Disaster-Relief-Project-Part-1

Mahin Ganesan, Wyatt Priddy, and Taylor Tucker

2024-03-12

Introduction

In the aftermath of a devastating earthquake that rattled Haiti, displaced people are living in makeshift shelters awaiting support from aid workers. The aid workers, mainly from the United States military, are trying to reach the dispersed camps. With communication lines being down and the terrain being impassable, there are challenges in providing relief in a time- sensitive manner.

It is known that the makeshift shelters are largely constructed with blue tarps, therefore the Rochester Institute of Technology deployed airplanes to collect high resolution geo-referenced imagery. This will help us identify where displaced persons are based on the tarps.

To determine where to allocate aid, we need to use data mining against the thousands of photos taken each day which human eyes can not efficiently filter through. Determining the location in a timely manner will be paramount to rendering aid successfully and saving human life.

The primary aim of this experiment is to evaluate the efficacy of various classification algorithms in accurately and promptly identifying the presence of makeshift shelters and, by extension, the displaced persons residing within them. By harnessing the power of machine learning and image analysis, our objective is to develop a robust algorithm capable of rapidly scanning through the imagery data, pinpointing areas of interest, and facilitating timely intervention by rescue teams.

To facilitate efficient rescue efforts, we will need to determine a threshold where the number of false positives is minimal. This may not necessarily be the model that has the highest performance in accuracy but rather the highest performance with precision, or the proportion of true positives that are correctly identified by the model out of all true positives and false positive. If there are additional aid resources after rescuing all persons identified from our models, we can loosen our model specifications to classify more objects as blue tarps in an attempt to expand our search efforts.

Data

Team members have assisted our mission by classifying training data consisting of 63,241 data points for our investigation. There are 5 classifications that have been assigned to the pixel level data:

1. Blue Tarp
2. Rooftop
3. Soil
4. Various Non-Tarp
5. Vegetation

As expected when trying to find a needle in a haystack, our representation of misplaced persons (blue tarps) makes up only a small portion of the data set. Just 3.2% of the total data set, or 2022 records, are classified

as `blue tarp`. After inspecting the average color of the, it becomes apparent that even though blue tarps are a small section of the data their distinction in color should set them apart.

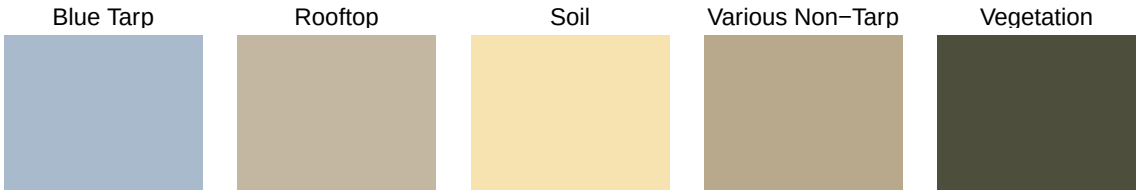


Figure 1: Average Color of Class

Vegetation and soil cover over 73% of the pictures as to be expected of pictures that largely will include countryside.

