Logistic Regression

Introduction

We are interested in trying a logistic regression approach to predicting which data points are part of the land engraved area (LEA), and which are part of the groove engraved area (GEA) in our 3D bullet land scans.

This is primarily a two-class classification problem. We will begin with logistic regression, and move to more sophisticated data processing or modeling as needed.

Current features of the data

Each land has been averaged across ten crosscuts, as well as shifted down so the lowest observed `value` is at 0; this column is referred to as `value_std`.

For each land, the residuals from both a robust linear model (2nd order) and a robust LOESS model have been saved.

Additional feature creation

We can define two additional columns, `depth` and `side`. `depth` represents the depth of each observed data point from the median observed `y` value. `side` represents whether the data point is to the left of the median or to the right of the median.

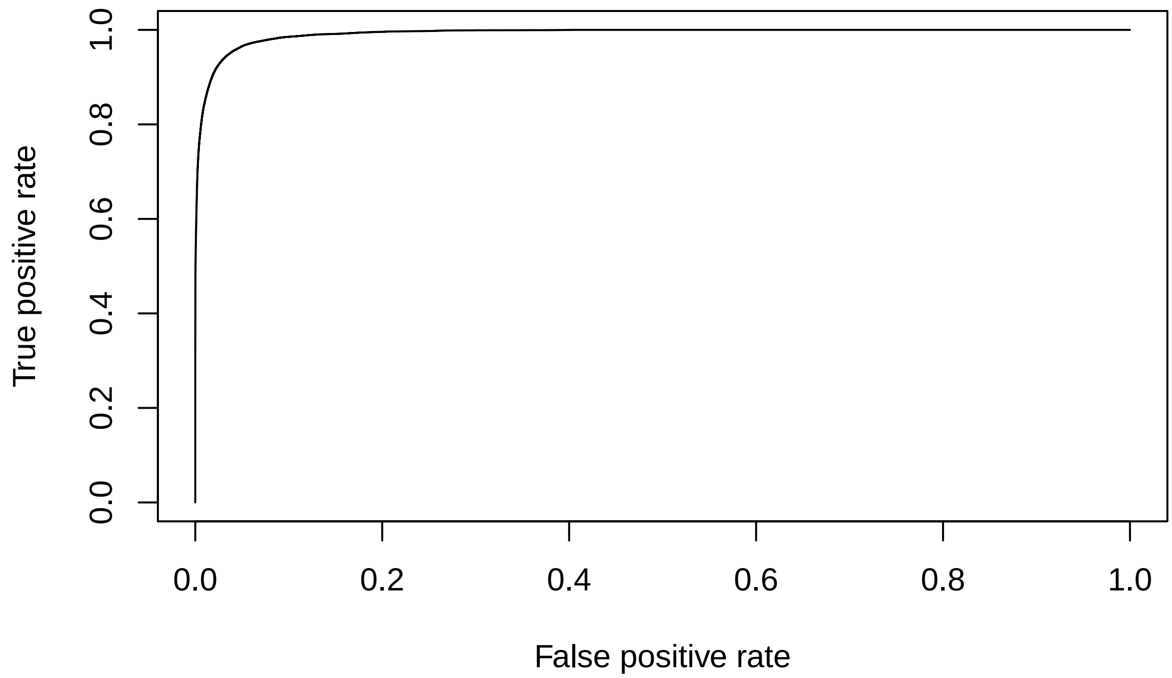
We also need to define a response variable to work with; here, we will take the manually identified `grooves` value from the `hamby44` dataset, and classify anything outside of this range as a response: 1, and anything inside this range a response: 0. This is to indicate that if the response is 1, that data point lies in the groove engraved area.

