Disaster Relief Project: Part I

Matthew Litz

Jul 06, 2021

Contents

DS 6030 | Summer 2021 | University of Virginia

Introduction

In 2010, residents of Haiti were displaced from their homes by a devastating earthquake. Survivors were known to have created makeshift tents using Blue Tarps. A team from Rochester Institute of Technology was able to collect high-altitude images of the regions of Haiti in which the shelters were captured. These images could be used to aid rescue workers in identifying the location of the survivors. However, due to the massive quantity of images taken, an algorithm is needed to quickly identify the Blue Tarp tents.

This analysis seeks to use the RGB data generated from the images and apply classification methods to identify the Blue Tarps and by extension, the survivors. A reference data set has been provided from which to train the classification algorithms on. Five different classification algorithms will be applied on the dataset and one will be recommended for use if a similar natural disaster were to strike in the future.

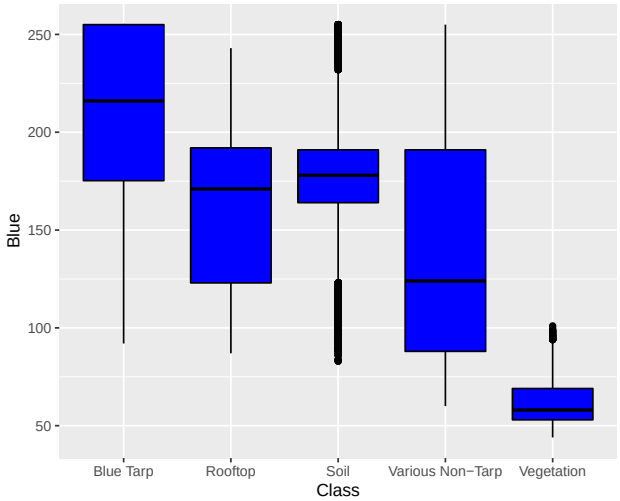
Training Data / EDA

The initial exploratory data analysis consisted of generating boxplots of the reference dataset objects (Blue Tarp, Rooftop, Soil, Various Non-Tarp, Vegetation) and the associated RGB colors. The boxplot for the color "blue" shows a clear higher median for identifying Blue Tarps, therefore producing confidence that of the 3 RGB predictors used in the classification, we appear to have at least one strong response. In addition,

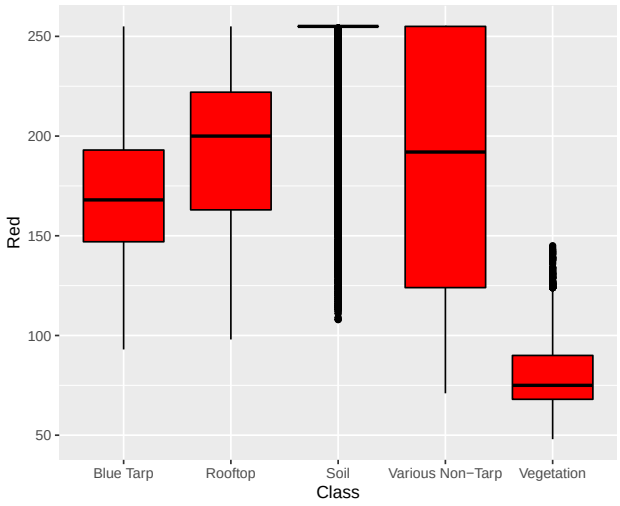
the correlation plot generated from the GGally package suggests that the color green could also have a strong predictive response.

Since we are solely interested in the Blue Tarp class, the classification models will be generated by predicting whether or not a given data set is a Blue Tarp. To achieve this, a new column was created which specifies if the image is a Blue Tarp.

