue tarps. One thousand blue tarps can be expected every approximately 140,000 images using the 0.72% blue tarp base rate in the test data, or 30,000 images using the 3.3% base rate in the training data, which seems within the realm of possibility based on the approximately 60 thousand and 2 million picture-sized training and test data sets respectively. However, one disadvantage of the QDA model is the sizable decrease in specificity between the cross-validation and holdout data sets (0.840 to 0.695). This implies a higher risk of overfitting, especially when compared to the performance stability of logistic regression and LDA models. This, combined with QDA's more exotic functional

form, are yellow flags for the model.

K-nearest neighbor (KNN) also performs well, particularly in cross-validation with high accuracy and precision, though it shows some variability in the holdout set. Penalized Logistic Regression and SVM models (including linear, polynomial, and radial basis function kernels) demonstrate strong ROC AUC scores and high precision, making them robust choices for distinguishing between classes. Random Forest exhibits high accuracy and precision in cross-validation, but there is a notable decrease in specificity in the holdout set, indicating potential overfitting.

Recommendation: Overall, Logistic Regression is recommended as the preferred model due to its balanced performance across metrics and stable results between cross-validation and holdout sets. This makes it a reliable choice for practical application in identifying blue tarps in disaster relief scenarios. While QDA offers slightly higher sensitivity, its risk of overfitting and decreased specificity in the holdout set make it a less stable option. KNN, Penalized Logistic Regression, and SVM models can be considered as robust alternatives based on specific operational needs and resource availability, especially in contexts where high precision and ROC AUC are prioritized. Random Forest, despite its strong cross-validation performance, should be used cautiously due to its potential for overfitting.

Conclusion

Best Performing Algorithm(s) in Cross-Validation and Hold-Out Data

Based on the analysis, Logistic Regression, KNN, Random Forest, and SVM Polynomial emerged as top performers in cross-validation, showing high accuracy, precision, ROC AUC, and sensitivity. Logistic Regression stood out for its balanced and stable performance across both cross-validation and hold-out sets, maintaining high accuracy (0.990), precision (0.998), ROC AUC (0.999), and balanced sensitivity (0.990) and specificity (0.988). While KNN, Random Forest, and SVM Polynomial showed exceptional cross-validation results, their hold-out performance revealed decreased specificity, indicating potential overfitting. Therefore, Logistic Regression is recommended as the preferred model for detecting blue tarps due to its robustness and consistency. KNN, Penalized Logistic Regression, and SVM models are strong alternatives for scenarios prioritizing high precision and ROC AUC, while Random Forest should be used cautiously due to its potential for overfitting. These models ensure reliable and effective performance in real-world detection scenarios, balancing accuracy, precision, sensitivity, and specificity, variations within the training dataset.

Justification of Findings

The findings from cross-validation and hold-out performance highlight the importance of validating models on unseen data. While KNN, Random Forest, and SVM Polynomial excelled in cross-validation, their hold-out performance reveals more nuanced strengths and weaknesses. For instance, while KNN maintained high accuracy and precision, its hold-out specificity decreased slightly. Similarly, Random Forest's hold-out specificity was lower compared to its cross-validation performance, and SVM Polynomial's hold-out specificity dropped significantly. This disparity underscores the necessity of evaluating models on independent datasets to ensure their reliability and robustness in real-world scenarios, where data may present different characteristics from the training set.

Recommendation for Detection of Blue Tarps

Considering the performance metrics and the application context of detecting blue tarps, **Logistic Regression** is recommended as the preferred model due to its balanced performance across metrics and stable results between cross-validation and holdout sets. This stability makes it a reliable choice for practical application in identifying blue tarps in disaster relief scenarios. While QDA offers slightly higher sensitivity, the risk of overfitting and decreased specificity in the holdout set make it a less stable option. KNN, Penalized Logistic Regression, and SVM models can be considered robust alternatives depending on specific operational needs and resource availability, especially in contexts where high precision and ROC AUC are prioritized. Random Forest, despite its strong cross-validation performance, should be used cautiously due to its potential for overfitting.

Relevance of Metrics to Application Context

In the context of detecting blue tarps, accuracy, precision, sensitivity, specificity, and ROC AUC are all crucial metrics. High accuracy and precision indicate the model's overall reliability and its ability to correctly identify true

positives with minimal false positives. Sensitivity is particularly important in this application to ensure that actual blue tarps are not missed. Specificity, while slightly lower, is still relevant as it reflects the model's ability to correctly identify negatives, reducing false alarms. ROC AUC provides a comprehensive measure of the model's discriminative ability across various threshold settings, ensuring balanced performance. Together, these metrics ensure that the selected model performs reliably and effectively in real-world detection scenarios.

Determination and justification of which algorithm works best:

Based on the performance metrics provided, Logistic Regression emerges as the most balanced model across all evaluated criteria, exhibiting particularly high specificity. This model consistently demonstrates robust performance, both in cross-validation and on the holdout data, which underscores its reliability and applicability in practical scenarios. Such consistent results across different datasets highlight its potential for effective real-world deployment, making it a dependable choice for predictive modeling.

What additional recommended actions can be taken to improve results?

To improve our model's performance, we must address the class imbalance present in our dataset, where only about 3% of the training data and 1% of the holdout data represent the 'Blue Tarp' variable. The 'Non-Blue Tarp' class overwhelmingly dominates, which can lead to misleading performance metrics and inadequate generalization for the minority class. Since accurately identifying the minority 'Blue Tarp' class is our primary objective, implementing strategies to mitigate this imbalance is crucial. These strategies could significantly enhance the model's ability to detect the less prevalent but critical 'Blue Tarp' instances, ensuring more reliable and effective outcomes.

What is it about this data formulation that allows us to address it with predictive modeling tools?

Within this dataset, we have a clearly defined target variable, 'Blue Tarp', which we aim to identify. This specificity makes it ideally suited for classification modeling. Moreover, the dataset includes a significant volume of data, particularly in the holdout set, providing a robust basis for training and testing the models. With enhanced techniques to address the class imbalance, such as resampling methods or advanced algorithms, we can further improve the model's accuracy and reliability. This substantial dataset, combined with improved handling of class imbalance, positions us well to effectively utilize predictive modeling tools to tackle this problem.

How effective do you think your work here could actually be in terms of helping to save human life?

The effectiveness of this project in saving human lives could be quite substantial considering the final performance metrics of the predictive model The models, particularly Logistic Regression, demonstrated high accuracy and an excellent balance of sensitivity and specificity. This precision in identifying Blue Tarps ensures that emergency responses can be more accurately targeted to areas where the tarps indicate potential distress or urgent need.

References

- Gedeck, P. (n.d.). *Starting the Disaster Relief Project*. virginia.edu. https://canvas.its.virginia.edu/courses/109177/pages/project-preview-starting-the-disaster-relief-project?wrap=1
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: With applications in R. Springer US Springer.
- International Journal of Scientific and Research Publications. (2023). IJSRP research paper format. Research Paper Format, Template for Research Paper. https://www.ijsrp.org/ijsrp-paper-format.php
- OpenAI. *ChatGPT* [Large language model]. https://chatgpt.com/c/0a4a7242-3f5f-452d-b1b2-058780c57f77. Converted word doc text to research paper format ab ove.

- Wikimedia Foundation. (2024, June 7). 2010 Haiti earthquake. Wikipedia. https://en.wikipedia.org/wiki/2010_Haiti_earthquake