Modeling

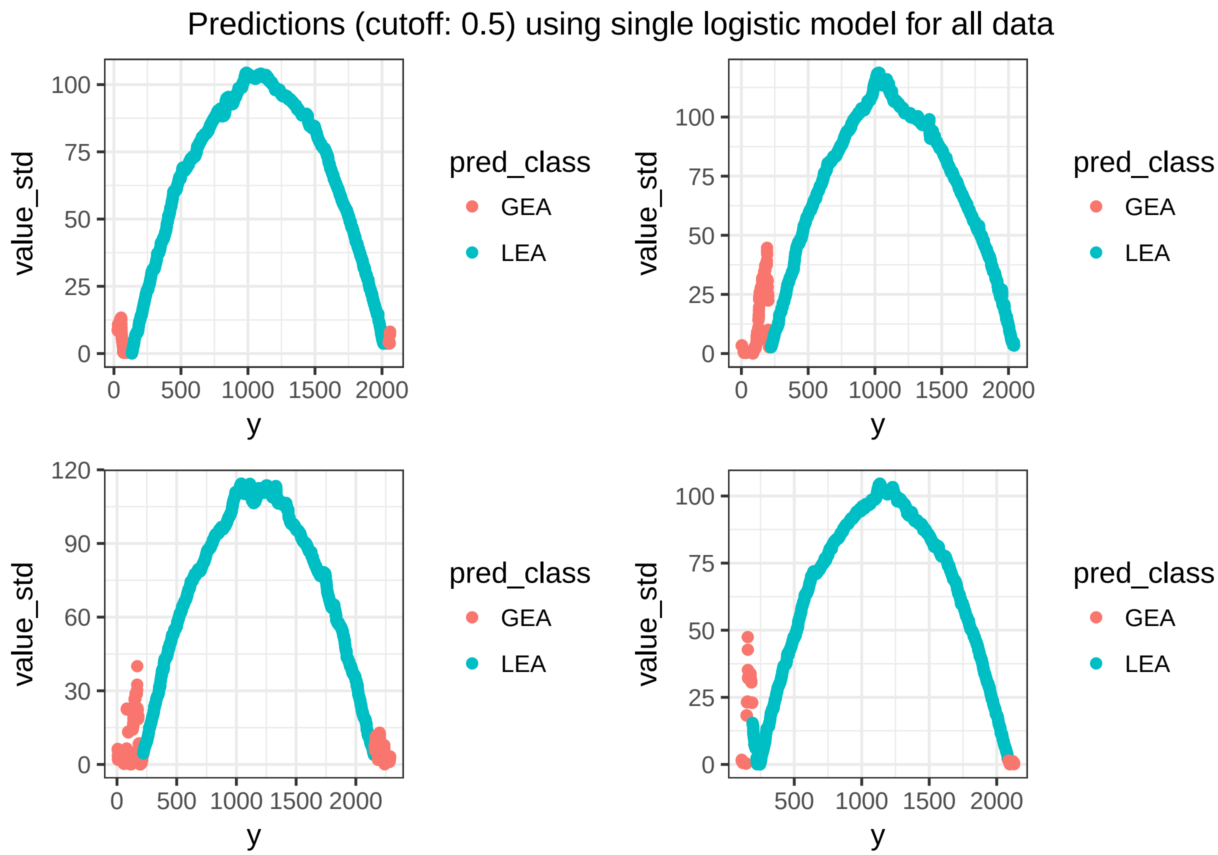
Now we are going to use glmnet to do a logistic regression.

```
##
## Call:
## glm(formula = response ~ rlo_resid + side + depth + side * depth,
##     family = "binomial", data = bullet.log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5930  -0.0189  -0.0004   0.0000   4.9034
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -3.141e+01  2.417e-01 -129.96   <2e-16 ***
## rlo_resid        7.415e-02  1.221e-03   60.70   <2e-16 ***
## sideright        4.585e+00  3.202e-01   14.32   <2e-16 ***
## depth            3.256e-02  2.551e-04  127.64   <2e-16 ***
## sideright:depth -5.536e-03  3.357e-04  -16.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 480451  on 691929  degrees of freedom
## Residual deviance:  93284  on 691925  degrees of freedom
## AIC: 93294
##
## Number of Fisher Scoring iterations: 10
```




```
## [1] 0.9923314
```



Traditional logistic regression (using `glm`) returns P-values equivalently 1 when dealing with a single bullet, which suggests we are overfitting with all the parameters included. Thus, we will use `glmnet` to do a ten-fold cross-validation of penalized logistic regression (LASSO).

However, it is important to note that the model fit to all data simultaneously has values equivalently 0, but seems to do a fairly good job of predicting locations (see above image).