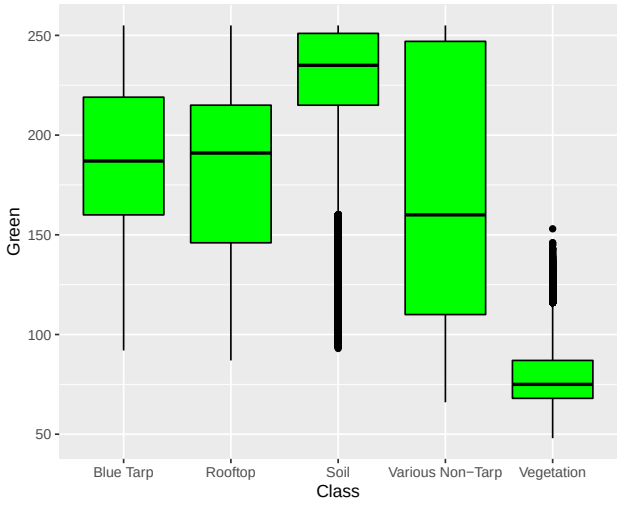
```
# Load Required Packages
library(tidyverse)
library(readr)
library(GGally)

data <- read_csv('HaitiPixels.csv')
attach(data)

data <- data %>%
  mutate(BlueTarp = if_else(Class == "Blue Tarp", "yes", "no"))

ggplot(data, aes(x=factor(Class), y=Blue)) +
  geom_boxplot(fill="blue", color="black") +
  labs(x="Class", y="Blue")
```



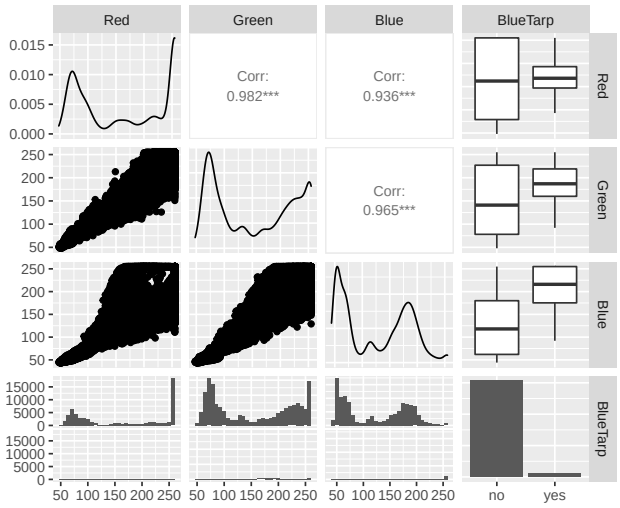

```
ggplot(data, aes(x=factor(Class), y=Red)) +
  geom_boxplot(fill="red", color="black") +
  labs(x="Class", y="Red")
```




```
ggplot(data, aes(x=factor(Class), y=Green)) +
  geom_boxplot(fill="green", color="black") +
  labs(x="Class", y="Green")
```




```
#remove classification variables
df=subset(data, select = -c(Class))

#data distributions
df %>% ggpairs(upper=list(continuous=wrap("cor", size=3)))
```



Model Training

Set-up

The training and testing data is first set up using a 60/40 split. The caret package is used extensively in this analysis. 10-fold cross validation was used for all models. The confusion matrix was generated using the "Blue Tarp" factor column in the test dataset.

```
#> [1] 37944 25297
```

