Decision Thresholds for Logistic Regression on Imbalanced Data

Jared Hileman, John McMann, Ranik Jelinek

2024-04-26

Abstract

Many statistical techniques are adopted and employed to properly model various data types in the field of data analytics. For supervised learning tasks, a popular modeling technique for binary classification is logistic regression. This classification method estimates the probability of the response being 1 given the data. More formally, we receive a vector of probabilities

$$\tilde{\pi} = Pr(\hat{Y} = 1|\mathbf{X}).$$

An essential part of the algorithm requires a decision boundary to convert a probability generated from a logistic regression model to an outcome (0 or 1). A natural choice for such a decision threshold is 0.5, meaning a predicted probability greater than or equal to 0.5 is coded to a success, and values below this threshold will map to a failure. While 0.5 is the industry standard for logistic regression, the efficacy of this cutoff value is questioned in the presence of imbalanced data, where one outcome may be more common than the other. Particularly, this is of interest when trying to balance a model's sensitivity and specificity. How can we choose a decision threshold that allows our model to balance prediction of both outcomes? In this project, we aim to explore the relationship between data imbalance and decision thresholds in balancing the sensitivity and specificity of logistic regression models.

Methodology

One difficulty in determining optimal cutoff values in real-world datasets is that it is difficult to control for the proportion of successes in the data. Additionally, many such datasets would be needed to appropriately gauge any sort of trend in optimal cutoff values. It is for this reason that we will generate our own datasets, where parameter values, like proportion, can be controlled and where repeated simulation is fast.

R Package: ImbalancedLR

To facilitate the simulation process, we generated a rough-hewn R package, ImbalancedLR, to easily collect functions that are helpful in various processes. These processes including data generation, logistic regression model application, calculation of relevant metrics, and decision on the optimal cutoff value for a model based on the calculated metrics.

Data Generation

Several key considerations go into simulating data for logistic regression.

1. The response must be binomial, or $Y_i \sim Bernoulli(p_i)$.

2. We must select one or more features/predictors for the data. In this case, we chose to generate data with one predictor, $X \sim N(\mu, 1)$.

3. The response must have a relationship with the predictor(s). Since $Y_i \sim Bernoulli(p_i)$, it follows that

$$Y_i \sim Bernoulli(p_i = g(X_i)).$$

This constrains our function to be $0 \leq g(X) \leq 1$. We use a natural choice for g, the logistic function, given by

$$p_i = g(X_i) = \frac{e^{X_i}}{1 + e^{X_i}}.$$

4. We must be able to generate datasets that are imbalanced. In particular, we must be able to control $Pr(Y_i = 1|X_i)$.

From our choice of the logistic function to model probabilities, we inherit an elegant solution to this problem: the logit function. Solving g for X_i gives

$$\begin{aligned} X_i &= log\left(\frac{p_i}{1-p_i}\right) \\ \mathbb{E}(X_i) &= \mathbb{E}\left[log\left(\frac{p_i}{1-p_i}\right)\right] \\ &= log\left(\frac{p_i}{1-p_i}\right) \\ &= \mu. \end{aligned}$$

So, by centering X_i such that $X_i \sim N(\mu, 1)$, we expect to be able to generate data with proportion of success approximately p_i. In this simulation, we wish to generate data for imbalances $p = 0.05, 0.1, ..., 0.9, 0.95$.

5. Lastly, we need to ensure some noise is added into the data to avoid perfect prediction. Randomness is present in two apsects of data generation: the generation of X_i, where $\mathbf{X} = \text{rnorm}(n, \mu, 1)$, and $\mathbf{Y} = \text{rbinom}(n, 1, p = g(X))$.

These considerations result in datasets with one predictor $\mathbf{X}$ and one response $\mathbf{Y}$.

We generate five datasets for each desired level of imbalance described in (4) for $n = 100, 1000, 10000$. This results in 95 datasets per level of n, so 285 datasets in total.

Metrics

We will use two metrics to determine the optimal decision boundary for a particular.

Youden's J Statistic

A popular method for determining an optimal cutoff value can be found using the Youden's J Statistic, defined as

$$J = \text{sensitivity} + \text{specificity} - 1.$$

Since sensitivity and specificity both range from 0 to 1, $-1 \leq J \leq 1$. Ideally, we wish to have $J = 1$. This would indicate that the model perfectly classifies successes and failures. Thus, we will attempt to maximize Youden's J. Doing so ensures we are maximizing either sensitivity, specificity, or both.