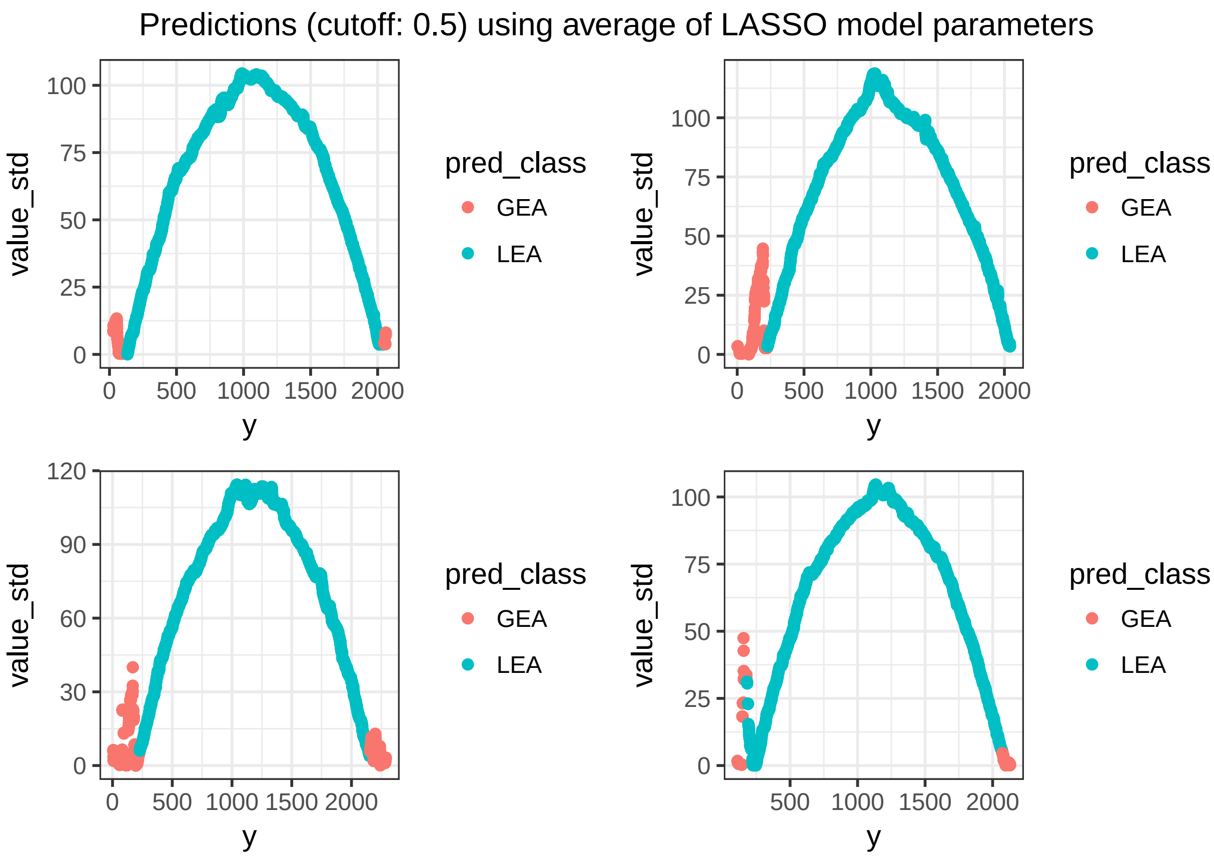
First, we are going to fit an individual model to each of the bullet LEA's we have in the Hamby44 set, and average the parameter values from each of them (Dr. Hofmann's initial suggestion).

```
hamby44 <- hamby44 %>% mutate(glmnet_fits = purrr::map(ccdata_w_resid, .f = function(bullet){
  bullet.model <- bullet[!is.na(bullet$rlo_resid),]
  X <- model.matrix( ~ rlo_resid + side + depth + side*depth - 1, bullet.model)

  # L1 regularized logistic regression
  fit <- cv.glmnet(x = X, y = bullet.model$response, family = 'binomial', type.measure = 'class', alpha
  return(fit)
}), matrix_fits = purrr::map(glmnet_fits, .f = function(fits){
  fits <- as.matrix(coef(fits))
}))

model_avg <- apply(simplify2array(hamby44$matrix_fits), 1, mean)
model_avg
```

```
##     (Intercept)       rlo_resid        sideleft       sideright
##   -1.208364e+02    1.195118e-01    2.171256e+00   -1.254012e-02
##           depth sideright:depth
##    1.238682e-01   -1.788110e-04
```

Now, we want to use the averaged logistic regression parameters to fit the model to all of the bullets.

