Disaster Relief Project

Group 11: Becky Desrosiers, Abner Casillas-Colon, Rachel Daniel

2024-04-23

Introduction

The 2010 earthquake in Haiti was a devastating natural disaster that caused extreme damage and displaced millions of people. After this disaster, the rescue workers needed to deliver food, water, and other resources to the people of Haiti, but it was challenging locating the people in need over this large area. These challenges included roads blocked by rubble and the inability to communicate because of damage to communication infrastructure. The rescue workers needed another strategy other than physically looking for them on land to locate and reach these displaced persons quickly and more efficiently.

The people of Haiti who were displaced by the earthquake were using blue tarps as temporary shelter. This knowledge was utilized to locate these people after imagery was collected by aircraft flown by a rescue team from the Rochester Institute of Technology. Blue tarps could be searched for within these images by the rescue team who would then go to these coordinates and find them. However, that strategy would have been too slow, and resources would not have been delivered in time. A different strategy that could be used to locate people more efficiently in need of resources was to continue to use these images but instead use data-mining algorithms to search the images.

This report explores various classification methods that could be useful in locating these blue tarps within the images taken by aircraft. The algorithms tested and explored include those that utilize K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression with and without penalty, a boosted ensemble model using XGBoost, and three different support vector machines (SVMs): one with a linear kernel, one with polynomial and one radial. We expect at least one of these methods to work more efficiently and more accurately than human efforts as these algorithms can handle more data, perform complex operations, and potentially fit to the patterns of the data to identify the blue tarps. The chosen algorithm will likely miss fewer blue tarps than a human would. It is critical to identify the algorithm that will perform the best so that aid can be delivered to displaced people in dire circumstances. This is an important data-mining problem that could have a large impact on human life. This report will identify the algorithm that most accurately and most efficiently identifies these blue tarps so that the displaced Haitians can receive desperately needed resources in time.

Data

The datasets used consist of Class as a response variable and Red, Green, and Blue values as the predictor variables. There are a total of 63,241 records in the training dataset and 2,004,177 in the holdout set, with no missing values. RGB values have a range of 0-255. In the training dataset, Red and Green values share a minimum value of 48 and Blue has a minimum of 44. In the holdout set, the Red and Green minimums are 27 and 28, respectively, and the lowest Blue value is 25. For Class there are five distinct values: Blue Tarp, Rooftop, Soil, Various Non-Tarp, and Vegetation.

The box plots in Figure 1 showcase the distribution of the range of RGB values grouped by each of the observed categories. We notice that, while the Blue Tarp category overlaps with some other categories on each feature, there is no one category that strongly overlaps with Blue Tarp on all three values. Therefore we have good reason to believe that the Blue Tarp category will be distinguishable from the others in our models. The Blue Tarp class has RGB values that range from 150-200 Red, 160-250 Green, and 175-250 Blue. The highest values are blue, indicating a high saturation of blue in the target class, which is to be expected, since we are looking for blue tarps.

When viewing the boxplots, there is some overlap of red and green values for the rooftop class and the various non-tarp class with the blue tarp class. The blue values do not overlap as much across classes as the blue tarps have the highest saturation of blue pixels. As we work to identify the best model to identify blue tarps, we will keep in mind that there is some overlap of red and green values of the various non-tarp class and the rooftop class with the blue tarp class. This overlap may potentially make up a portion of the false positives in our final model so we will try to minimize false positives as much as possible while also minimizing false negatives.

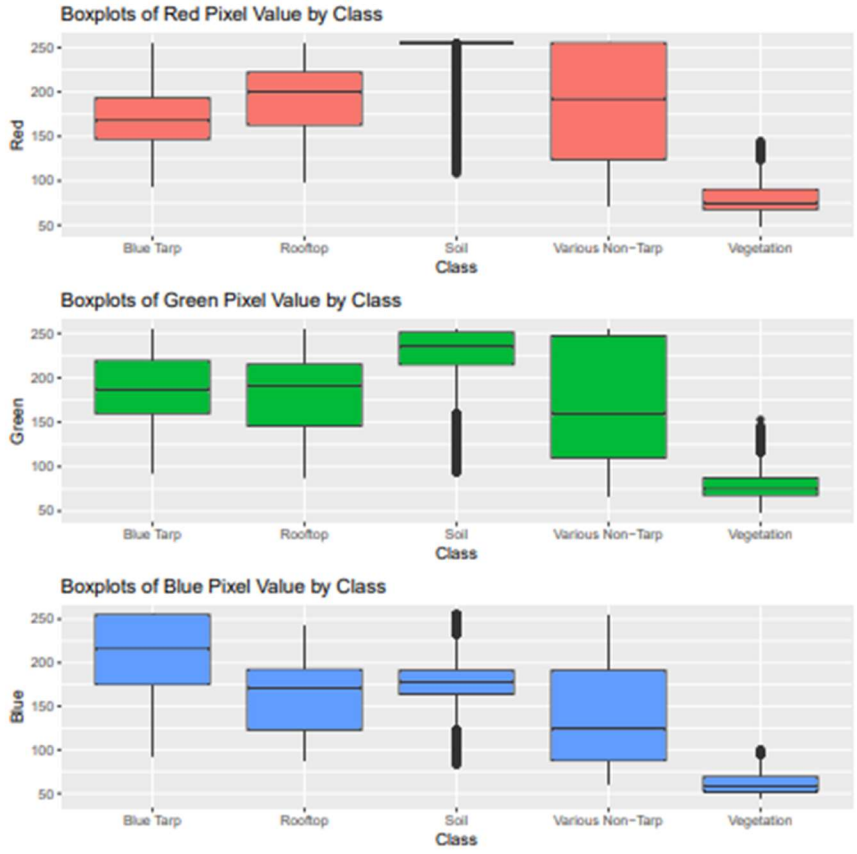

Figure 1: Box plots of the training data pixel values broken out by each colour and class.

The count plot in Figure 2 displays the total counts grouped by class for the training dataset. Blue Tarp has the smallest amount of values at 2022. This is important to note for the threshold selection later on in the model. It is also promising to note that the two largest values by far are vegetation and soil, as they are the two classes that overlap the least with blue tarps on RGB values, and thus are least likely to represent false positive or false negative values when predicting for tarps. Rooftops and Various Non-Tarps however, do account for 14,647 pixels which have closer overlaps with the green and blue values for blue tarps. These classes are more likely to be mistaken by a model as blue tarps.

While it is important to assess RGB values for each class in order to understand our date, our main priority is to identify blue tarps. Furthermore, some of the models we are exploring, such as logistic regression, can only effectively distinguish between two classes. Since we are targeting only one class, we now transition into a binary problem where we classify the data between only Tarps and Non-Tarps.

Both the bar chart in Figure 3 and Table 1 highlight the values of Blue-tarps versus Non-Tarps. We note that the proportion of tarp pixels in the total dataset is 3%. When we conduct our threshold selection across all models, this will be important as the models will default to a 50-50 threshold. With such a rare target event, it is likely that we will need to significantly lower the threshold in order to correctly identify as many blue tarps as possible. We also need to keep in mind that an indiscriminant model which assigns every pixel to the Non-Tarp class will have a 97% accuracy, so the requirement for an effective model will be a higher accuracy than might appear very attractive at a glance.

In the box plots in Figure 4, we can again see the overlap of the red and green pixel values of the Non-Tarp class with the Tarp class, as well as the relative separation between the blue pixel values of Tarp and Non-Tarp. We expect our models to be able to capitalize on this distinction. We expect any shrinkage models to weight the Blue feature more heavily, because it appears to provide the most information to distinguish blue tarps from other pixels.

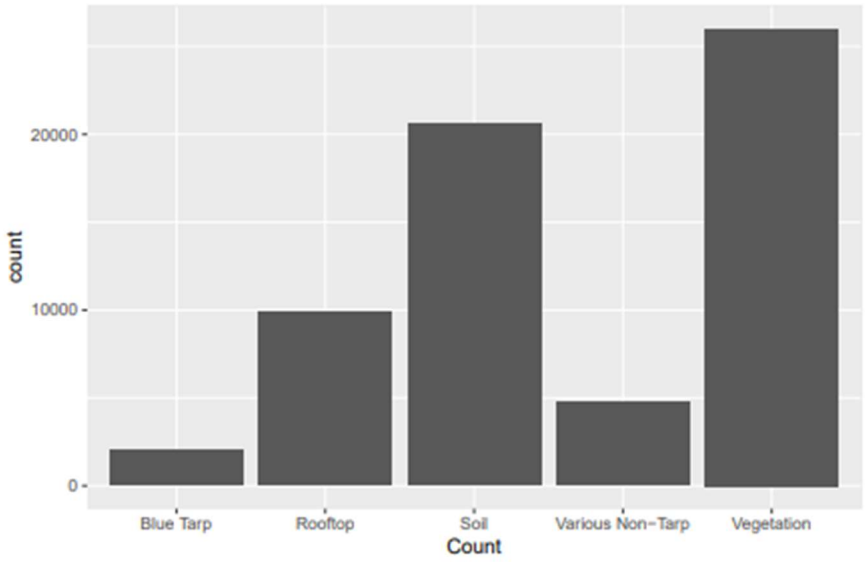

Figure 2: Count plot of total counts for the training data broken out by each class