Distance Metric

While maximizing the Youden's J Statistic is effective for maximizing sensitivity and specificity, it does not completely capture the notion of balancing the two model characteristics in the way that we would like. We can optimize a distance metric to correct for this. We use a simple notion of distance between two scalars,

$$D = |\text{sensitivity} - \text{specificity}|$$

We use these two metrics in tandem to find an ideal decision threshold. We impose that the optimal decision threshold will be c such that

$$\max_{c \in [0,1]} \{J\}$$
$$\text{subject to}$$
$$D < 0.1$$

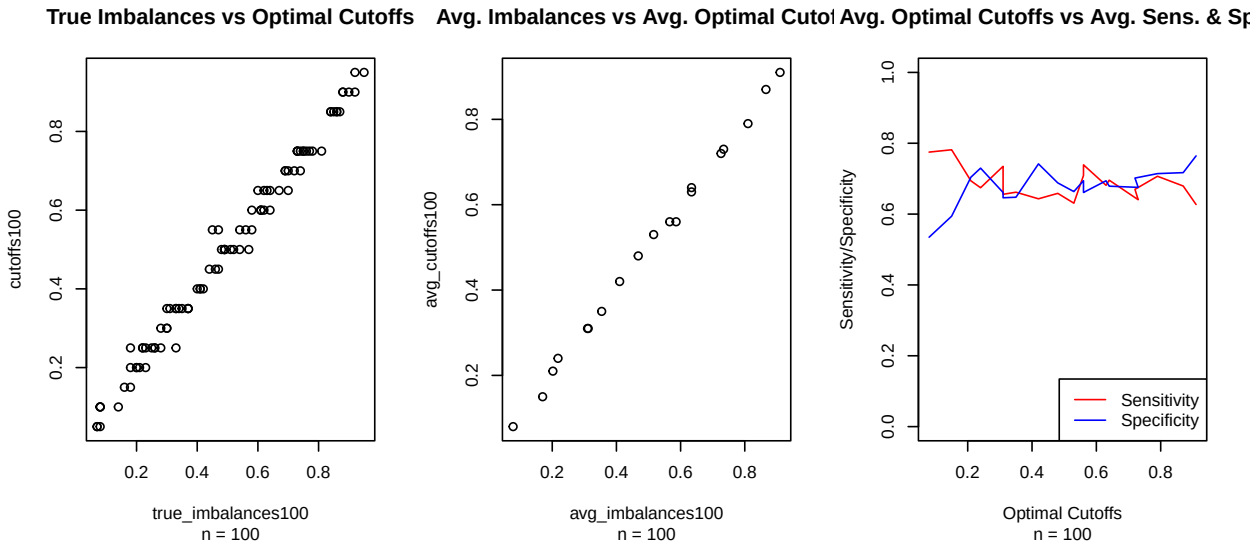
In the instance that a distance metric under 0.1 cannot be achieved, then the maximum Youden's J statistic is used.

