Predicting Initial Coin Offering Success

11/17/2021

Ryan Combs, Adam Guskiewicz & Thomas McDonnell

Table of Contents

1. Introduction

The cryptocurrency boom over the past half-decade has sparked a lot of research into how to capitalize on this bullish market. From November 2016 to November 2021, the market capitalization of the cryptocurrency market has jumped from 14.01 billion USD to 2.866 trillion USD, a 20,356.8 % increase (Statista). As one can infer from that statistics, 100$ invested in the diversified cryptocurrency market in 2016 would now be worth over 2 million dollars. These extreme gains are part of the reason for the current craze over the crypto market. Many investors don't know where to start, and stochastically invest in small coins in hopes to 'get rich quick,' yet overall, this is a losing strategy. This paper will discuss predictions of success and magnitude of initial coin offerings (ICO) using previous ICO data.

Like an initial public offering (IPO) an ICO is a tactic used by companies, especially startups, to raise money. In an IPO, a company 'goes public,' and sells off shares of their company to raise money normally to finance new factories, new product development, etc. Similarly, startups use ICO's to sell off coins to raise money for certain projects, generally related to blockchain technology. The majority of these ICO's are unsuccessful, but when a success does occur, the potential gains are massive. The goal of our paper is to see which variables are effective in predicting success and magnitude of success in ICO's. Using this data, we can develop investing strategies to maximize our potential gain while minimizing our risk.

2. Literature Review

The main study conducted on this topic analyzed the performance of companies after their ICO. The study was performed by Paul Momtaz and Christian Fisch. The paper found that there are significant ways to obtain above-market performance when investing in ICO's, meaning that there likely are many variables that can be used to predict ICO success (Fisch & Momtaz, 2020).

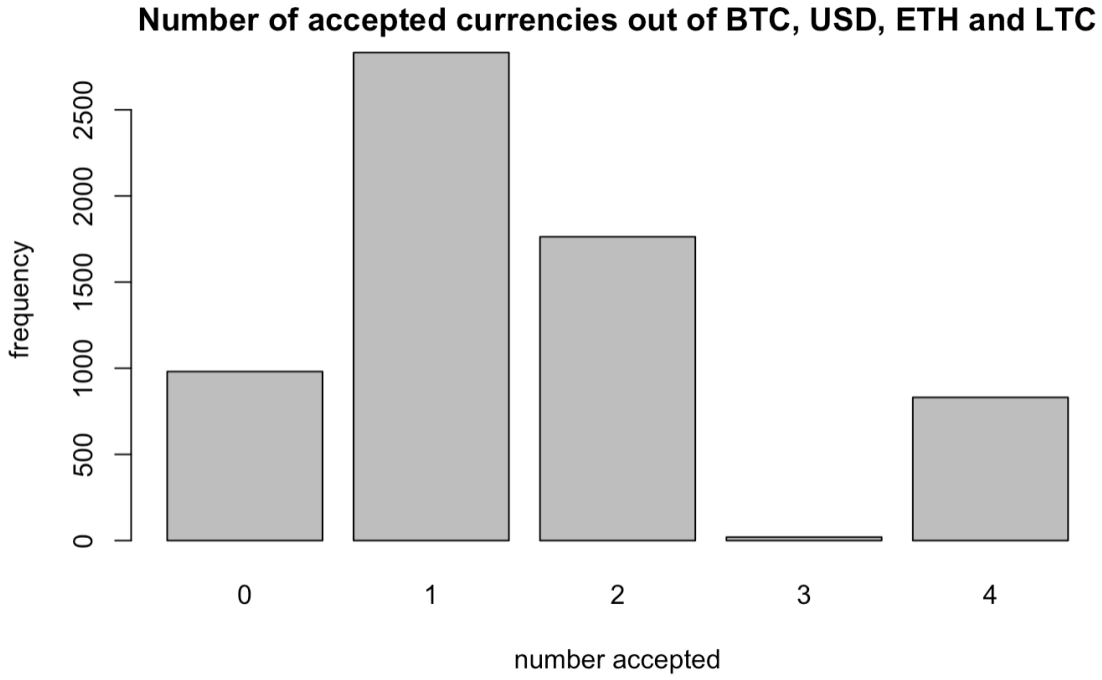
Another paper by Kale-ab Tessera uses machine learning techniques to predict the price of ICO's. This differs from our paper as we are more looking at success vs failure, and amount of money raised, although one of the variables that we use in our predictions is ICO price. He uses ridge regression and neural networks to create models, and found that using neural networks resulted in a lower MSE and higher R^2 value. Then, using this model he was able to create fairly consistent predictions of ICO price (Tessera, 2018).