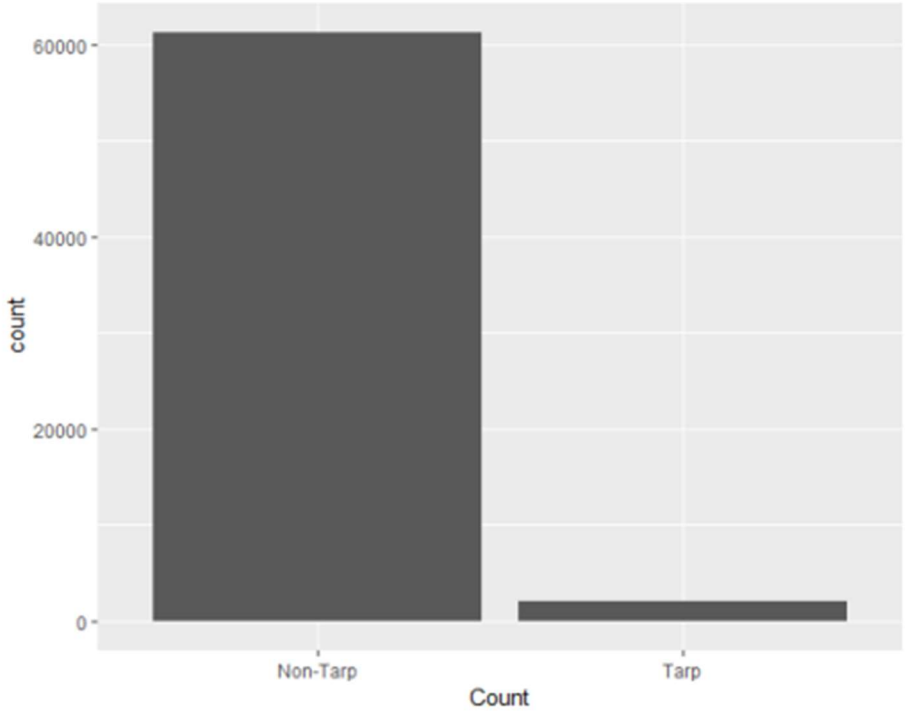

Figure 3: Count plot of Classes Blue-Tarp and Non-Tarp for the training data

Table 1: Count of Class

Var1	Freq
Non-Tarp	61219
Tarp	2022

Table 2: Count of Class for holdout set

Var1	Freq
Non-Tarp	1989697
Tarp	14480

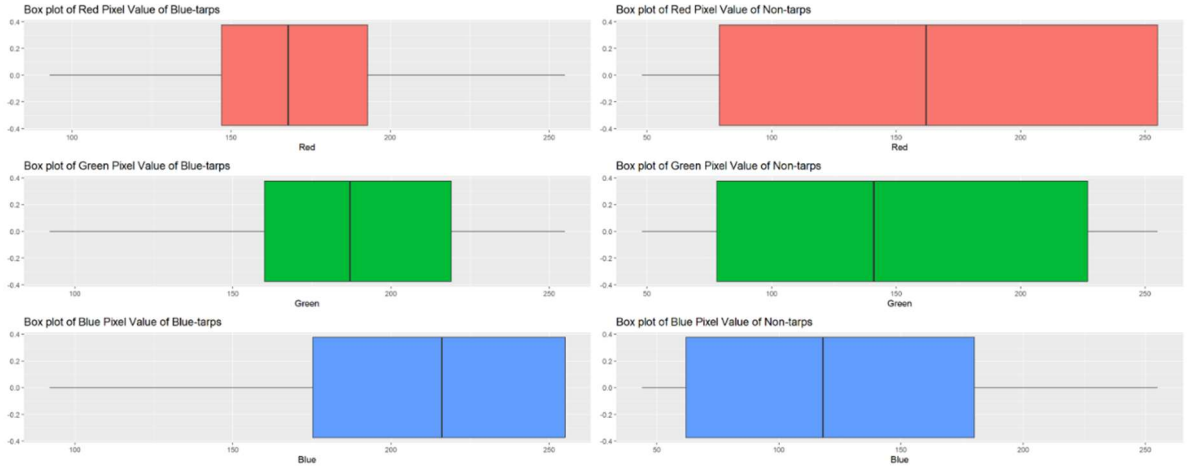

Figure 4: Box plots of the training data pixel values broken out by each color and class.

We now examine the holdout set to see how it compares to our training set. In Figure 5, we see a bar chart of the holdout data. Table 2 displays the corresponding numeric data. The difference in the count between the tarps and non-tarps is even more extreme than with the training dataset, where the blue-tarps pixels only make up about 0.7% of the data. We expect very high accuracy from our models' performance on the holdout set, because even an indiscriminant model assigning every observation to the Non-Tarp class will have an accuracy of 99.3%.

The box plots in Figure 6 show the distribution of RGB values for blue-tarps versus non-tarps. We observe that the distributions of the pixel values are different between the training data and the hold out set, with a much peakier distribution for Red and Green values, and most Blue values below 100 for Non-Tarps. The difference in the data has the potential to introduce significant variance error into our models, and we will need to pay attention to possible overfitting in our more flexible models, such as KNN.

Description of Methodology

This investigation was carried out using R, an open-source statistical analysis software that offers a variety of packages to assist in model building and model selection. We used the following packages to execute our analysis: tidyverse, tidymodels, ggcorrplot, GGally, patchwork, discrim, kernlab, and doParallel.

The first step of this analysis is to change the class category to reflect our variable of interest. The raw data classifies the pixels into Blue Tarp, Rooftop, Soil, Various Non-Tarp, and Vegetation, however we are only interested in whether the pixel is a Blue Tarp or not a Blue Tarp. We changed the factor outcome variable to have only two levels: Tarp and Non-Tarp. This is a classification problem, and we will explore the following models: KNN, LDA, QDA, Logistic Regression, Penalized Logistic Regression (a logistic regression model with tuning parameters for an elastic net), a boosted ensemble, and one linear, one polynomial, and one radial SVM.

Every model will be tuned except for the LDA and QDA, which have no parameters to tune. Logistic regression will also be investigated with zero penalty, meaning that no parameters will be tuned. Tuning parameters will be selected to maximize ROC AUC. Accuracy will not be used to select best parameters because the accuracy will be calculated at a 0.5 threshold, which will not provide the most accurate results for our purpose. The models will be tuned using a latin hypercube grid to explore the parameter space.

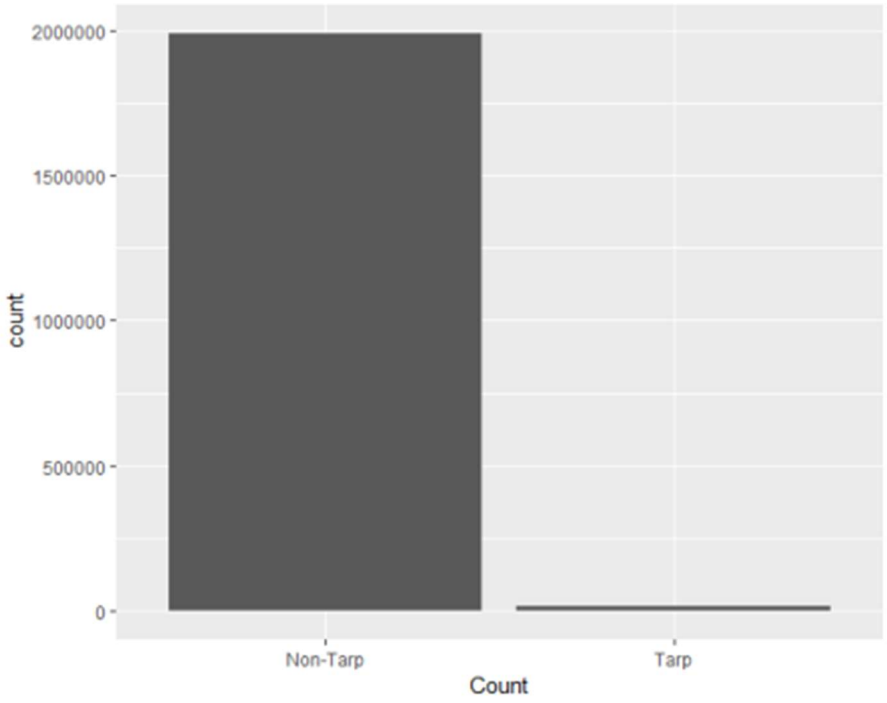

Figure 5: Count plot of Classes Blue-Tarp and Non-Tarp for the hold-out set.

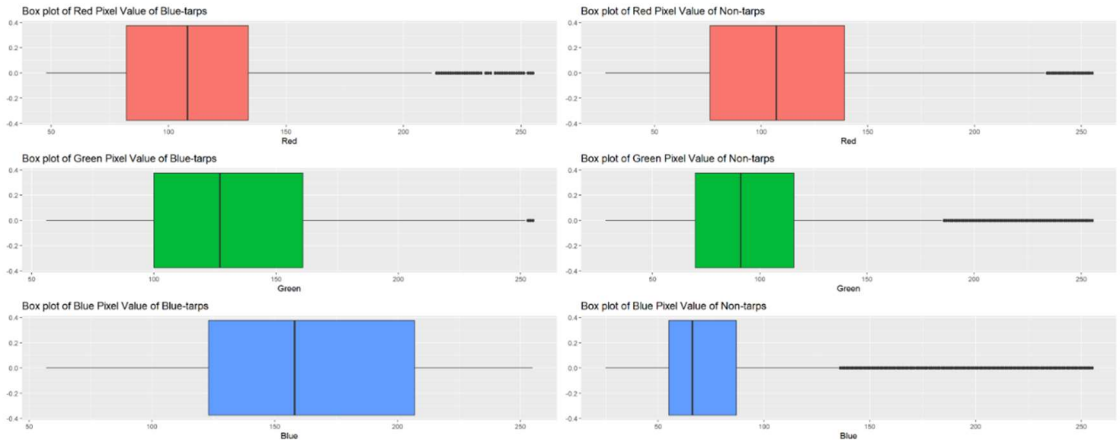

Figure 6: Box plots of the pixel values broken out by each color and class for the hold-out set.

The tuning parameters for each model are as follows:

- KNN:
  - k (number of neighbors)
- Logistic regression:
  - penalty
  - mixture
- boosted ensemble:
  - tree depth
  - learn rate
- linear SVM:
  - cost
  - margin
- polynomial SVM:
  - cost
  - margin
  - degree
- radial SVM:
  - cost
  - margin
  - rbf sigma

All models will be validated using tenfold cross-validation. ROC AUC of the model on the training data subset will be estimated using tenfold cross-validation, and other metrics will be estimated based on the model's predictions at the chosen threshold, since tidyverse packages do not currently support cross-validation at a threshold other than 0.5. With an overall sample size of 63,241, this allows for each fold to have 6,324 records. With this amount of data, a tenfold cross validation is a reliable method to evaluate the performance of the training sets for all models. All metrics on the holdout data will be estimated based on the models' predictions at the chosen threshold. We set the seed for all random processes to 6030.

As established during the EDA, the proportion of Blue Tarps to all other classes is approximately 3%. This indicates that a 50% threshold would be a poor choice for the final model because our issue is concerned with finding as many blue tarps as possible. While five metrics will be explored for threshold selection (accuracy, j-index, sensitivity, specificity, and precision) our primary metrics of interest will be sensitivity and j-index. Sensitivity is especially important because we want to minimize false negatives - those could represent individuals that could be missed when response time is extremely important. The j-index is chosen as a metric that balances sensitivity and specificity. This will be valuable in models were responders have limited resources to check potential points of interest. While maximizing based on j-index may result in a higher false negative rate, we will greatly reduce false positives, which will make sure that we can make the most out of the time and resources allocated to the relief effort.