Logistic Regression Results

```
trControl <- caret::trainControl(method="cv", number=10,
                                 savePredictions=TRUE,
                                 classProbs = TRUE,
                                 allowParallel=TRUE)

logitFit = train(fmla, data=df,
                 method="glm",
                 family='binomial',
                 trControl=trControl)

logitFit
```

```
#> Generalized Linear Model
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56917, 56916, 56917, 56917, 56917, 56916, ...
#> Resampling results:
#>
#>   Accuracy   Kappa
#>   0.9952879  0.9206973
```

```r
#predict values and eastablish levels
logit_pred <- predict(logitFit, X.test, type="raw")
levels(logit_pred) <- c('yes','no')

confusionMatrix(logit_pred, cm_truth)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   yes    no
#>        yes 24494   101
#>        no     31   671
#>
#>                Accuracy : 0.9948
#>                  95% CI : (0.9938, 0.9956)
#>     No Information Rate : 0.9695
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.9078
#>
#>  Mcnemar's Test P-Value : 1.905e-09
#>
#>             Sensitivity : 0.9987
#>             Specificity : 0.8692
#>          Pos Pred Value : 0.9959
#>          Neg Pred Value : 0.9558
#>              Prevalence : 0.9695
#>          Detection Rate : 0.9683
#>    Detection Prevalence : 0.9722
#>       Balanced Accuracy : 0.9340
#>
#>        'Positive' Class : yes
#>
```

Linear Discriminant Analysis (LDA) Results

```r
set.seed(125)

trControl <- caret::trainControl(method="cv", number=10,
                                 savePredictions=TRUE,
                                 classProbs = TRUE,
                                 allowParallel=TRUE)

ldaFit = caret::train(BlueTarp ~ ., data=df,
                method='lda',
                trControl=trControl)

ldaFit
```

```
#> Linear Discriminant Analysis
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'no', 'yes'
```

```
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56917, 56917, 56917, 56917, 56918, 56917, ...
#> Resampling results:
#>
#>   Accuracy   Kappa
#>   0.9839187  0.7526843
```

```
#predict values and establish levels
lda_pred <- predict(ldaFit, X.test, type="raw")
levels(lda_pred) <- c('yes','no')

confusionMatrix(lda_pred, cm_truth)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   yes    no
#>        yes 24288   168
#>        no    237   604
#>
#>                Accuracy : 0.984
#>                  95% CI : (0.9824, 0.9855)
#>     No Information Rate : 0.9695
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.7407
#>
#>  Mcnemar's Test P-Value : 0.0007276
#>
#>             Sensitivity : 0.9903
#>             Specificity : 0.7824
#>          Pos Pred Value : 0.9931
#>          Neg Pred Value : 0.7182
#>              Prevalence : 0.9695
#>          Detection Rate : 0.9601
#>    Detection Prevalence : 0.9668
#>       Balanced Accuracy : 0.8864
#>
#>        'Positive' Class : yes
#>
```

Quadratic Discriminant Analysis (QDA) Results

```
set.seed(17)

trControl <- caret::trainControl(method="cv", number=10,
                                 savePredictions=TRUE,
                                 classProbs = TRUE,
                                 allowParallel=TRUE)

qdaFit = caret::train(BlueTarp ~ ., data=df,
                method='qda',
```

```
                trControl=trControl)
qdaFit
```

```
#> Quadratic Discriminant Analysis
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56917, 56917, 56917, 56917, 56916, 56917, ...
#> Resampling results:
#>
#>   Accuracy   Kappa
#>   0.9945921  0.9056618
```

```
#predict values and establish levels
qda_pred <- predict(qdaFit, X.test, type="raw")
levels(qda_pred) <- c('yes','no')

confusionMatrix(qda_pred, cm_truth)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   yes    no
#>        yes 24517   139
#>        no      8   633
#>
#>                Accuracy : 0.9942
#>                  95% CI : (0.9932, 0.9951)
#>     No Information Rate : 0.9695
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.893
#>
#>  Mcnemar's Test P-Value : < 2.2e-16
#>
#>             Sensitivity : 0.9997
#>             Specificity : 0.8199
#>          Pos Pred Value : 0.9944
#>          Neg Pred Value : 0.9875
#>              Prevalence : 0.9695
#>          Detection Rate : 0.9692
#>    Detection Prevalence : 0.9747
#>       Balanced Accuracy : 0.9098
#>
#>        'Positive' Class : yes
#>
```