Some examples of this are below:

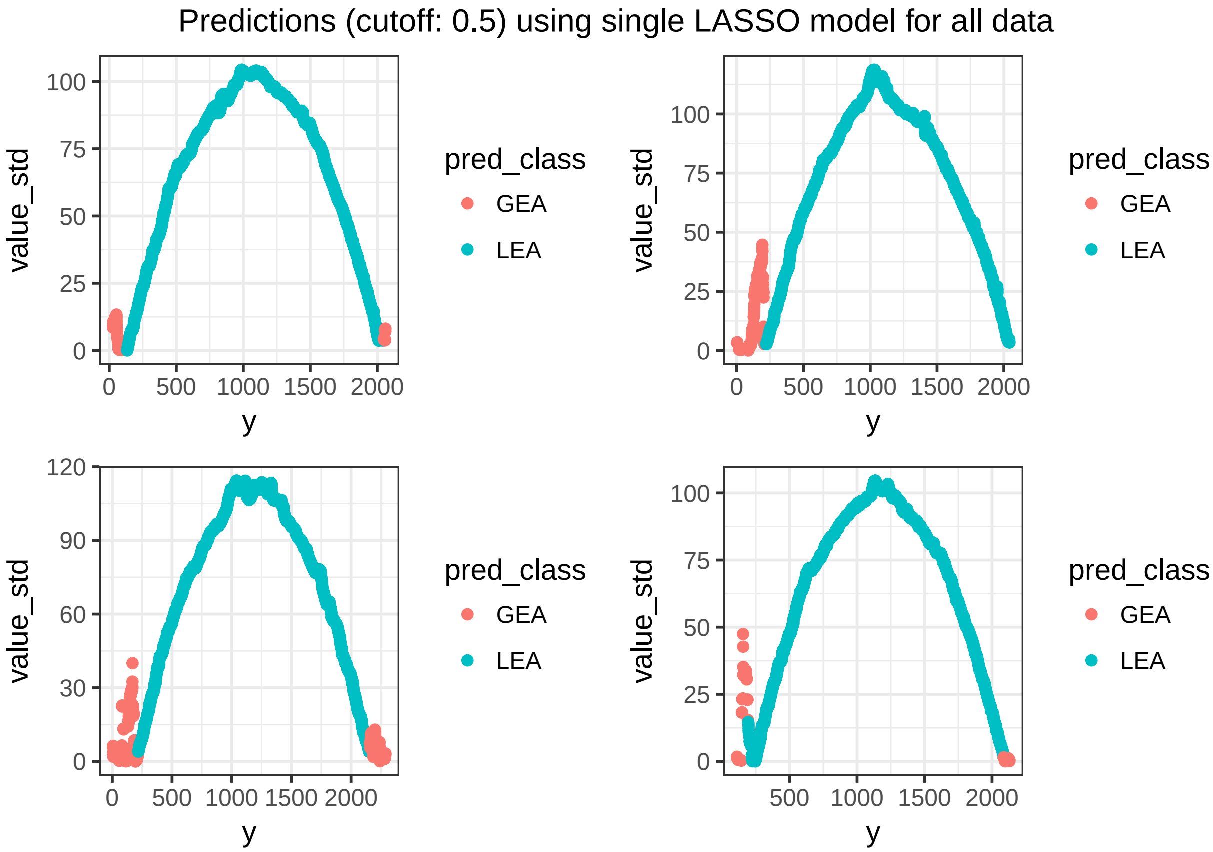


Note that when we fit individual models, the ROC curves are essentially perfect on each bullet.