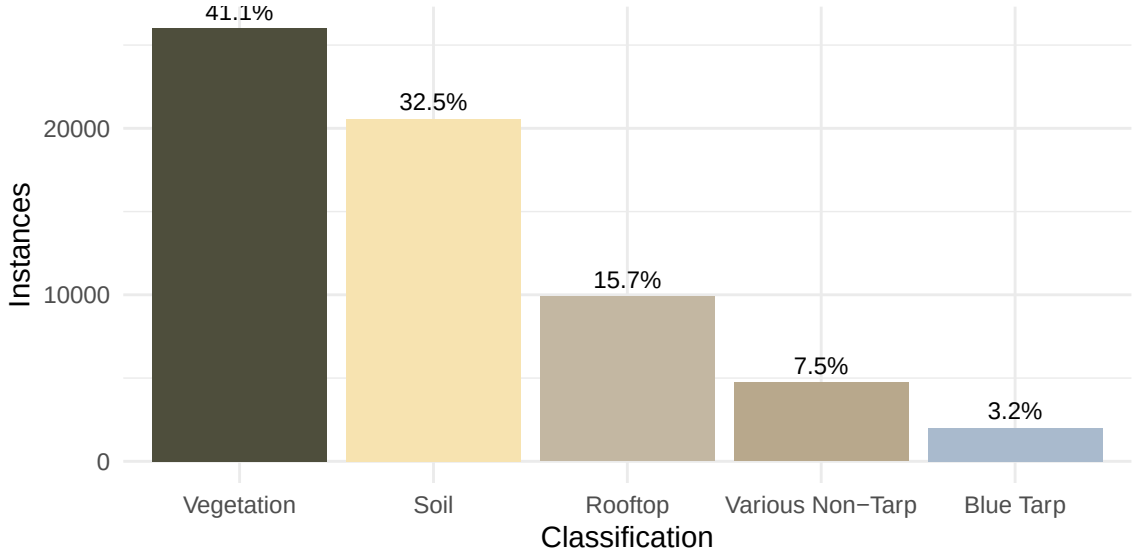


Figure 2: Distribution of Classifications in Training Set

Description of Methodology

Given that our sample data set only has 3.2% of the target class, we will use stratification to account for the imbalance in our test and train split. We are looking to ensure that each there is even representation between our two sets to ensure a robust analysis when training our models and subsequently predicting on the test data.

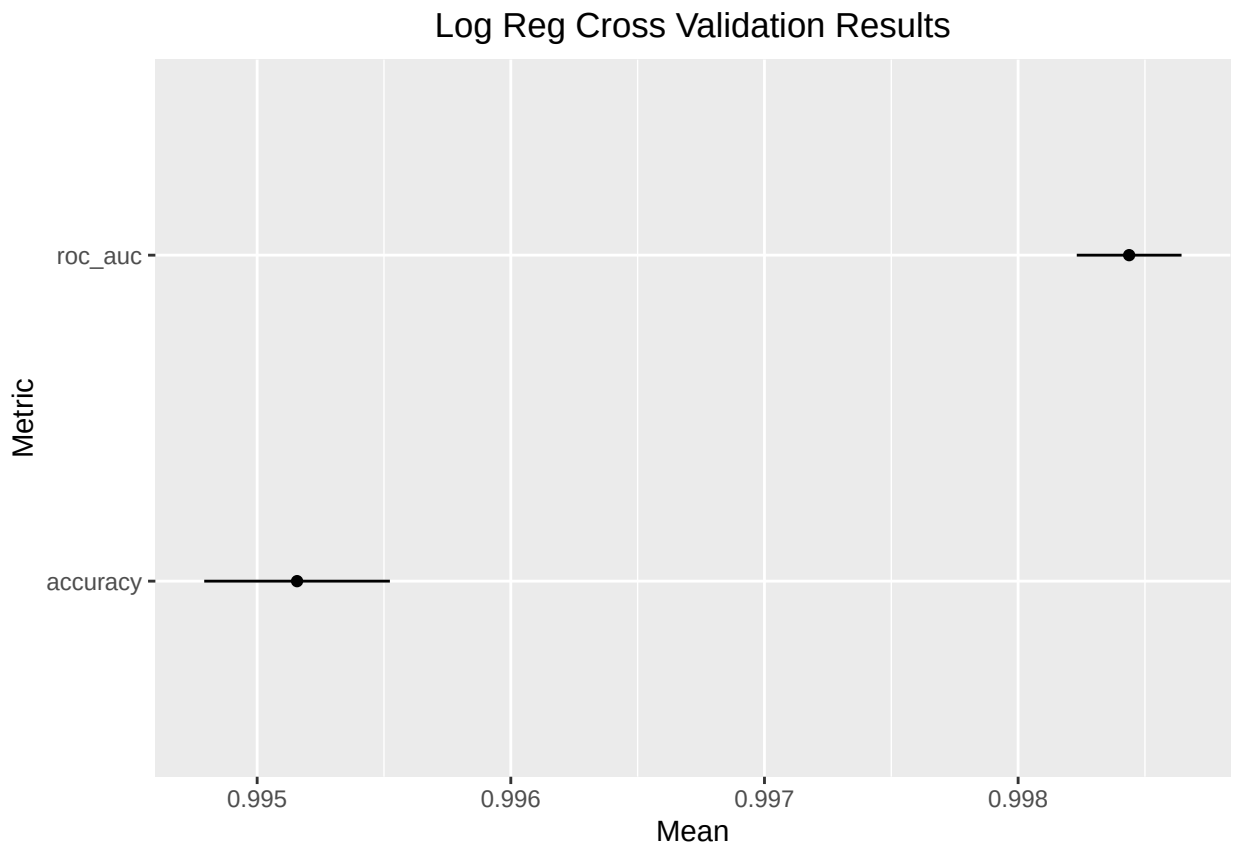
Each model is being trained with 10-fold cross-validation to ensure the models are generalizing well and stable over the data set rather than performing well on a single subsection of the train/test split.