K-Nearest Neighbors (KNN) results

The ideal tuning parameter for kNN was automatically calculated through the caret package as k=7.

```
trControl <- caret::trainControl(method="cv", number=10,
                                 savePredictions=TRUE,
                                 classProbs = TRUE,
                                 allowParallel=TRUE)

knnFit = caret::train(BlueTarp ~ ., data=df,
                method='knn',
                preProcess = c("center", "scale"),
                trControl=trControl)

knnFit
```

```
#> k-Nearest Neighbors
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'no', 'yes'
#>
#> Pre-processing: centered (3), scaled (3)
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56917, 56916, 56917, 56917, 56917, 56917, ...
#> Resampling results across tuning parameters:
#>
#>   k  Accuracy   Kappa
#>   5  0.9971696  0.9543660
#>   7  0.9972170  0.9552441
#>   9  0.9971696  0.9545079
#>
#> Accuracy was used to select the optimal model using the largest value.
#> The final value used for the model was k = 7.
```

```
#predict values and establish levels
knn_pred <- predict(knnFit, X.test, type="raw")
levels(knn_pred) <- c('yes','no')

confusionMatrix(knn_pred, cm_truth)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   yes    no
#>        yes 24492    28
#>        no     33   744
#>
#>                Accuracy : 0.9976
#>                  95% CI : (0.9969, 0.9982)
#>     No Information Rate : 0.9695
#>     P-Value [Acc > NIR] : <2e-16
#>
#>                   Kappa : 0.9594
#>
#>  Mcnemar's Test P-Value : 0.6085
#>
#>             Sensitivity : 0.9987
#>             Specificity : 0.9637
```

```
#>          Pos Pred Value : 0.9989
#>          Neg Pred Value : 0.9575
#>              Prevalence : 0.9695
#>          Detection Rate : 0.9682
#>    Detection Prevalence : 0.9693
#>       Balanced Accuracy : 0.9812
#>
#>        'Positive' Class : yes
#>
```

Penalized Logistic Regression (PLR)

The final model applied was PLR. This model utilizes the glmnet model with ridge regression. The turning parameter that provides the best performance was 0.003981072.

```
lambdas = 10^seq(0, -4, by=-0.1)

trControl = caret::trainControl(method='cv',
                                number=10,
                                savePredictions=TRUE,
                                classProbs=TRUE,
                                allowParallel=TRUE)

tuneGrid = expand.grid(lambda=lambdas, alpha=0)

plrFit = caret::train(fmla, data=df,
                  method='glmnet',
                  family='binomial',
                  thresh=0.74,
                  trControl=trControl,
                  tuneGrid=tuneGrid)
plrFit
```

```
#> glmnet
#>
#> 63241 samples
#>     3 predictor
#>     2 classes: 'no', 'yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 56917, 56917, 56917, 56917, 56917, 56916, ...
#> Resampling results across tuning parameters:
#>
#>   lambda        Accuracy   Kappa
#>   0.0001000000  0.9757278  0.3799580839
#>   0.0001258925  0.9757278  0.3799580839
#>   0.0001584893  0.9757278  0.3799580839
#>   0.0001995262  0.9757278  0.3799580839
#>   0.0002511886  0.9757278  0.3799580839
#>   0.0003162278  0.9757278  0.3799580839
#>   0.0003981072  0.9757278  0.3799580839
#>   0.0005011872  0.9757278  0.3799580839
#>   0.0006309573  0.9757278  0.3799580839
#>   0.0007943282  0.9757278  0.3799580839
```

```
#>   0.0010000000  0.9757278  0.3799580839
#>   0.0012589254  0.9757278  0.3799580839
#>   0.0015848932  0.9757278  0.3799580839
#>   0.0019952623  0.9757278  0.3799580839
#>   0.0025118864  0.9757278  0.3799580839
#>   0.0031622777  0.9757278  0.3799580839
#>   0.0039810717  0.9757278  0.3799580839
#>   0.0050118723  0.9726285  0.2446742412
#>   0.0063095734  0.9695767  0.0894610590
#>   0.0079432823  0.9680429  0.0009540522
#>   0.0100000000  0.9680271  0.0000000000
#>   0.0125892541  0.9680271  0.0000000000
#>   0.0158489319  0.9680271  0.0000000000
#>   0.0199526231  0.9680271  0.0000000000
#>   0.0251188643  0.9680271  0.0000000000
#>   0.0316227766  0.9680271  0.0000000000
#>   0.0398107171  0.9680271  0.0000000000
#>   0.0501187234  0.9680271  0.0000000000
#>   0.0630957344  0.9680271  0.0000000000
#>   0.0794328235  0.9680271  0.0000000000
#>   0.1000000000  0.9680271  0.0000000000
#>   0.1258925412  0.9680271  0.0000000000
#>   0.1584893192  0.9680271  0.0000000000
#>   0.1995262315  0.9680271  0.0000000000
#>   0.2511886432  0.9680271  0.0000000000
#>   0.3162277660  0.9680271  0.0000000000
#>   0.3981071706  0.9680271  0.0000000000
#>   0.5011872336  0.9680271  0.0000000000
#>   0.6309573445  0.9680271  0.0000000000
#>   0.7943282347  0.9680271  0.0000000000
#>   1.0000000000  0.9680271  0.0000000000
#>
#> Tuning parameter 'alpha' was held constant at a value of 0
#> Accuracy was used to select the optimal model using the largest value.
#> The final values used for the model were alpha = 0 and lambda = 0.003981072.
```

```
#best tuning parameter
plrFit$bestTune
```

```
#>    alpha      lambda
#> 17     0 0.003981072
```