3. Data

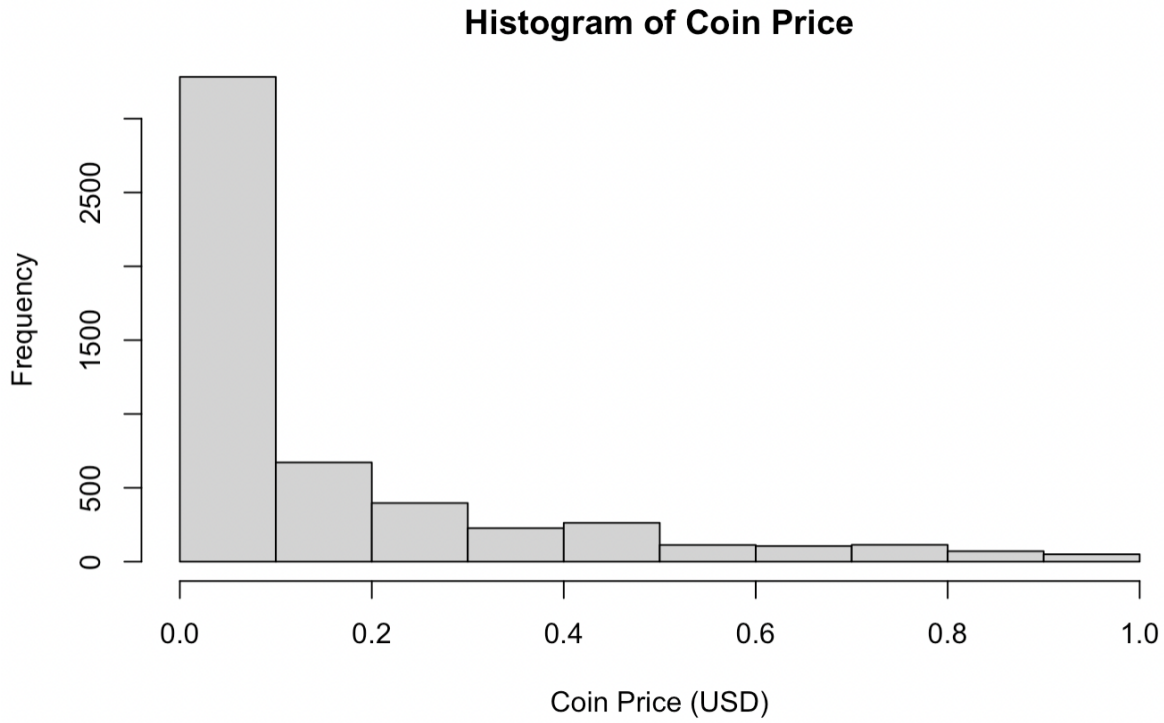
The data set utilized in our study had 6,427 observations of ICO's dating from 2014 through 2020. The data set was created by Paul Momtaz, a computer scientist at UCLA. Out of these ICO's, 2053 were successful. We quantify success as having raised any amount of USD, so an ICO can still be considered a success without making huge gains. There were 35 variables of interest including price of the coin, hard and soft cap, start and end date, sold tokens, accepted currency and more. We measure the magnitude of success as the amount raised in US dollars. Also, for every coin there is a LinkedIn link, a white paper link and the link to the website of the startup. One of the issues with this dataset that make some analysis difficult is for some variables such as 'minimum investment' the values are in different units such as bitcoin or ethereum. This

results in these being categorical variables with no good way to transform them into a normalized unit since they are represented with multiple different types of currency.