To select a model and determine model performance, we will also rely heavily on sensitivity and j-index, for similar reasons. We will also look at the area under the ROC curve, as this represents overall model performance, and the threshold can be chosen to optimize j-index or sensitivity.

Results of Model Fitting, Tuning Parameter Selection, and Evaluation

K-Nearest Neighbors

We now build a K-nearest neighbors (KNN) model with the number of neighbors, k, tuned to find the best fit. Figure 7 shows the ROC AUC of the model when fitted with different numbers of neighbors. It shows that the AUC of the ROC curve increases with k, maxing before 50, and leveling out at higher tuning values. The highest AUC is **0.9943** and belongs to the model with k = 40 neighbors, so we use this tuning metric moving forward.

With the number of neighbors tuned to k = 40, our model has an ROC AUC of 0.9943. The ROC curve will be displayed in the next section, when all of the models are compared. The next step is to select the appropriate threshold for our model. Figure 8 shows five different metrics plotted against the threshold value, with thresholds ranging from 0.01 to 0.99. The metrics were calculated using tenfold cross-validation. The vertical lines show that sensitivity is maximized anywhere between 0.01 and 0.09, and j-index is maximized at 0.09. Therefore, we will use threshold = 0.09 in order to maximize both the sensitivity and j-index.

Our final model has k = 40 nearest neighbors and a selection threshold of 0.09. The corresponding confusion matrix, representing this model’s performance on the training dataset, is shown below.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	1999	291
Non-Tarp	23	60928

The confusion matrix shows that our final model misses only 23 Blue Tarp pixels in the training set and falsely identifies 291 Non-Tarp pixels.

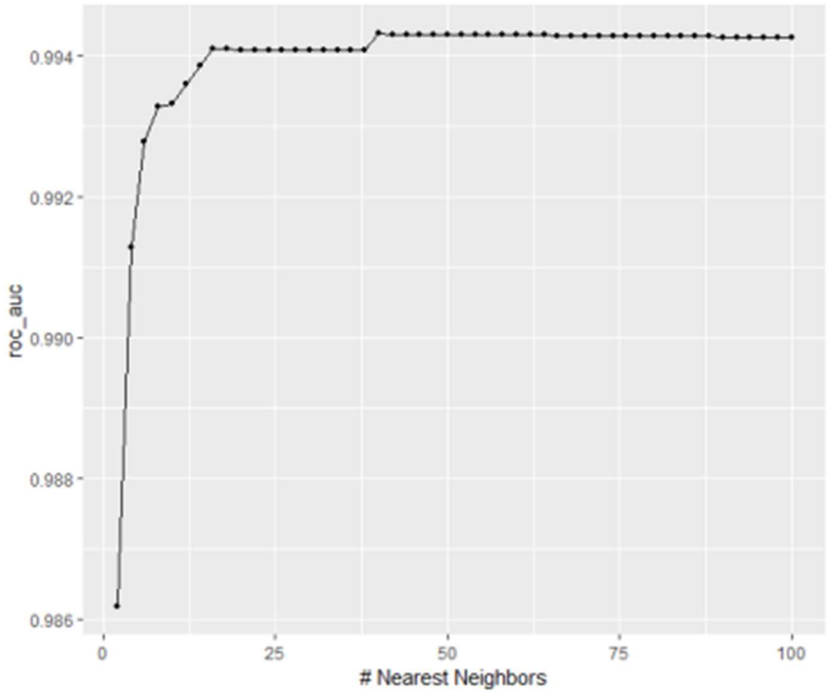

Figure 7: ROC AUC based on number of principal components and nearest neighbors.

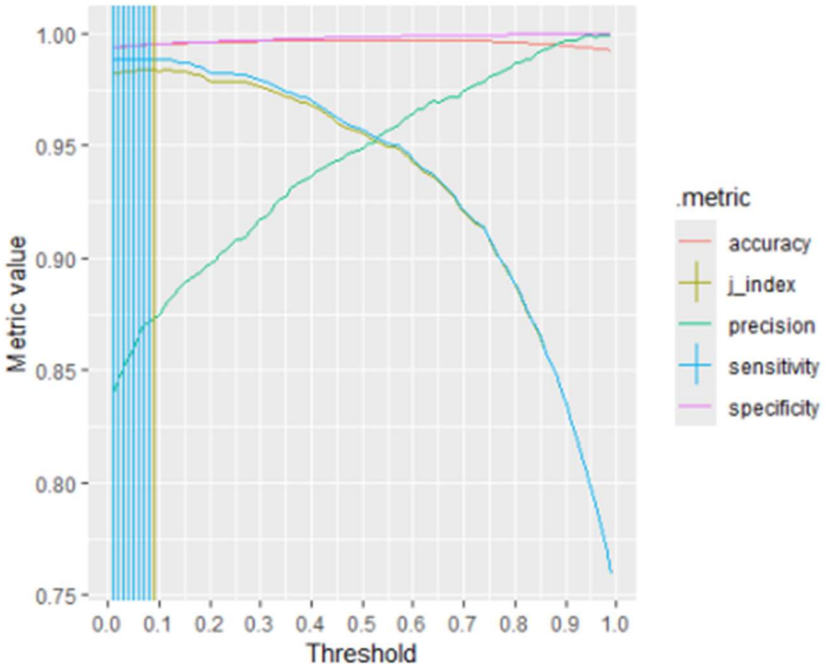

Figure 8: Five metrics based on threshold value for the KNN model with k = 40 nearest neighbors. Vertical lines show maximum sensitivity between 0.01 and 0.09, and maximum j-index at 0.09.

On the holdout set, the KNN model with 40 neighbors produces a ROC AUC of 0.9643 and the chosen threshold classifies the data as shown in the confusion matrix below:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	13452	31424
Non-Tarp	1028	1958273

The KNN model performs a little worse on the testing data, which is to be expected. The true positive (sensitivity) rate dropped from 0.9886 to 0.9290 and the j-index dropped from 0.9839 to 0.9132. That is a significant drop, and there is a possibility of some overfitting here. In the next section, we will compare the models to judge the amount of overfitting.

Linear Discriminant Analysis

The second model under consideration is linear discriminant analysis (LDA). We found that the LDA model has a ROC AUC of 0.9888, which means that it performs well on the data and classifies blue tarp pixels versus non-tarp pixels much better than a random model. The ROC curve corresponding to the LDA model will be displayed with the other curves in the next section. To choose an appropriate threshold, we reference Figure 9, which displays the threshold plot with the plotted values of the chosen metrics from 0.01 to 0.99.

A threshold of 0.01 maximizes both sensitivity and j-index, so we will use this threshold moving forward. After threshold selection, the confusion matrix for the model was assessed to find the number of false negatives, false positives, and true positives. We found that the LDA model at the chosen threshold misses 231 blue tarps and falsely identifies 1284 blue tarps, while correctly identifying 1791 of the 2022 total tarp pixels in the dataset.

		Truth
Prediction	Tarp	Non-Tarp
Tarp	1791	1284
Non-Tarp	231	59935

The LDA model performs with a high level of accuracy (0.9760), however, due to the rare nature of the blue tarps, a model classifying all pixels as Non-Tarp would have an accuracy of 0.9680. We will compare this model in the next section in order to determine if the performance is significantly useful. As the confusion matrix shows, the LDA model misclassifies a significant number of blue tarps.

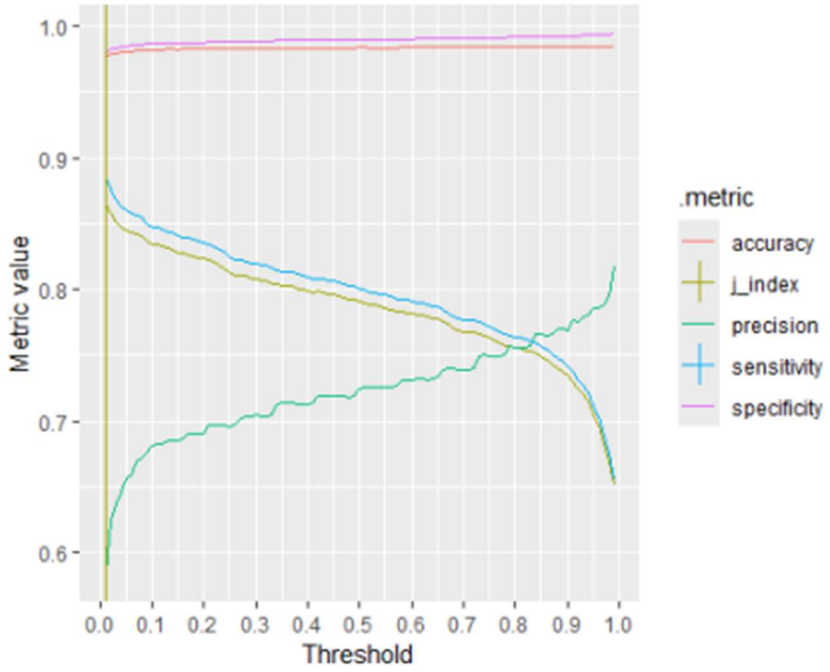

Figure 9: Four metrics based on threshold value for the LDA model. Vertical lines show maximum j-index and sensitivity at 0.01.

On the holdout set, the LDA model produces a ROC AUC of 0.9921 and the chosen threshold classifies the data as shown in the confusion matrix below.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	13860	63633
Non-Tarp	620	1926064

From the confusion matrix, we can calculate a sensitivity of 0.9572 and j-index 0.9252, which is actually higher than the j-index for the training set (0.8648). The other metrics decreased marginally, however, which is expected for a test set. The amount of decrease in metrics does not appear significant enough to indicate problematic overfitting.

Quadratic Discriminant Analysis

The next model to be considered is based on quadratic discriminant analysis (QDA). We found that the model's AUC was 0.9982, which means it performs a lot better than a random model. The next step was selecting the best threshold based on sensitivity and j_index. We found that the maximum sensitivity is at a threshold of 0.01, while the maximum j-index was at 0.02. These metrics at each threshold value are shown within the threshold plot in Figure 10.