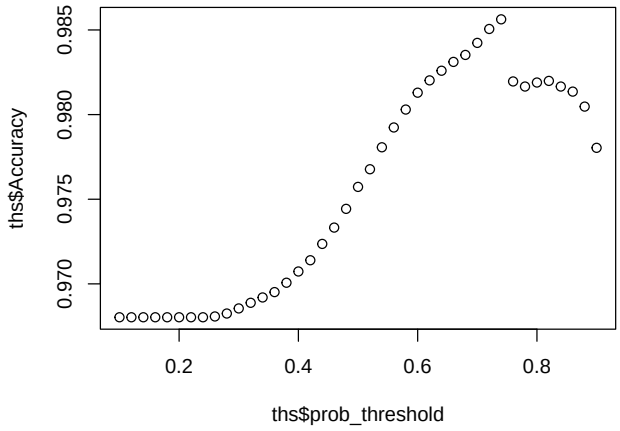
```
#predict values and establish levels
plr_pred <- predict(plrFit, X.test, type="raw")
levels(plr_pred) <- c('yes','no')

confusionMatrix(plr_pred, cm_truth)
```

```
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   yes    no
#>        yes 24525   579
#>        no      0   193
#>
```

```
#>                Accuracy : 0.9771
#>                  95% CI : (0.9752, 0.9789)
#>     No Information Rate : 0.9695
#>     P-Value [Acc > NIR] : 1.005e-13
#>
#>                   Kappa : 0.3926
#>
#>  Mcnemar's Test P-Value : < 2.2e-16
#>
#>             Sensitivity : 1.0000
#>             Specificity : 0.2500
#>          Pos Pred Value : 0.9769
#>          Neg Pred Value : 1.0000
#>              Prevalence : 0.9695
#>          Detection Rate : 0.9695
#>    Detection Prevalence : 0.9924
#>       Balanced Accuracy : 0.6250
#>
#>        'Positive' Class : yes
#>
```

Threshold Selection for Penalized Logistic Regression

The threshold that produced the highest accuracy (0.74) was back-inserted into the PLR model.