One variable of interest that we wanted to expand on was the currency that the company accepted for initial investments. To simplify this, we split it up into 4 dummy variables: accepting ethereum, accepting bitcoin, accepting USD and accepting litecoin. This lets us examine these variables individually, and also examine the total amount of currency accepted to see if it had an effect on ICO success. An interesting point of observation was that almost no ICO's accepted three of these currencies. The most frequent were BTC and ETH, but way more of the coins accepted all four of the main currencies than just three.



Below is a Histogram of the coin price in the ICO for prices below one USD. As you can see, the majority of these coins are just a fraction of a dollar, and we hypothesize that this variable will have a significant positive relationship on ICO success.



4. Methods

We are dealing with a classification problem, so logistic regression, regularization, and tree-based methods are the most relevant approaches to our objective. We chose to apply logistic regression, K-nearest neighbors, ridge, LASSO, decision trees, boosting, and random forests.

4.1. Logistic Regression

For our logistic regression, we first took a sample of 5000 observations from our complete dataset to be used as our training set. The remaining 1428 observations were placed into our testing dataset. We used the glm() function to create our logistic regression in R, with our variable "success" being our response variable, and 17 variables from the dataset as our predictors. With our training data and the glm(), in which we set the family to "binomial" we produced the following output:

```
Call:
glm(formula = success ~ rating + teamsize + platform + token_for_sale +
    kyc + price_usd + soft_cap_usd + hard_cap_usd + distributed_in_ico +
    whitelist + bonus + bounty + accepting_btc + accepting_ltc +
    accepting_usd + accepting_eth + duration, family = "binomial",
    data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0169  -0.8785  -0.5760   1.0524   2.9435

Coefficients:
                            Estimate           Std. Error z value             Pr(>|z|)
(Intercept)        -3.19196829300631579  0.17355199670041951 -18.392 < 0.0000000000000002 ***
rating              1.08214803646099300  0.06433515789858384  16.820 < 0.0000000000000002 ***
teamsize            0.02483727783072220  0.00490938091093752   5.059     0.00000042113697 ***
platform           -0.11821069435750964  0.06668493236513127  -1.773              0.07628 .
token_for_sale      0.00000000000006402  0.00000000000013438   0.476              0.63379
kyc                -0.09079696477511687  0.09364599822626898  -0.970              0.33226
price_usd          -0.00002789495190916  0.00005129091808972  -0.544              0.58654
soft_cap_usd        0.00000000019540533  0.00000000017185743   1.137              0.25553
hard_cap_usd       -0.00000000003424505  0.00000000002983324  -1.148              0.25102
distributed_in_ico -0.19606274955640893  0.11840994322581737  -1.656              0.09776 .
whitelistNo        -0.85114692916521129  0.09681296344637901  -8.792 < 0.0000000000000002 ***
whitelistYes       -0.84320209714127781  0.11827140516468324  -7.129     0.00000000000101 ***
bonus              -2.99014360664260836  0.26910723630013939 -11.111 < 0.0000000000000002 ***
bounty             -0.16671035923746114  0.08022559076167651  -2.078              0.03771 *
accepting_btc       0.12697784368051684  0.07837284750591672   1.620              0.10519
accepting_ltc       0.11284107211861655  0.10540538686747845   1.071              0.28437
accepting_usd       0.02272735119570952  0.17542697579674910   0.130              0.89692
accepting_eth      -0.07665656506130000  0.09774958554528630  -0.784              0.43291
duration           -0.00142461189408695  0.00045754166936385  -3.114              0.00185 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6396.2  on 4998  degrees of freedom
Residual deviance: 5514.6  on 4980  degrees of freedom
AIC: 5552.6
```

The summary shows the change in log odds of success for a one unit increase in the respective variables. We can see that although many variables are significant, many are not. At the 10% level, we see that rating, teamsize, distributed_in_ico, the whitelist dummy, bonus, bounty, and duration are all significant.