Libraries used: - tidymodels - tidyverse - probably

Results

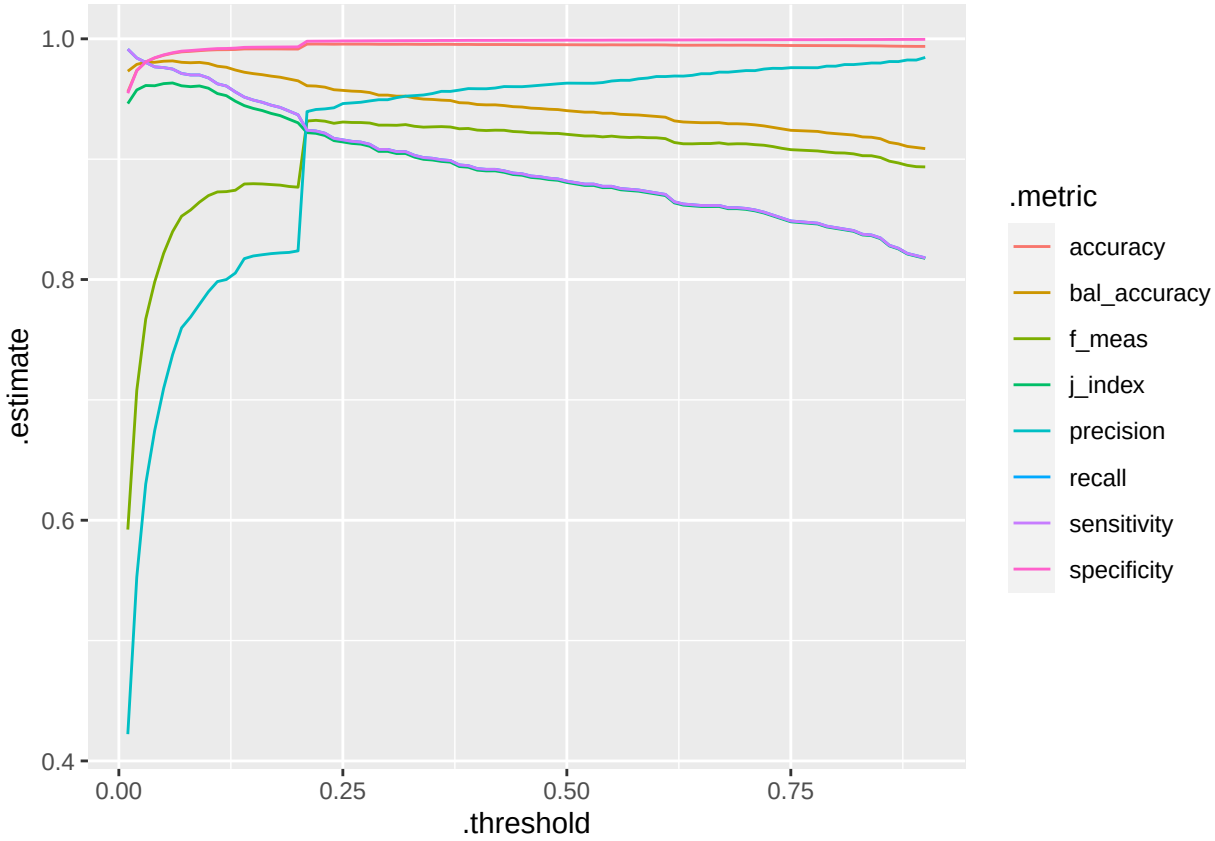
Logistic Regression Model

A




```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```




```
## # A tibble: 8 x 3
##   .metric      .estimator .estimate
##   <chr>        <chr>          <dbl>
## 1 j_index      binary         0.909
## 2 specificity  binary         0.910
## 3 sensitivity  binary         0.999
## 4 accuracy     binary         0.996
## 5 f_meas       binary         0.998
## 6 precision    binary         0.997
## 7 recall       binary         0.999
## 8 bal_accuracy binary         0.955
```

```
## # A tibble: 8 x 3
##   .metric      .estimator .estimate
##   <chr>        <chr>          <dbl>
## 1 j_index      binary         0.948
## 2 specificity  binary         0.995
## 3 sensitivity  binary         0.954
## 4 accuracy     binary         0.955
## 5 f_meas       binary         0.976
## 6 precision    binary         1.00
## 7 recall       binary         0.954
## 8 bal_accuracy binary         0.974
```

Conclusion

TEXT

Appendix

```
knitr::opts_chunk$set(echo=FALSE)
knitr::opts_chunk$set(cache=TRUE, autodep=TRUE)
knitr::opts_chunk$set(fig.align="center", fig.pos="H")
# Set up Parallel Processing
library(doParallel)

cl <- makePSOCKcluster(parallel::detectCores(logical = FALSE))

registerDoParallel(cl)