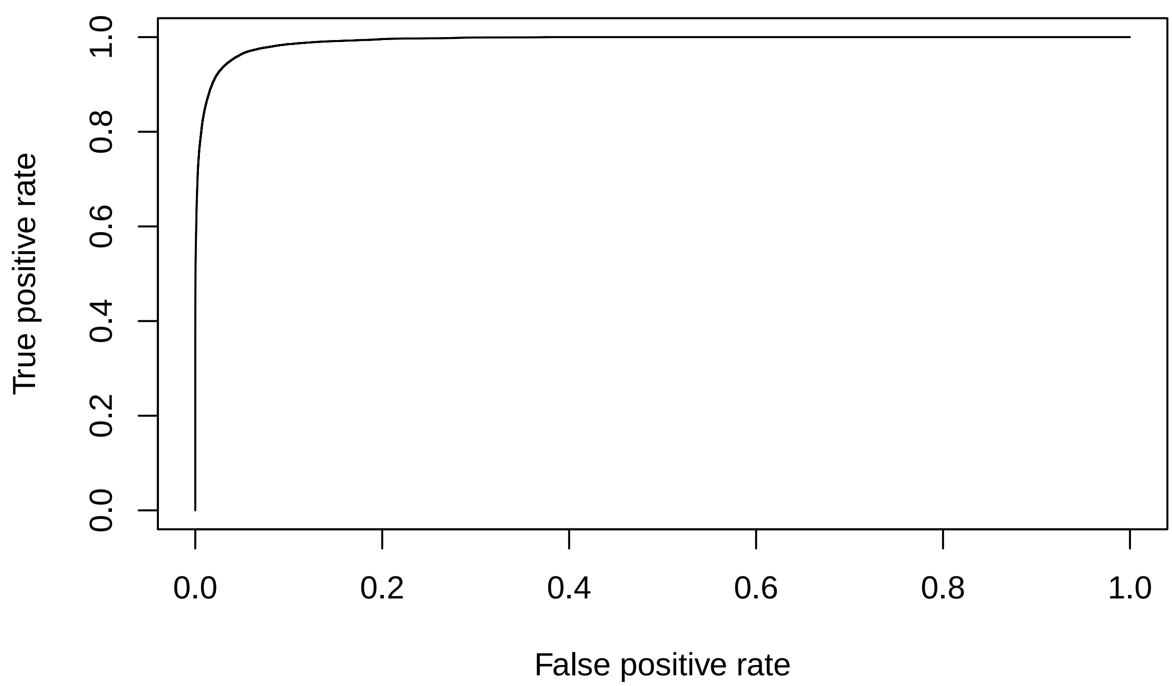
Now let's try this combining all the data into one large data frame and fitting ONE logistic regression model to it... then we can see if the ROC curves are a little more reasonable.

```
##          y     value value_std      pred absresid    resid  rlo_pred
## 45 28.380  9.215558  8.613795 -22.65220 31.26599 31.26599 -23.35250
## 46 29.025  9.083502  8.481739 -22.50094 30.98268 30.98268 -23.20053
## 47 29.670  9.617718  9.015955 -22.34978 31.36574 31.36574 -23.04865
## 48 30.315 11.395110 10.793347 -22.19871 32.99206 32.99206 -22.89686
## 49 30.960 10.855820 10.254057 -22.04773 32.30179 32.30179 -22.74516
## 50 31.605 10.247462  9.645699 -21.89685 31.54255 31.54255 -22.59356
##    rlo_absresid rlo_resid side    depth left_groove right_groove response
## 45     31.96630  31.96630 left 1050.382    129.7959     2022.594        1
## 46     31.68227  31.68227 left 1049.737    129.7959     2022.594        1
## 47     32.06460  32.06460 left 1049.092    129.7959     2022.594        1
## 48     33.69020  33.69020 left 1048.447    129.7959     2022.594        1
## 49     32.99922  32.99922 left 1047.803    129.7959     2022.594        1
## 50     32.23926  32.23926 left 1047.158    129.7959     2022.594        1
##            1   1
## 45 0.9902732 GEA
## 46 0.9898918 GEA
## 47 0.9899982 GEA
## 48 0.9909693 GEA
## 49 0.9903295 GEA
```

```
## 50 0.9895922 GEA
```



We can also look at the ROC curve from when we originally fit the data.




```
## [1] 0.9922574
```

Next steps:

Going to hold out a training/testing set next and try that - current ROC/AUC aren't accurate representations of model performance on a hold-out set.

Note: The AUC reported above is most likely so high due to the lack of testing set hold-out AS WELL AS the fact that we are dealing with SO MANY data points - a small smattering of misidentifications - even if they are really important mistakes in our eyes - is still a very small percentage of the overall data points.

Additionally, might try thinning out middle 50% of data. We are dealing with an unbalanced response variable; there are way more 0 (LEA) values than 1 (GEA) values.

Additional features to try:
- Quadratic term for robust LOESS residuals
- Standardizing robust LOESS residuals
- rlo_resid greater than cutoff value?

If these don't initially work, I am suggesting we try random forest (Alicia suggested BART? seems like it could be overkill) or another two-class classification procedure to attack this. I do think the initial results are promising though!!