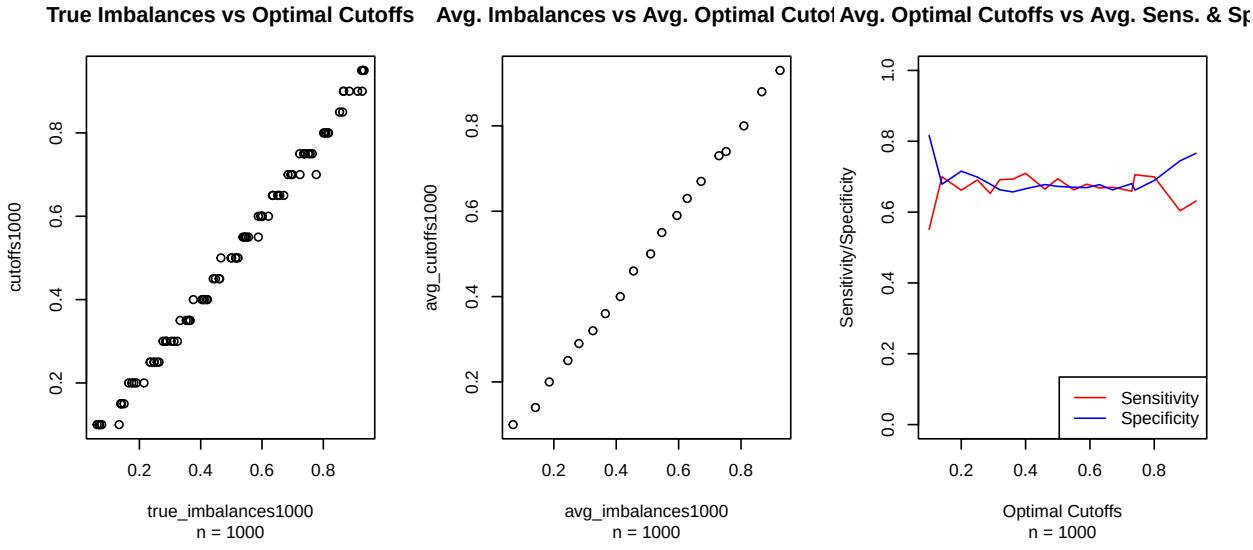
Simulation

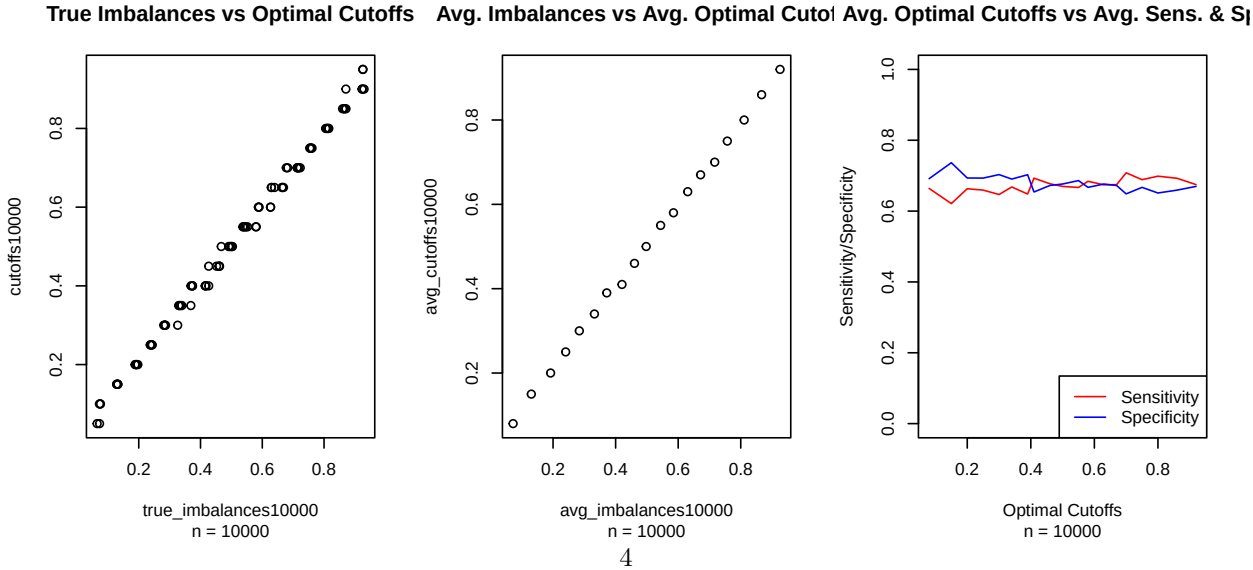
The simulation algorithm is as follows:

1. For each dataset, generate a logistic regression model.
2. For each cutoff value, $c = 0.05, 0.1, ..., 0.9, 0.95$, classify the results of the logistic regression model based on c. In particular, let $\pi_i = Pr(\hat{Y}_i = 1|X_i)$. For $\pi_i < c, Y_i = 0$. $Y_i = 1$ otherwise. This results in one prediction vector per cutoff value.
3. Calculate the sensitivity, specificity, and accuracy for each prediction vector.
4. Looking across the prediction vectors, choose the optimal decision threshold c' using the metrics defined above.
5. Compare the optimal decision threshold c to the true proportion of successes in the data.