# Load Libraries
library(tidyverse)
library(tidymodels)
library(probably)

# Read in Data
haiti <- read_csv('https://gedeck.github.io/DS-6030/project/HaitiPixels.csv', show_col_types=FALSE) %>%
            mutate(BlueTarp= factor(ifelse(Class=="Blue Tarp", "Yes", "No"),labels=c("No", "Yes")))

# View Average Color of Each Class
haiti %>%
  group_by(Class) %>%
    summarize(R = mean(Red),
              G = mean(Green),
              B = mean(Blue))%>%
    mutate(hex = rgb(R, G, B, maxColorValue = 255))%>%
    ggplot(aes(x = 1, y = 1, fill = hex)) +
  geom_tile() +
  scale_fill_identity() +
  theme_void() +
  facet_wrap(~Class, ncol=5)

# Show different classifications in data set
haiti %>%
  group_by(Class) %>%
  summarize(count=n()) %>%
  mutate(percent_of_total = sprintf('%.1f%%', count / sum(count) * 100)) %>%
    ggplot(aes(x = reorder(Class, -count), y = count, fill=Class)) +
    geom_bar(stat = "identity") +
    scale_fill_manual(values=c('#A9BACD', '#C3B7A2' , '#F7E3B0','#B8A88C', '#4E4E3C')) +
    geom_text(aes(label = percent_of_total), vjust = -0.5, size = 3) +
    labs(x = "Classification", y = "Instances") +
    theme_minimal()+
    guides(fill = "none")
```

```r
# Set seed
set.seed(81718)

# Create initial split for 80/20 with stratified sampling on Hazardous
haiti_split <- initial_split(haiti, prop=.8, strat=BlueTarp)