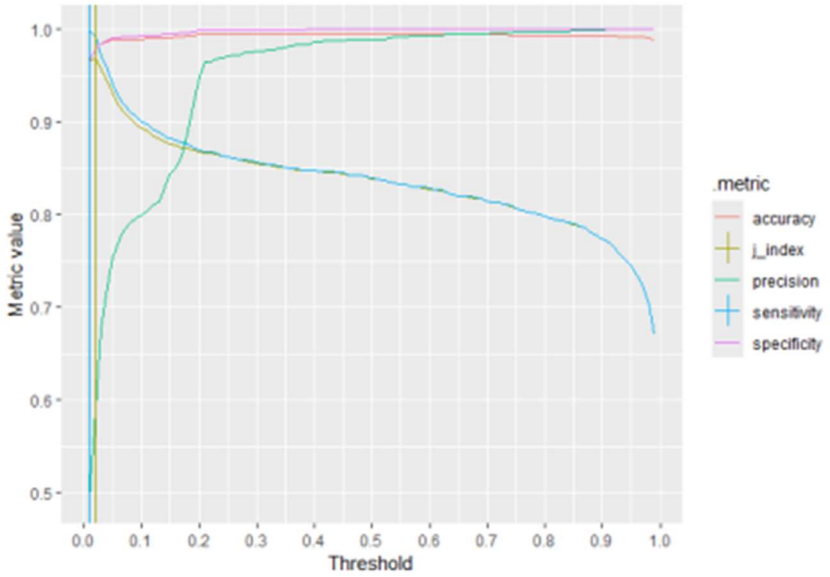

Figure 10: Four metrics based on threshold value for the QDA model. Vertical lines show maximum j-index and sensitivity at 0.02 and 0.01, respectively.

Since sensitivity and j-index are both important for this data, we choose a threshold in between 0.01 and 0.02 for this model, and we will use threshold = 0.015 going forward. This threshold value seeks to maximize both metrics, resulting in a j-index of 0.9692 and sensitivity of 0.9975. The confusion matrix below represents the QDA model's performance on the training dataset.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	2017	1732
Non-Tarp	5	59487

The model performs very well at the selected threshold in terms of locating tarp pixels, missing only five (5). However, it also misclassifies a large number (1732) of non-tarp pixels. When choosing a final model, it will be important to keep in mind the practicality of the model.

The QDA model appears to perform better than the LDA model in terms of sensitivity and j-index, so it seems that the data benefits from some flexibility in the predictive algorithm. The risk of more flexibility, however, is overfitting. We now check the confusion matrix for the holdout data.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	13586	71822
Non-Tarp	894	1917875

We can see that the QDA model performs worse on the testing data: the accuracy dropped from 0.9725 to 0.9637 and the true positive rate dropped from 0.9975 to 0.9383. It's possible that the flexibility of the method led to overfitting, however we will compare the change in metrics between the models in the next section to get a better idea of how significant the change is.

Logistic Regression Analysis and Penalized Logistic Regression Analysis

We investigated two types of logistic regressions: with and without penalty. The standard logistic regression model does not have any parameters to tune, and the penalized model applies a penalty value on predictor variables to attempt to shrink less important features. The two tuning parameters of interest are mixture and penalty. Mixture is a metric that will determine if the model is closer to a ridge regression or lasso regression and penalty measures how strongly the predictor variables are penalized, resulting in more shrinkage. The tuning results are displayed in Figure 11.

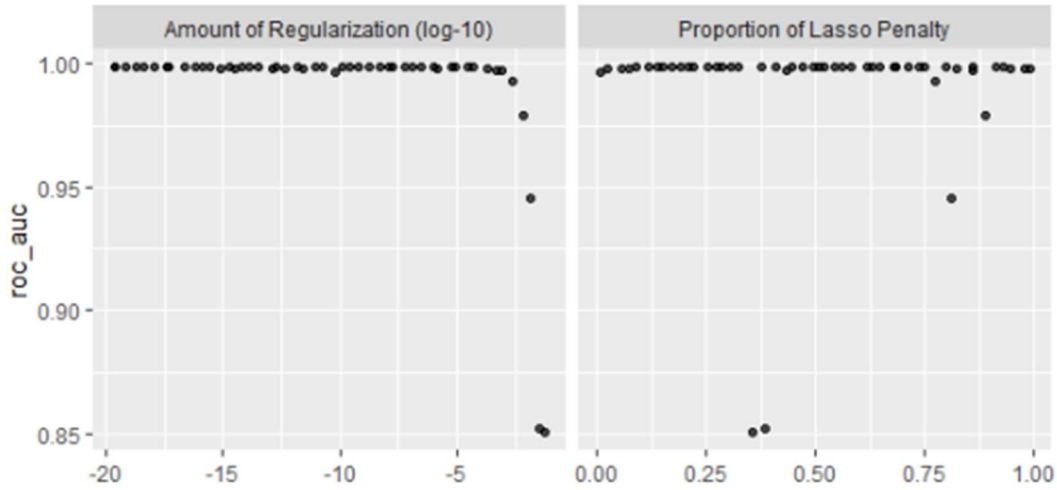

Figure 11: ROC based on penalty and mixture values. Left hand chart corresponds to penalty value, right hand chart corresponds to mixture.

For the penalized model the parameters were tuned to mixture = 0.208 and penalty = 1.279e-10. From examining Figure 11, we can see that larger penalty values incur worse performance after about 10e-5. The relatively low mixture value causes the model to be closer to ridge regression, which does not reduce the coefficients of any predictors to zero like lasso. The optimal parameter values reflect how important all predictor variables are for identifying the class of the observation in the model, and that reducing all values proportionally does not benefit the model to a large degree. The ROC AUC value for the tuned model is 0.9986, while the untuned model has a ROC AUC of 0.9985. Because the difference between the models is very small based on penalty and AUC, a practical approach may choose the standard logistic regression for its simplicity. However, if our purposes do not include interpretability, we may choose the penalized model if it shows even a slight advantage in predictive capability.

Our two metrics of interest for threshold selection are the j-index and sensitivity which can be seen along with accuracy, precision, and specificity in Figures 12 and 13. For the logistic regression model, threshold = 0.05 maximizes the j-index and threshold = 0.01 maximizes sensitivity. For the penalized logistic regression model, threshold = 0.04 maximizes the j-index and threshold = 0.01 maximizes sensitivity. As the overall split of blue tarp vs non blue tarp is 3%, we will utilize the .05 threshold for the standard logistic regression and .04 for the penalized logistic regression to maximize j-index. Selecting these thresholds will result in more of a balance between sensitivity and specificity, substantially reducing the number of false positives compared to the .01 threshold.

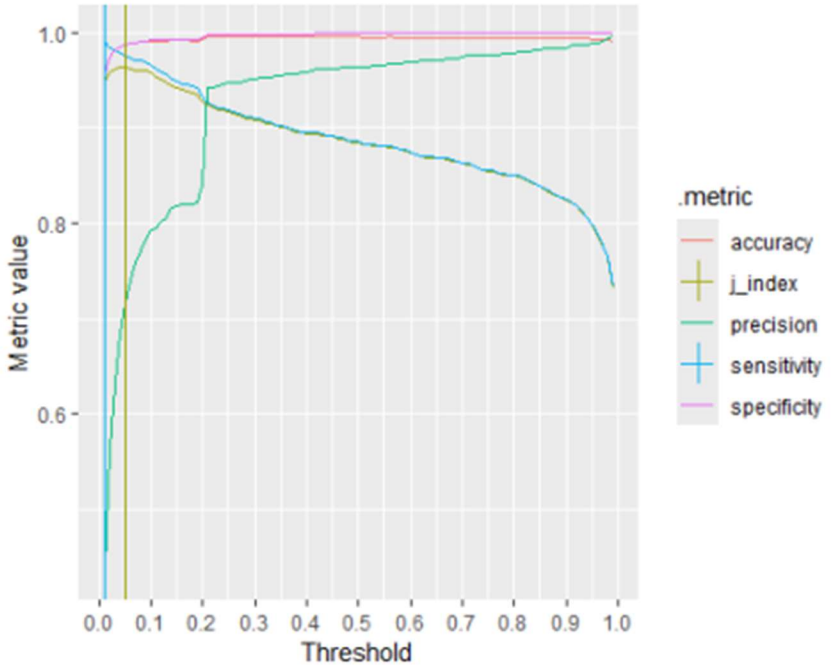

Figure 12: Five metrics based on threshold value for the Logistic Regression model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.05.

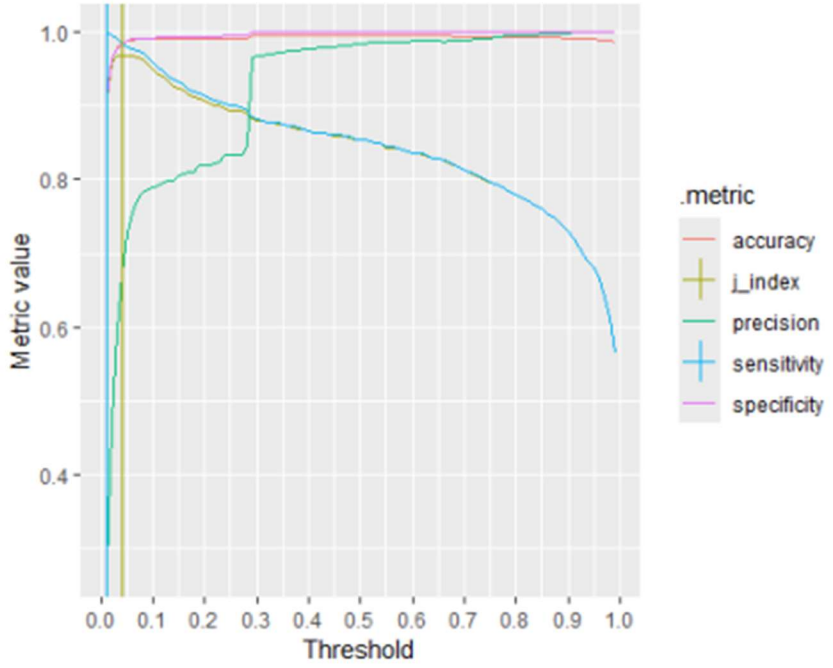

Figure 13: Five metrics based on threshold value for the Penalized Logistic Regression model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.04.

The confusion matrices below show the breakdown of each model's predictions at the chosen thresholds on the training dataset.

Standard Logistic Regression:

		Truth
Prediction	Tarp	Non-Tarp
Tarp	1974	786
Non-Tarp	48	60433

Penalized Logistic Regression:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	1990	961
Non-Tarp	32	60258

The confusion matrices show that the penalized model is more sensitive, resulting in fewer false negatives. However, it comes at the cost of almost 200 extra false positives. The performance on new data is now investigated. The untuned model produced an ROC AUC of 0.9994 and the tuned model had an AUC of 0.9997, indicating higher performance that reflects the metrics from cross-validation. To investigate further, the confusion matrices on the holdout set are displayed below.