ROC Curves

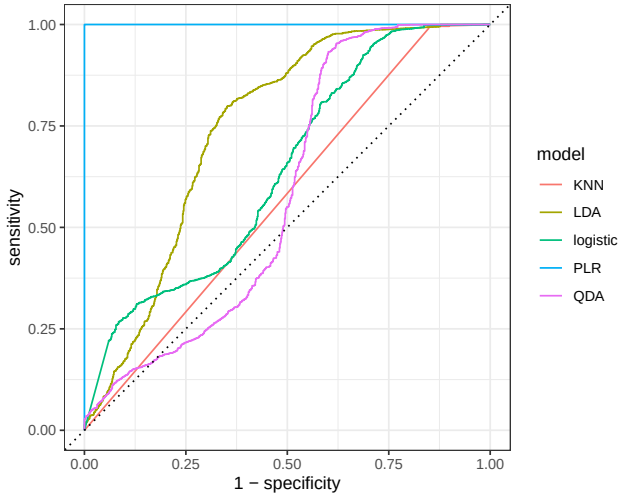
The probabilities were calculated and assembled into a tibble for all 5 models.

```r
#calculate probabilities
preds = tibble(
  logistic = predict(logitFit, X.test, type="prob")[,1],
  LDA = predict(ldaFit, newdata=X.test, type='prob')[,1],
  QDA = predict(qdaFit, newdata=X.test, type='prob')[,1],
  KNN = predict(knnFit, newdata=X.test, type='prob')[,1],
  PLR = predict(plrFit, newdata=X.test, type='prob')[,1]
)

#Y data from test data
Y.test <- test$BlueTarp
```

```
#Calculate evaluation data by which to determine ROC Curves
eval_data = preds %>%
  mutate(truth = Y.test) %>%
  pivot_longer(cols=-truth, names_to="model", values_to="score")  %>%
  rowwise() %>%
  mutate(prob = if_else(truth == "yes", 1-score, score))
```

Using the probability data, the ROC curves were calculated using the yardstick package.

```
outcome_levels = c("no", "yes")
eval_data = eval_data %>%
  mutate(truth = factor(truth, outcome_levels))


#-- Make ROC curve data
ROC_data = eval_data %>%
  group_by(model) %>%
  yardstick::roc_curve(truth,prob)

#PlotROC curves
ROC_data %>% ggplot2::autoplot()
```




```
#calculate AUC
auc = eval_data %>%
  group_by(model) %>%
  yardstick::roc_auc(truth, prob)
auc
```

```
#> # A tibble: 5 x 4
#>   model    .metric .estimator .estimate
#>   <chr>    <chr>   <chr>          <dbl>
#> 1 KNN      roc_auc binary         0.571
#> 2 LDA      roc_auc binary         0.744
#> 3 logistic roc_auc binary         0.641
#> 4 PLR      roc_auc binary         1
#> 5 QDA      roc_auc binary         0.585
```

Results (Cross-Validation)

The following table presents a table of key metrics collected during this analysis:

Model	Tuning	AUROC	Threshold	Accuracy	TPR	FPR	Precision
Log Reg	N/A	0.6496648	N/A	0.995	0.9986	0.1165	0.9963
LDA	N/A	0.7714629	N/A	0.985	0.9907	0.1933	0.9983
QDA	N/A	0.6103066	N/A	0.9949	0.9996	0.1498	0.9952
KNN	N/A	0.5571301	N/A	0.9977	0.9985	0.0282	0.9991
Penalized Log Reg	0.003981072	1.0000000	0.74	0.9757	1	0.7875	0.9755

Conclusions

Judging the methods by the AUC, which provides a cumulative measure of performance, PLR is the superior classification method for this application. Of the 5 classification methods tested, only PLR and LDA are recommended for use in future applications.

Penalized Linear Regression is the recommended method for identifying survivors in Blue Tarps. This is further substantiated by its AUC value of 1. This method has shifted potential false negatives to false positives and true negatives when compared to the other methods. The test data results show the PLR never classifies a Blue Tarp incorrectly, which is critical for search and rescue operations.