# Create training data set
train_data <- training(haiti_split)

# Create test data set
test_data <- testing(haiti_split)

# Set up 10-fold cross-validation
resamples <- vfold_cv(train_data, v=10, strata=BlueTarp)

# Set settings for control resamples
cv_control <- control_resamples(save_pred=TRUE)

# Define performance metrics
performance_metrics <- metric_set(j_index, specificity, sensitivity, accuracy, f_meas, precision, recall


get_ROC_plot <- function(model, train_data, test_data, model_name){
  # Augment train and test data with predicted probabilities
  roc_train <- augment(model, train_data) %>%
    roc_curve(truth = BlueTarp, .pred_Yes, event_level = "second") %>%
    mutate(Dataset = "Train")

  roc_test <- augment(model, test_data) %>%
    roc_curve(truth = BlueTarp, .pred_Yes, event_level = "second") %>%
    mutate(Dataset = "Test")

  # Combine train and test ROC curve data
  roc_data <- bind_rows(roc_train, roc_test)

    # Plot ROC curves for train and test data with different colors
  autoplot(roc_data) +
    geom_line(aes(x = 1 - specificity, y = sensitivity, color = Dataset))+
    labs(title = model_name, x = "False Positive Rate", y = "True Positive Rate")
}

# Create Function to Visual Train Metrics
visualize_training <- function(fit_resample_results, title){

    aggregate_metrics <-  bind_rows(logreg_fit_cv$.metrics) %>%
          group_by(.metric) %>%
          summarize(Mean = mean(.estimate),
                    std_err = sd(.estimate) / sqrt(n()))%>%
          rename(Metric=.metric)

    aggregate_metrics %>%
      ggplot(aes(x=Mean, y=Metric, xmin=Mean-std_err, xmax=Mean+std_err)) +
      geom_point() +
```

```
      geom_linerange() +
      ggtitle("Log Reg Cross Validation Results") +
      theme(plot.title = element_text(hjust = 0.5))
}


# Test Thresholds
performance_func_1 <- function(model, data){
                    threshold_perf(model %>% augment(train_data),
                                   BlueTarp,
                                   .pred_Yes,
                                   thresholds = seq(0.01, 0.9, 0.01), event_level="second",
                                   metrics=performance_metrics)
}

# Pick best J-Index as Threshold
max_precision <- function(performance_data){
          performance_data %>%
              filter(.metric == 'recall') %>%
              filter(.estimate == max(.estimate))
}


# Create Formula
formula <- BlueTarp ~ Red + Green + Blue

# Create Recipe
rec <- recipe(formula, data=train_data) %>%
    step_normalize(all_numeric_predictors())

# Create Log Model
logreg_model <- logistic_reg() %>%
    set_engine("glm") %>%
    set_mode("classification")

# define and execute the cross-validation workflow
logreg_wf <- workflow() %>%
    add_model(logreg_model) %>%
    add_recipe(rec)

# Cross Validate Model
logreg_fit_cv <- logreg_wf %>%
                     fit_resamples(resamples=resamples, control=cv_control)




# Visualize logReg Fit
visualize_training(logreg_fit_cv)

# Fit Model
logreg_model <- logreg_wf %>% fit(train_data)
```

```r
# Get Performance Thresholds
threshold_performance <- performance_func_1(logreg_wf%>%fit(train_data))

# Visualize Performance Thresholds
threshold_performance%>%
      ggplot(aes(x = .threshold, y = .estimate, color=.metric))+
      geom_line()

# Run Model on Test Data
logreg_results <- logreg_model %>% augment(test_data)

# Change Pred Class metric based on threshold testing
logreg_results$.threshold_pred_class <- as.factor(ifelse(logreg_results$.pred_Yes >= min(max_precision(t

# View results before threshold
performance_metrics(logreg_results, truth=BlueTarp, estimate=.pred_class)

# View Results After implementing threshold
performance_metrics(logreg_results, truth=BlueTarp, estimate=.threshold_pred_class)
```