Standard Logistic Regression:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	14479	193475
Non-Tarp	1	1796222

Penalized Logistic Regression:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	14480	206144
Non-Tarp	0	1783553

Both models demonstrate almost perfect sensitivity, with the penalized model correctly identifying every blue tarp pixel. However, the one final pixel identified by the penalized model comes at the expense of over twelve thousand (12,000) misclassified pixels that are not of interest. One pixel is very unlikely to represent an entire new tarp, but 12,000 false positives could represent many locations searched without finding any people in need of help. Both models have extremely high sensitivity compared to the others and a much lower specificity.

Boosting Model using XGBoost

Next we investigate a boosted tree model that creates an ensemble of decision trees based on the results of previous trees. We chose this decision tree model over a Random Tree model since it learns from previous results and can potentially provide us with a more accurate predictive model. We performed tuning on the parameters tree_depth and learn_rate. Figure 14 shows the AUC for various parameter selections for this model. It reveals that the best learning rate is 0.316 with its best tree depth at 13. We will be keeping this model with these parameters as they give us the highest AUC of 0.9993.

We then choose the most appropriate threshold for our model with our chosen parameters. Displayed in Figure 15 are various metrics plotted against increasing threshold values. The plot reveals that the maximum sensitivity for this model is very slightly above zero at around .01 and the maximum J-index around the same at around .04. We will utilize the threshold where the j-index is at its greatest for our model, which is a threshold of 0.04.

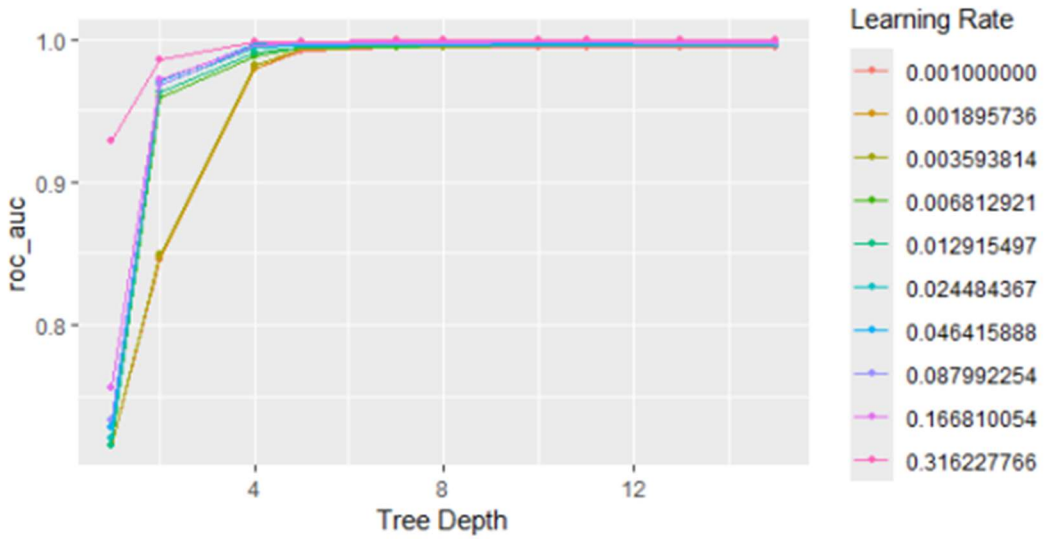

Figure 14: ROC values of the Boosted Gradient model that considers different learning rates along the tree depth

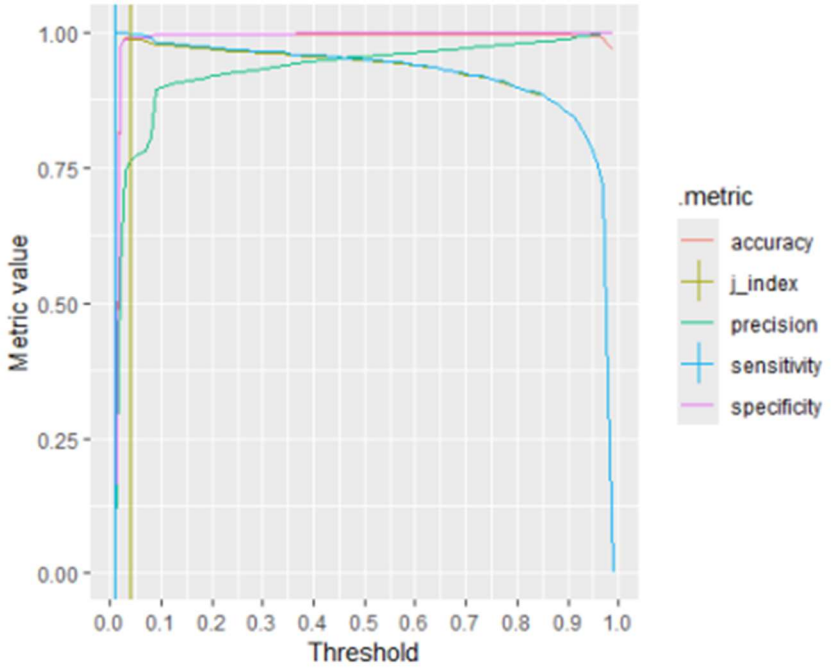

Figure 15: Five metrics based on threshold value for the XGBoost Model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.04.

The confusion matrix for the model with our chosen parameters on the training dataset is displayed below. It reveals that there are only five pixels mislabeled as Non-tarp when they are actually Tarp, which indicates a good model fit. Five pixels would not make a large difference since many pixels will make up one Tarp. There are 626 pixels that are identified as Tarp when they are actually Non-Tarps, but that is also not too high of a number pixel-wise.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	2017	626
Non-Tarp	5	60593

When testing the fitted XGBoost model on the holdout set (displayed below), the AUC is a little lower than the AUC for the training set at around 0.9700. We notice a drop in performance, but the model still performs very well overall. We can

see in the confusion matrix displayed below that there are more tarp pixels missed (617 vs 5 for the training data) and even more misidentified as Tarp when they are actually Non-Tarp pixels. However, the holdout set is a great deal larger than the training data so it makes sense that more will be missed or misidentified. The percentage of tarps missed or falsely identified is slightly higher for the holdout set, but still considerably small and not concerning.

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	13863	64505
Non-Tarp	617	1925192

Support Vector Machine Model

Support vector machines are models which frequently perform well in a wide variety of applications and are worth considering for this analysis. We analyzed support vector machines with three kernel types: linear, polynomial, and radial. Support vector machines have various tuning parameters to consider depending on the kernel chosen. The first parameter that is tuned for all models is cost, which represents a measure of how much leeway is given for observations to be on the incorrect side of the separating hyperplane. A larger C value creates a model with higher bias but lower variance and vice versa. Our next parameter is insensitivity margin, which indicates the value of the epsilon in the loss function. This value effectively determines the width of our area around the hyperplane for values that will be penalized. The polynomial kernel has a degree parameter that signifies the order of the kernel function (i.e. 2nd degree vs 3rd degree polynomial). The sigma in the radial function represents how the decision boundary is drawn. A larger sigma will reduce variance but increase bias and vice versa.

The Linear SVM performed best with a cost of 31.49 and a margin of 0.14 These parameter values allow for more flexibility, which makes sense as the data set may not fit a perfectly linear decision boundary. With these parameters, the model allows for more misclassifications of values near the decision boundary. Figure 16 shows the results of tuning the linear SVM.

The plot of tuning results for the polynomial SVM in Figure 17 highlights the deficiencies with the linear model when examining the parameter tuning. The lowest performing is a 1st degree (linear model) followed by a 2nd degree model, and the best model is a 3rd degree polynomial. The optimal cost is .1 and the optimal margin is .003. This model has a much smaller cost and narrower margin relative to the linear model which creates a model with relatively higher variance but lower bias. These parameters also reflect how the data is likely to be non-linear in nature.

The Radial SVM has a cost of 25.52, a margin of .13 and a Sigma of .006. Figure 18 shows the results of tuning the radial SVM. The cost and margin here are more in line with the linear model, though as the figure highlights, these parameters are relatively consistent throughout the range. This allows the radial function to be slightly more biased to the dataset and outperform when it comes to classifications.

The two chosen threshold metrics to determine threshold are the j-index and sensitivity. Figures 19 through 21 show the thresholds that maximize our chosen metrics. We choose threshold values at 0.06, 0.03, and 0.02 for the linear, polynomial, and radial SVM models respectively. We choose to maximize the j-index. The confusion matrices on the training dataset are displayed below.

Linear SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	1968	851
Non-Tarp	54	60368

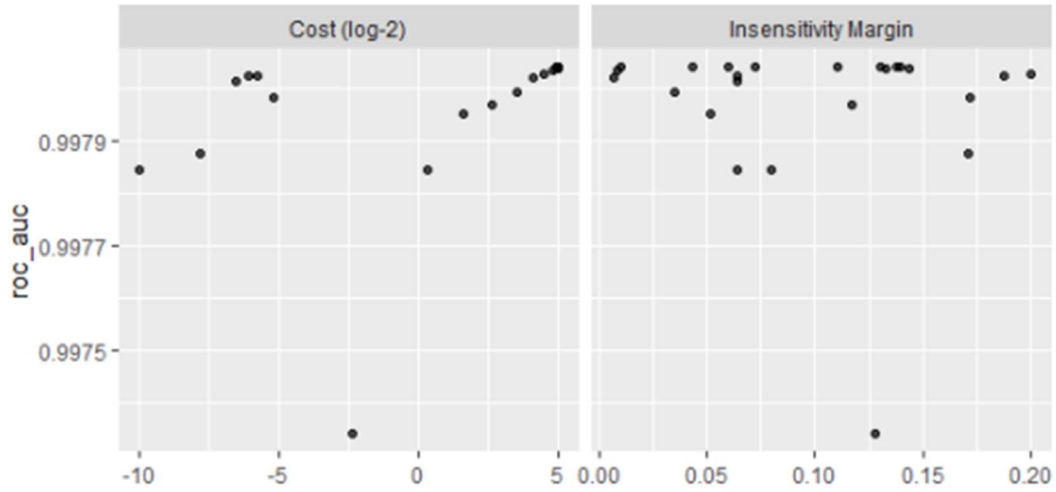

Figure 16: Cost value for the SVM to the left that dictates cost for values outside of margin and insensitivty margin dictates tolerance levels of margin