Results


True Imbalances vs Optimal Cutoffs
Avg. Imbalances vs Avg. Optimal Cuto
Avg. Optimal Cutoffs vs Avg. Sens. & Sp
cutoffs100
true_imbalances100
n = 100
avg_cutoffs100
avg_imbalances100
n = 100
Sensitivity/Specificity
Optimal Cutoffs
n = 100
Sensitivity
Specificity


True Imbalances vs Optimal Cutoffs
Avg. Imbalances vs Avg. Optimal Cuto
Avg. Optimal Cutoffs vs Avg. Sens. & Sp
cutoffs1000
true_imbalances1000
n = 1000
avg_cutoffs1000
avg_imbalances1000
n = 1000
Sensitivity/Specificity
Optimal Cutoffs
n = 1000
Sensitivity
Specificity


True Imbalances vs Optimal Cutoffs
Avg. Imbalances vs Avg. Optimal Cuto
Avg. Optimal Cutoffs vs Avg. Sens. & Sp
cutoffs10000
true_imbalances10000
n = 10000
avg_cutoffs10000
avg_imbalances10000
n = 10000
Sensitivity/Specificity
Optimal Cutoffs
n = 10000
Sensitivity
Specificity

```
##                       Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)       0.004820008 0.006392478  0.7540125 4.611589e-01
## avg_imbalances100 0.988890977 0.011507534 85.9342230 7.084059e-24
```

```
## [1] 0.9977032
```

```
##                        Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)        0.008899837 0.005329238   1.670002 1.132271e-01
## avg_imbalances1000 0.984959333 0.009536830 103.279535 3.128503e-25
```

```
## [1] 0.9984088
```

```
##                         Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)          0.01703584 0.003535810   4.818088 1.606070e-04
## avg_imbalances10000 0.96912483 0.006333482 153.016118 3.945282e-28
```

```
## [1] 0.9992745
```

We generate several diagnostic and summary plots to ensure our chosen metrics are working as expected and to observe the relationship between decision thresholds and the true imbalance in our data. As a diagnostic tool to make sure our implementation was choosing an appropriate optimal decision threshold, we looked at the line plot of cutoff versus sensitivity and specificity. Both specificity and sensitivity seemed largely stagnant regardless of the cutoff value and approximately equal, which indicates our metrics are functioning as expected. We utilized scatter plots of the true imbalance versus the optimal decision threshold, as defined above, to observe their relationship.

Discussion

It is apparent that the default decision threshold of 0.5 is not optimal for balancing sensitivity and specificity when running logistic regression on imbalanced data. Visually, there is a very strong linear correlation between level of imbalance and chosen cutoff value, where the association between optimal cutoff values and true imbalance in the corresponding dataset is nearly an identical relationship, cutoff = imbalance.

We fit linear models to the data and verified that this was the case. For all levels of n, we see that the R^2 values are very high, indicating that the model is a good fit for the data. The coefficients for the linear models are also very close to 1, indicating that the relationship between the two variables is nearly identical.

Although a significant result was found regarding optimal cutoff values, some limitations were present throughout this process. One such limitation is perhaps a lack of generalizability to other, real-life datasets. In these data, we used only one predictor from the Gaussian family to generate our data. Unfortunately, this does not account for datasets with an asymmetric or noisy shapes.

Additionally, we observed a relatively low accuracy in our model (anywhere from 60% - 70%). Perhaps this is indicative that the response does not have a strong enough relationship with the predictor. While this would normally be problematic, since we are optimizing for specificity and sensitivity, we don't worry too much about the accuracy limitations.

Through continued study of this problem, different metrics could be implemented to optimize for a particular decision threshold. It would be interesting to apply these problems to real-world datasets with high imbalance and test our simulation-driven theory.