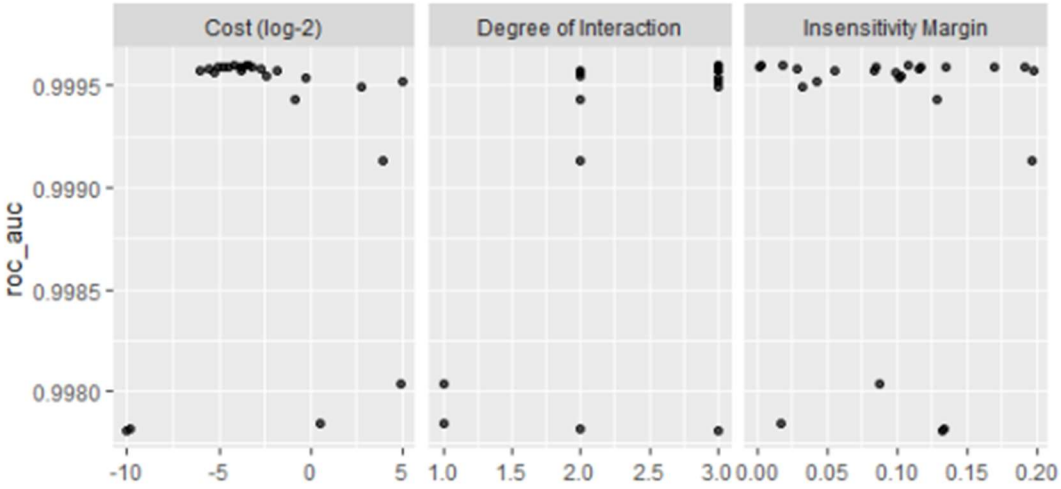

Figure 17: Cost value for the SVM to the left that dictates cost for values outside of margin, degree of interaction refers to the degree of the interaction term, insensitivty margin dictates tolerance levels of margin

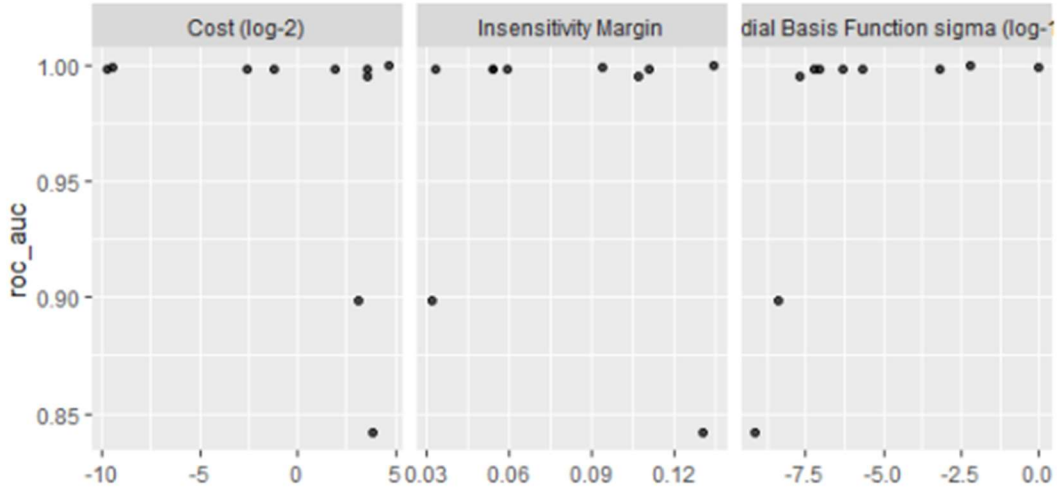

Figure 18: Cost value for the SVM to the left that dictates cost for values outside of margin,insensitivty margin dictates tolerance levels of margin, sigma represents how flexible the decision boundary is with larger numbers being more biased

Polynomial SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	2018	635
Non-Tarp	4	60584

Radial SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	2014	781
Non-Tarp	8	60438

We note that all of the models catch most blue tarps, with the linear SVM lagging behind the others in terms of true positives. The polynomial SVM has less false positives and false negatives when compared to the other two models. This may indicate that it is the beat model of the three. All of the model metrics will be compared in the next section, including those for the models discussed previously.

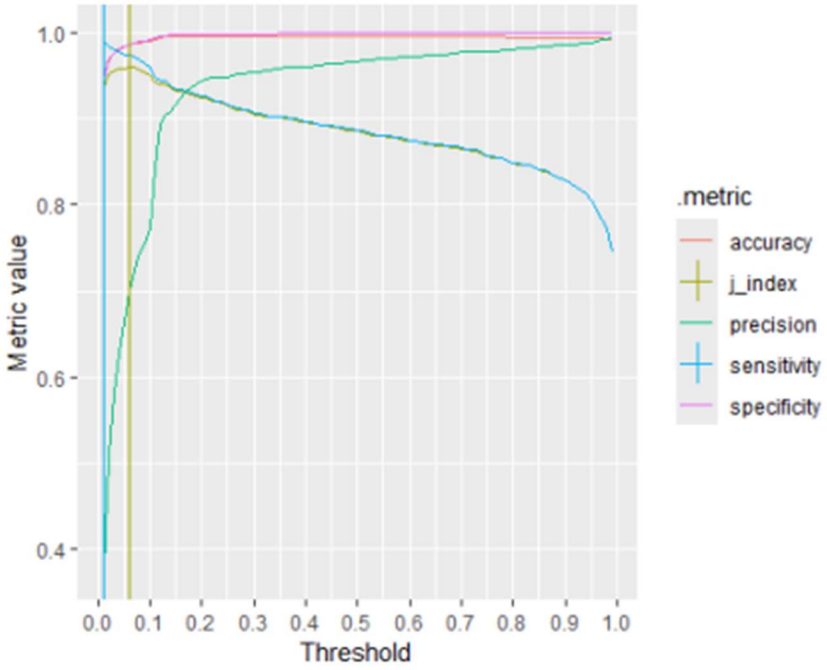

Figure 19: Five metrics based on threshold value for the Linear SVM Model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.04.

Below are the confusion matrices for all the SVM on the holdout dataset. All of the models perform well in the holdout sets with the Polynomial and Radial obtaining the same number of true positives and the linear shortly behind. The Polynomial model has the lowest number of false positives with 101,993 as opposed to the 170,042 of the radial, and 204,855 of the Linear. This is a fairly large difference in performance and will be worth examining further in the results section to follow.

Linear SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	14478	204855
Non-Tarp	2	14478

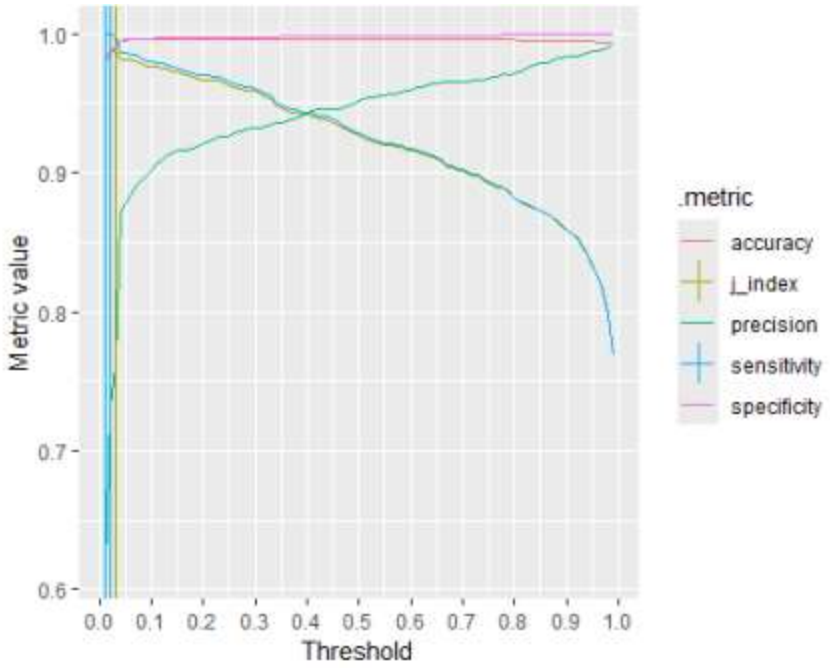

Figure 20: Five metrics based on threshold value for the Polynomial SVM Model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.04.

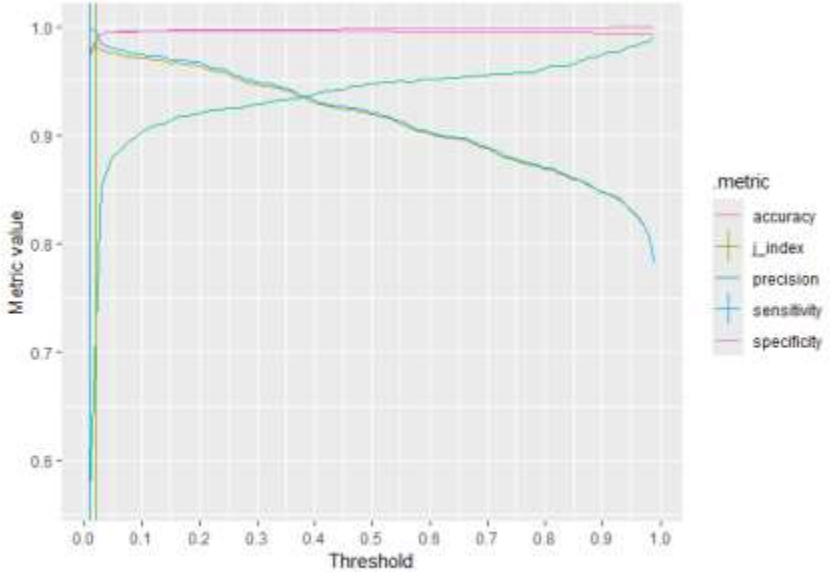

Figure 21: Five metrics based on threshold value for the Radial SVM Model. Vertical lines show maximum j-index and sensitivity between 0.01 and 0.04.

Polynomial SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	14480	101993
Non-Tarp	0	1887704

Radial SVM:

	Truth	
Prediction	Tarp	Non-Tarp
Tarp	14480	170042
Non-Tarp	0	1819655

Results

When making a final selection and recommendation of what model to use we begin by evaluating the CV ROC of all the models. Figure 22 shows the ROC curves with Table 3 showing the AUC metric for each model. All of the ROC curves show excellent performance, which is confirmed by the AUC values. Every model has a very high AUC value with the lowest being the LDA at .9888 and the highest being the Polynomial SVM at .9996 (all within about 1% of each other).

Table 4 shows the other metrics for each model on the training dataset at a given threshold (refer to the previous section for a discussion of threshold selection). All of the models perform similarly with KNN having the highest sensitivity and

Polynomial SVM having the highest j-index. The j-index ranges from 0.8648 for the LDA to 0.9876 for the KNN model. The sensitivity ranges from 0.9717 for the QDA to 0.9952 for the KNN model.

Next, we evaluate the ROC curves of all the models on the holdout dataset. Figure 23 shows the ROC curves with Table 5 showing the AUC metric for each model. The penalized logistic regression has the highest AUC at 0.9996 with the polynomial SVM coming in a close second with 0.9995. Every model has a very high AUC value. The KNN trails the other models at 0.9643, whereas all of the other models have values above 0.99.

Table 5 shows all metrics for each model on the holdout dataset at the chosen threshold (refer to the previous section for a discussion of threshold selection). The j-index values range from 0.8964 for penalized logistic regression to 0.9487 for the polynomial SVM. We noticed that three different models -- penalized logistic, polynomial SVM, and radial SVM had a true positive rate of 100%. The logistic regression and the linear SVM are both 0.9999. The KNN had the lowest sensitivity at 0.9290.

Finally, we evaluated the models for over and underfitting. Figure 25 shows the results of this analysis. None of the models show a large difference between the training and the holdout datasets with respect to j-index or ROC AUC. The XGBoost model and the KNN model have the largest discrepancy between the cross validation and holdout AUCs. Even though the objective loss was only 3% these two models were top performers based on the CV metrics and the other models did not drop off as much. In conjunction these two observations indicate that the XGBoost and KNN models may be overfitted.

Table 3: Comparison of cross-validated AUC for all models

model	roc_auc
KNN	0.9943
LDA	0.9888
QDA	0.9982
logreg	0.9985
penalized log	0.9986
XGBoost	0.9993
linear SVM	0.9980
polynomial SVM	0.9996
radial SVM	0.9995

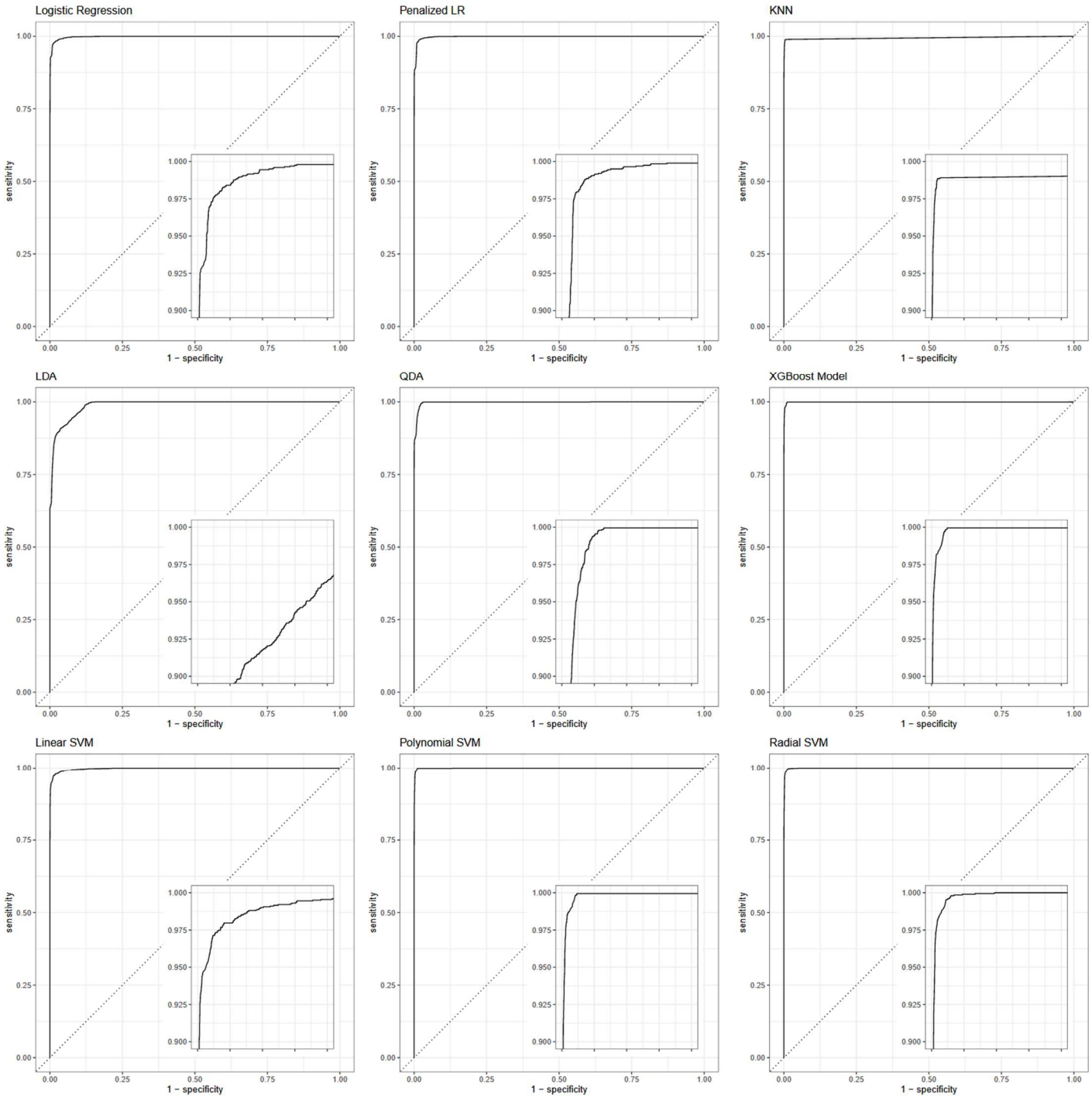

Figure 22: CV ROC plots for all models

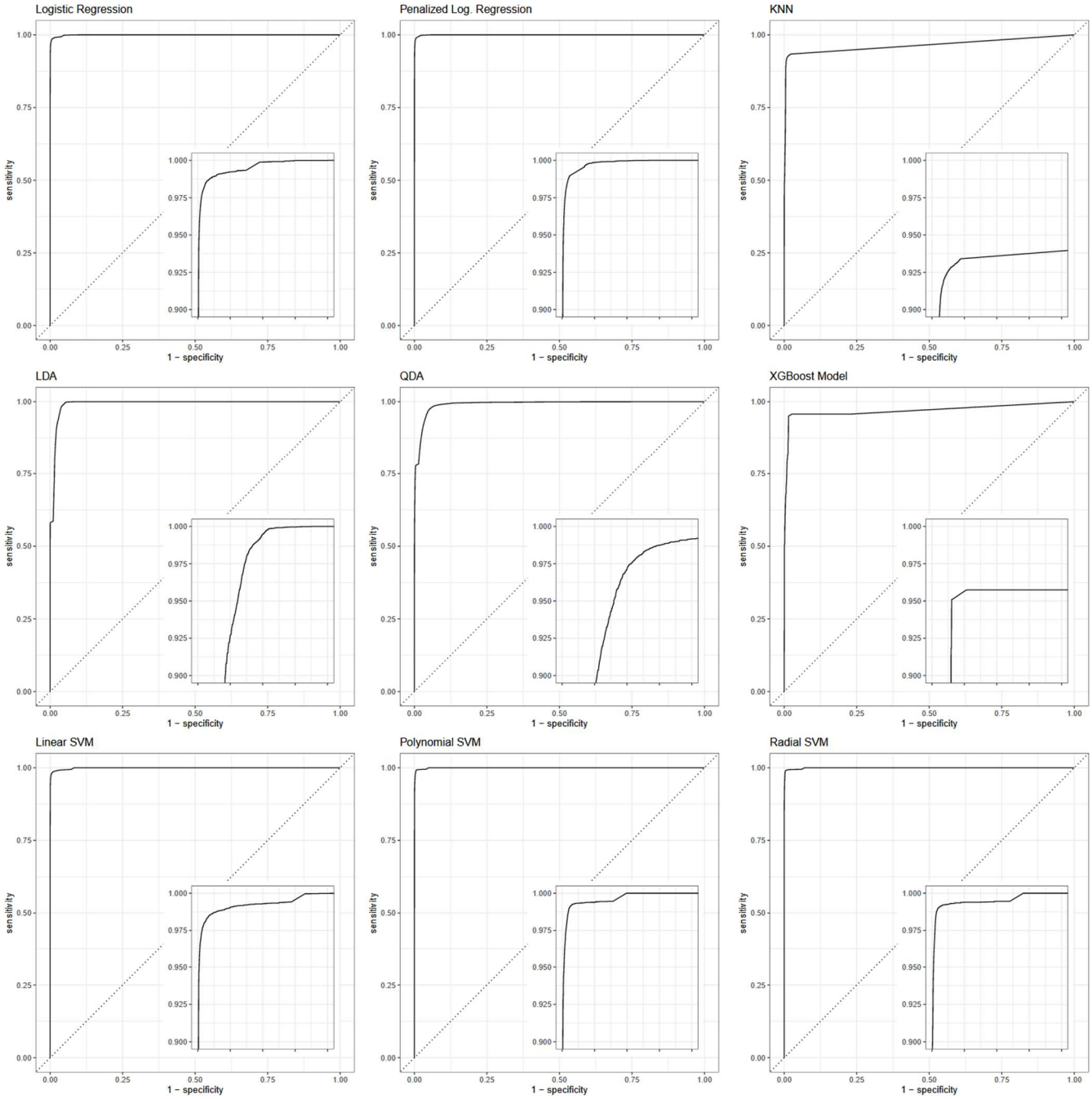

Figure 23: Holdout ROC plots for all models

Table 4: Comparison of Threshold Metrics for all models

model	j_index	specificity	sensitivity	accuracy	precision	threshold
KNN	0.9839	0.9952	0.9886	0.9950	0.8729	0.090
LDA	0.8648	0.9790	0.8858	0.9760	0.5824	0.010
QDA	0.9692	0.9717	0.9975	0.9725	0.5380	0.015
logreg	0.9634	0.9872	0.9763	0.9868	0.7152	0.050
penalized log	0.9685	0.9843	0.9842	0.9843	0.6743	0.040
XGBoost	0.9873	0.9898	0.9975	0.9900	0.7631	0.040
linear SVM	0.9594	0.9861	0.9733	0.9857	0.6981	0.060
polynomial SVM	0.9876	0.9896	0.9980	0.9899	0.7606	0.030
radial SVM	0.9833	0.9872	0.9960	0.9875	0.7206	0.020