Appendix

```
############ GENERATE DATA FOR n = 100 #############
datalist100 <- list()
cutoff_info100 <- list()

centered <- centering(seq(0.05, .95, by = 0.05))

for (i in seq_along(centered)) {
  datalist100[[i]] <- list()  # Initialize inner list
  cutoff_info100[[i]] <- list()  # Initialize inner list

  for (j in 1:5) {
    datalist100[[i]][[j]] <- data_gen(size = 100, center = centered[i])
    cutoff_info100[[i]][[j]] <- best_cutoff(datalist100[[i]][[j]])
  }

  datalist100[[i]]$centered <- centered[i]
  cutoff_info100[[i]]$centered <- centered[i]
}

########## RESULTS FOR n = 100 #############

cutoffs100 <- vector()
true_imbalances100 <- vector()


avg_cutoffs100 <- vector()
avg_imbalances100 <- vector()
avg_sensitivity100 <- vector()
avg_specificity100 <- vector()

for (i in 1:length(cutoff_info100)) {
  these_cutoffs100 <- vector()
  these_imbalances100 <- vector()
  this_sensitivity100 <- vector()
  this_specificity100 <- vector()
  for (j in 1:5)  {
    curr_cutoff100 <- cutoff_info100[[i]][[j]]$optimal_cutoff
    curr_true_imbalance100 <- cutoff_info100[[i]][[j]]$true_imbalance
    curr_sens <- cutoff_info100[[i]][[j]]$optimal_sensitivity
    curr_spec <- cutoff_info100[[i]][[j]]$optimal_specificity
    cutoffs100 <- c(cutoffs100, curr_cutoff100)
    true_imbalances100 <- c(true_imbalances100,
                            curr_true_imbalance100)
    these_cutoffs100 <- c(these_cutoffs100, curr_cutoff100)
    these_imbalances100 <- c(these_imbalances100, curr_true_imbalance100)
    this_sensitivity100 <- c(this_sensitivity100, curr_sens)
    this_specificity100 <- c(this_specificity100, curr_spec)
  }
  avg_cutoffs100 <- c(avg_cutoffs100, mean(these_cutoffs100))
  avg_imbalances100 <- c(avg_imbalances100, mean(these_imbalances100))
  avg_sensitivity100 <- c(avg_sensitivity100, mean(this_sensitivity100))
```

```r
  avg_specificity100 <- c(avg_specificity100, mean(this_specificity100))
}



########## GENERATE DATA FOR n = 1000 ############

datalist1000 <- list()
cutoff_info1000 <- list()

centered <- centering(seq(0.05, .95, by = 0.05))

for (i in seq_along(centered)) {
  datalist1000[[i]] <- list()  # Initialize inner list
  cutoff_info1000[[i]] <- list()  # Initialize inner list

  for (j in 1:5) {
    datalist1000[[i]][[j]] <- data_gen(size = 1000, center = centered[i])
    cutoff_info1000[[i]][[j]] <- best_cutoff(datalist1000[[i]][[j]])
  }

  datalist1000[[i]]$centered <- centered[i]
  cutoff_info1000[[i]]$centered <- centered[i]
}

########## RESULTS FOR n = 1000 ############

cutoffs1000 <- vector()
true_imbalances1000 <- vector()

avg_cutoffs1000 <- vector()
avg_imbalances1000 <- vector()
avg_sensitivity1000 <- vector()
avg_specificity1000 <- vector()

for (i in 1:length(cutoff_info1000)) {
  these_cutoffs1000 <- vector()
  these_imbalances1000 <- vector()
  for (j in 1:5)  {
    curr_cutoff1000 <- cutoff_info1000[[i]][[j]]$optimal_cutoff
    curr_true_imbalance1000 <- cutoff_info1000[[i]][[j]]$true_imbalance
    cutoffs1000 <- c(cutoffs1000, curr_cutoff1000)
    true_imbalances1000 <- c(true_imbalances1000,
                         curr_true_imbalance1000)
    these_cutoffs1000 <- c(these_cutoffs1000, curr_cutoff1000)
    these_imbalances1000 <- c(these_imbalances1000, curr_true_imbalance1000)
  }
  avg_cutoffs1000 <- c(avg_cutoffs1000, mean(these_cutoffs1000))
  avg_imbalances1000 <- c(avg_imbalances1000, mean(these_imbalances1000))
}


########## GENERATE DATA FOR n = 10000 ############
```

```r
datalist10000 <- list()
cutoff_info10000 <- list()

centered <- centering(seq(0.05, .95, by = 0.05))

for (i in seq_along(centered)) {
  datalist10000[[i]] <- list()  # Initialize inner list
  cutoff_info10000[[i]] <- list()  # Initialize inner list

  for (j in 1:5) {
    datalist10000[[i]][[j]] <- data_gen(size = 10000, center = centered[i])
    cutoff_info10000[[i]][[j]] <- best_cutoff(datalist10000[[i]][[j]])
  }

  datalist10000[[i]]$centered <- centered[i]