Table 5: Comparison of Metrics on Holdout Dataset for all models

model	roc_auc	j_index	specificity	sensitivity	accuracy	precision
KNN	0.9643	0.9132	0.9842	0.9290	0.9838	0.2998
LDA	0.9921	0.9252	0.9680	0.9572	0.9679	0.1789
QDA	0.9915	0.9022	0.9639	0.9383	0.9637	0.1591
logreg	0.9994	0.9027	0.9028	0.9999	0.9035	0.0696
penalized log	0.9997	0.8964	0.8964	1.0000	0.8971	0.0656
XGBoost	0.9700	0.9250	0.9676	0.9574	0.9675	0.1769
linear SVM	0.9991	0.8969	0.8970	0.9999	0.8978	0.0660
polynomial SVM	0.9995	0.9487	0.9487	1.0000	0.9491	0.1243
radial SVM	0.9994	0.9145	0.9145	1.0000	0.9152	0.0785

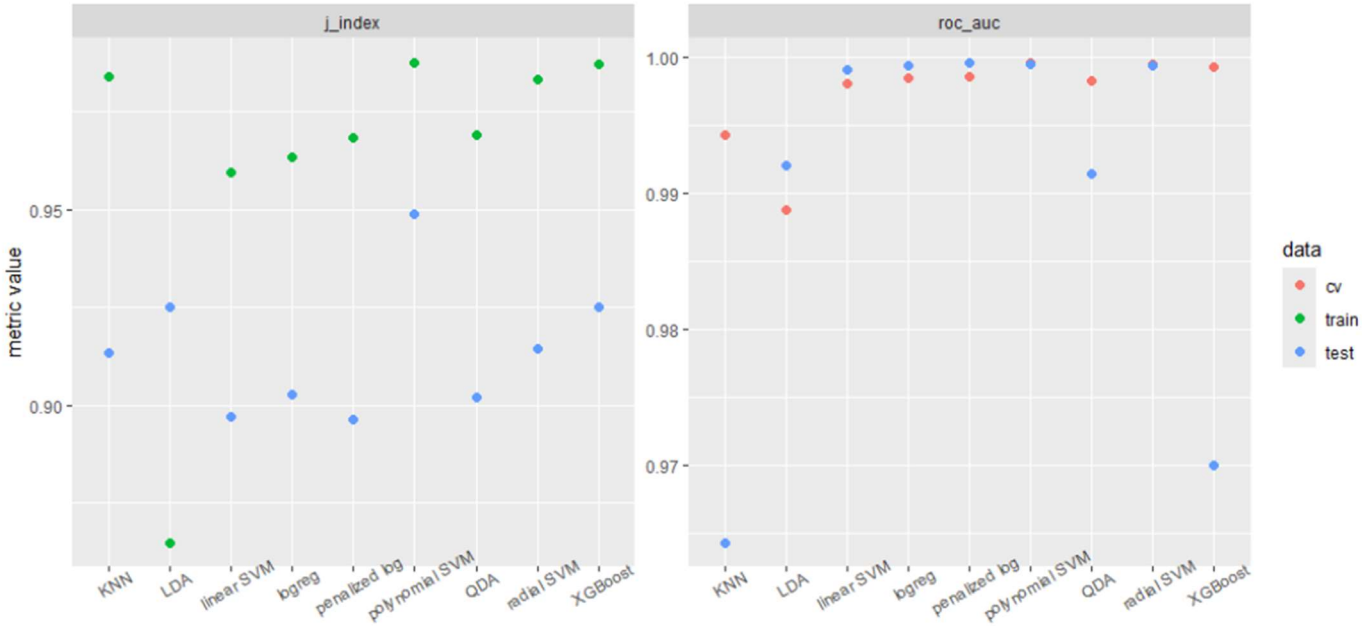

Figure 24: Plot showing J Index and ROC AUC Values of all plots to compare for potential overfitting

Conclusion

Conclusion 1

We recommend the polynomial support vector machine as the algorithm to use for the detection of blue tarps. We found that this method results in the best algorithm that classifies each pixel as blue tarp pixel or a non-tarp pixel after the performance metrics for each model were assessed on the holdout set. The polynomial support vector machine model had the best performance or close to the best performance for multiple metrics such as AUC, J-index, and sensitivity. While it is important to consider all metrics when determining the best model, in this context, it is most important to identify the majority of the tarp pixels and limit the number of false negatives so that we can locate most of the displaced persons. The sensitivity and j-index were the metrics that we mainly focused on with this goal in mind. The polynomial support vector machine model had the highest J-index and one of the highest sensitivity values when tested using the hold-out set. For these reasons, we believe the polynomial support vector machine model will be most useful and accurate in identifying blue tarps. This model will play a major role in providing resources to displaced persons and potentially saving human lives.

Conclusion 2

The models that performed the best in cross-validation were the polynomial support vector machine model, the radial support vector machine model, and the XGBoost model. The polynomial support vector machine model had the highest AUC, followed by the radial support vector machine model, and then the XGBoost model. Their AUCs were very high, but they were not much higher than the AUCs for the other models with the lowest being an AUC of 0.9888 for LDA. While all models perform well on the data, the best performing models indicate that the data benefits from some non-linear flexibility and separation when trained and tested with the training data. This is demonstrated by the polynomial support vector machine performing better than the linear algorithms, such as the LDA model and the linear support vector model. However, even though these non-linear models perform better, it is important to keep in mind that they don't perform substantially better and the linear models still have a great performance based on AUC. This could potentially mean that while the data requires some separation flexibility, it may not require a large amount of flexibility that other models may provide, especially when it comes to the hold-out data.

The models that performed the best on the hold-out set based on AUC were the penalized logistic regression model and the polynomial support vector machine model. The AUCs of these models were very close at 0.9997 and 0.9995, respectively. It is important to note that the polynomial support vector machine model also had one of the best performances in cross-validation on the training set, which tells us it may be one of the best models for this data. The model that performed the worse was the KNN model with an AUC of 0.9643. Again, all models performed very well, but this time flexibility was not as valuable. The high AUC of the penalized logistic regression model revealed that the hold-out set potentially required less flexibility than the training set. When J-index was considered, the polynomial support vector machine model continued to be the best model with the LDA model following behind. This further demonstrates that algorithms with some flexibility perform best on the hold-out set, however it is a bit contradictory that a non-linear model and a linear model performed the best here. Based on these results, we could see that it would be important for us to consider a few metrics when deciding on our recommended model. In terms of sensitivity, the penalized logistic regression model, the polynomial support vector machine model, and the radial support vector machine model all performed the best, which further instilled the idea that some flexibility and non-linearity is valuable, but only to a certain extent.

Conclusion 3

The goal of this modeling is ultimately to assist in predicting blue tarps to assist humanitarian efforts in the result of a natural disaster. With that in mind, one area that presented a fascinating part of follow on research was the threshold selection. We ultimately selected the model with an optimal J-index but for any sort of model selection coordinating with a response team is crucial to determining a final threshold. Natural disasters often necessitate extremely fast response time as the first 24 hours are vital in saving lives. With that in mind, we asserted that resources are somewhat limited and balancing false positives and false negatives was more valuable than committing to reducing false negatives. In real world scenarios, developing a model in conjunction with subject matter experts is integral to the process in ensuring that effective prioritization of metrics is done to align with the objective of the analysis.

Conclusion 4

We found our results to be compatible because the same type of decision boundaries performed similarly across models. The performance of each of the models would be impacted by the shape of the data, and is especially apparent in the comparison of the LDA and QDA models. Even though the performance on the test dataset was comparable between the two, the QDA suffered more loss in j-index and ROC AUC between the training and holdout sets, while the LDA actually improved its performance metrics on the test set. This indicates that the QDA may have slightly overfit the training set in comparison to the LDA, and the true shape of the data may in fact fit a linear decision boundary better. In addition, all of the linear models (linear SVM and tuned/untuned logistic regression) had near-perfect ROC curves. The fact that the KNN and boosted models appeared overfitted and had the lowest AUCs on the holdout dataset indicates further that the assumptions of the other models are in fact true and the decreased bias error of these models does not make up for the variance error introduced by such a flexible method. However, the linear models suffered from lower j-index and accuracy overall on the holdout dataset, and the fact that the radial and polynomial SVMs performed better indicates that the boundary may be more complicated than just linear.

Conclusion

Throughout this process, we discovered that with larger data sets, some trade-offs are augmented and have larger impacts on results. The trade-off that had the most impact for us was the trade-off between computational time and parameter tuning. The more parameters we tuned or the more iterations we set for each model, the longer the computational time. In emergency situations like this, timely aid is extremely important, and sometimes predictive accuracy may need to be decreased in order to maximize the use of time and resources. Furthermore, sometimes it may be best to tune only one specific parameter depending on the context of the data. When training these algorithms, we needed to carefully choose our tuned parameters so that the computational time was not substantially long. As much as we wanted to tune many parameters, this was not realistic and we had to choose less tuning parameters in order to use our time efficiently and minimize computational time. We needed to find a balance between the amount of time it would take and the number of false positives and false negatives we would have to deal with. We kept these goals in mind when deciding on which model we would recommend.

Conclusion 6

As was mentioned before in this report, sensitivity has been a very important performance measure in our analysis. It is essential to maximize the sensitivity (true positive rate) of our predictive models because we want to be sure to find every blue tarp, each of which represents an actual person in need of help. Also discussed was j-index, which combines the measures of sensitivity and specificity to give a more rounded picture of the model efficacy. Maintaining a high specificity metric will help us not waste resources on pixels identified as blue tarps which are actually areas of no interest. The other metrics displayed are accuracy and precision. The accuracy metric represents all correctly identified pixels. Accuracy also represents a combination of sensitivity and specificity, however it is less useful because a baseline model that predicts only non-tarps will already have a very high accuracy because of the rarity of blue tarps. Precision describes the proportion of predicted blue tarps which actually turned out to be blue tarps, and is another way of looking at how well our model avoids false positives, along with specificity. Again, this metric is a little less relevant to our analysis and, for our purposes, the relevant information is captured in the j-index.