  cutoff_info10000[[i]]$centered <- centered[i]
}



########## RESULTS FOR n = 10000 ############


cutoffs10000 <- vector()
true_imbalances10000 <- vector()

avg_cutoffs10000 <- vector()
avg_imbalances10000 <- vector()
avg_sensitivity10000 <- vector()
avg_specificity10000 <- vector()

for (i in 1:length(cutoff_info10000)) {
  these_cutoffs10000 <- vector()
  these_imbalances10000 <- vector()
  for (j in 1:5)  {
    curr_cutoff10000 <- cutoff_info10000[[i]][[j]]$optimal_cutoff
    curr_true_imbalance10000 <- cutoff_info10000[[i]][[j]]$true_imbalance
    cutoffs10000 <- c(cutoffs10000, curr_cutoff10000)
    true_imbalances10000 <- c(true_imbalances10000,
                          curr_true_imbalance10000)
    these_cutoffs10000 <- c(these_cutoffs10000, curr_cutoff10000)
    these_imbalances10000 <- c(these_imbalances10000, curr_true_imbalance10000)
  }
  avg_cutoffs10000 <- c(avg_cutoffs10000, mean(these_cutoffs10000))
  avg_imbalances10000 <- c(avg_imbalances10000, mean(these_imbalances10000))
}


########## PLOTS ############

par(mfrow = c(3, 3))
plot(true_imbalances100, cutoffs100, main = "True Imbalances vs Optimal Cutoffs",
```

```
     sub = "n = 100")

plot(avg_imbalances100, avg_cutoffs100,
     main = "Avg. Imbalances vs Avg. Optimal Cutoffs", sub = "n = 100")

plot(avg_cutoffs100, avg_sensitivity100,
     main = "Avg. Optimal Cutoffs vs Avg. Sens. & Spec.",
     type = "l", col = "red", xlab = "Optimal Cutoffs",
     ylab = "Sensitivity/Specificity", ylim = c(0, 1), sub = "n = 100")

lines(avg_cutoffs100, avg_specificity100, col = "blue")
legend("bottomright", legend = c("Sensitivity", "Specificity"),
       col = c("red", "blue"), lty = 1)


plot(true_imbalances1000, cutoffs1000,
     main = "True Imbalances vs Optimal Cutoffs", sub = "n = 1000")
plot(avg_imbalances1000, avg_cutoffs1000,
     main = "Avg. Imbalances vs Avg. Optimal Cutoffs", sub = "n = 1000")
plot(avg_cutoffs1000, avg_sensitivity1000,
     main = "Avg. Optimal Cutoffs vs Avg. Sens. & Spec.",
     type = "l", col = "red", xlab = "Optimal Cutoffs",
     ylab = "Sensitivity/Specificity", ylim = c(0, 1), sub = "n = 1000")
lines(avg_cutoffs1000, avg_specificity1000, col = "blue")
legend("bottomright", legend = c("Sensitivity", "Specificity"),
       col = c("red", "blue"), lty = 1)



plot(true_imbalances10000, cutoffs10000,
     main = "True Imbalances vs Optimal Cutoffs",
     sub = "n = 10000")

plot(avg_imbalances10000, avg_cutoffs10000,
     main = "Avg. Imbalances vs Avg. Optimal Cutoffs", sub = "n = 10000")

plot(avg_cutoffs10000, avg_sensitivity10000,
     main = "Avg. Optimal Cutoffs vs Avg. Sens. & Spec",
     type = "l", col = "red", xlab = "Optimal Cutoffs",
     ylab = "Sensitivity/Specificity", ylim = c(0, 1), sub = "n = 10000")

lines(avg_cutoffs10000, avg_specificity10000, col = "blue")
legend("bottomright", legend = c("Sensitivity", "Specificity"),
       col = c("red", "blue"), lty = 1)



########## FITTING MODELS #############



##### Linear regression model for n = 100 #####
```

```r
model100 <- lm(avg_cutoffs100 ~ avg_imbalances100,
               data = data.frame(avg_cutoffs100, avg_imbalances100))

model100_summary <- summary(model100)
model100_summary$coefficients
model100_summary$r.squared


##### Linear regression model for n = 1000 #####

model1000 <- lm(avg_cutoffs1000 ~ avg_imbalances1000,
               data = data.frame(avg_cutoffs1000, avg_imbalances1000))

model1000_summary <- summary(model1000)
model1000_summary$coefficients
model1000_summary$r.squared


##### Linear regression model for n = 10000 #####

model10000 <- lm(avg_cutoffs10000 ~ avg_imbalances10000,
               data = data.frame(avg_cutoffs10000, avg_imbalances10000))

model10000_summary <- summary(model10000)
model10000_summary$coefficients
model10000_summary$r.squared
```

You can find the public ImbalancedLR Github respository linked here: https://github.com/Jaredhileman/imbalancedLR