Next we used the predict() function to calculate the probability that each coin would succeed in raising money for their ICO. If the probability of success for the coin was over .5, that coin was listed as 1, or success. We compared the results of the predict function to our original testing data and received these results:

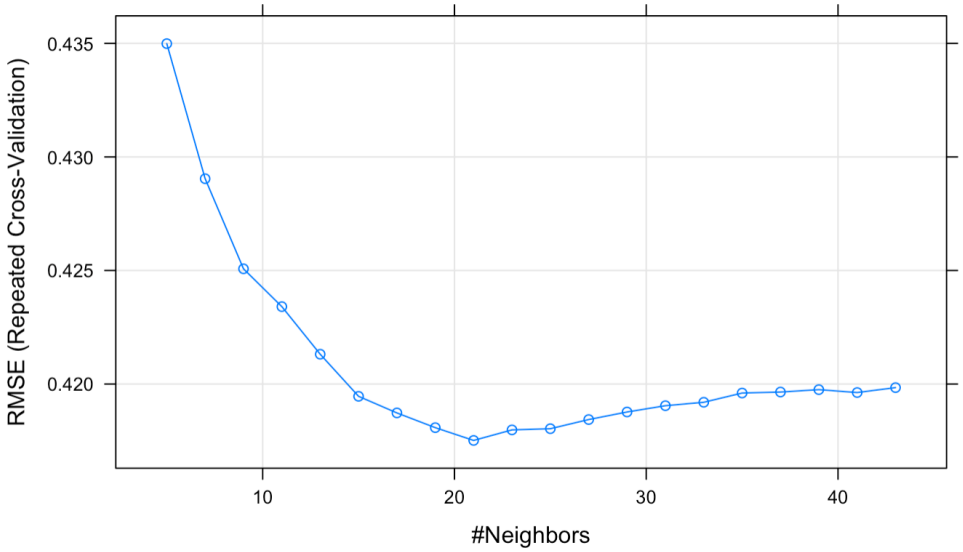
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1008  306
         1   56   57

               Accuracy : 0.7463
                 95% CI : (0.7229, 0.7687)
    No Information Rate : 0.7456
    P-Value [Acc > NIR] : 0.4899

                  Kappa : 0.135

 Mcnemar's Test P-Value : <0.0000000000000002
```

From this, we can see that our test misclassification rate is 362/1427 or 25.37%. Also, we see the 95% confidence interval for accuracy as (72.29% to 76.87%).

4.2. K-Nearest Neighbors (KNN)

K-nearest neighbors is a supervised machine learning algorithm that can be used for both classification and regression purposes. In our study we used KNN to discover which variables affect ICO success, and then make predictions. To start out, we did a standard testing and training split of 75% training and 25% testing. The output variable that we are attempting to

predict in this model is ICO success, which is defined as USD raised being greater than 0. Next, we removed the qualitative variables that weren't dummy's from the dataset such as 'coin_name' and 'github_link'. This left our training and testing datasets having 4,821 and 1,606 data points respectively with 25 variables of interest. Next, we centered and scaled all of the variables using R's 'preProcess' function. 22 variables were scaled and centered while 2 of them were ignored. After cross validation we found that the optimal number for K was 21 using the output below. K equal to 21 will give us the best estimate as it minimizes the mean squared error which is our goal.



Using K equal to 21, we proceeded to make our predictions on our testing dataset. Given the parameters, these predictions were not integers and were mainly ranging from 0 to 1, so we used the R round function to round to 0 digits resulting in all values taking 0 if it was less than .5 and 1 if it was greater than .5. Now we had a list of 1,606 values of either 0 or 1 using data from our testing set, so we compared the actual values of success with our predictions to see how accurate the model was.

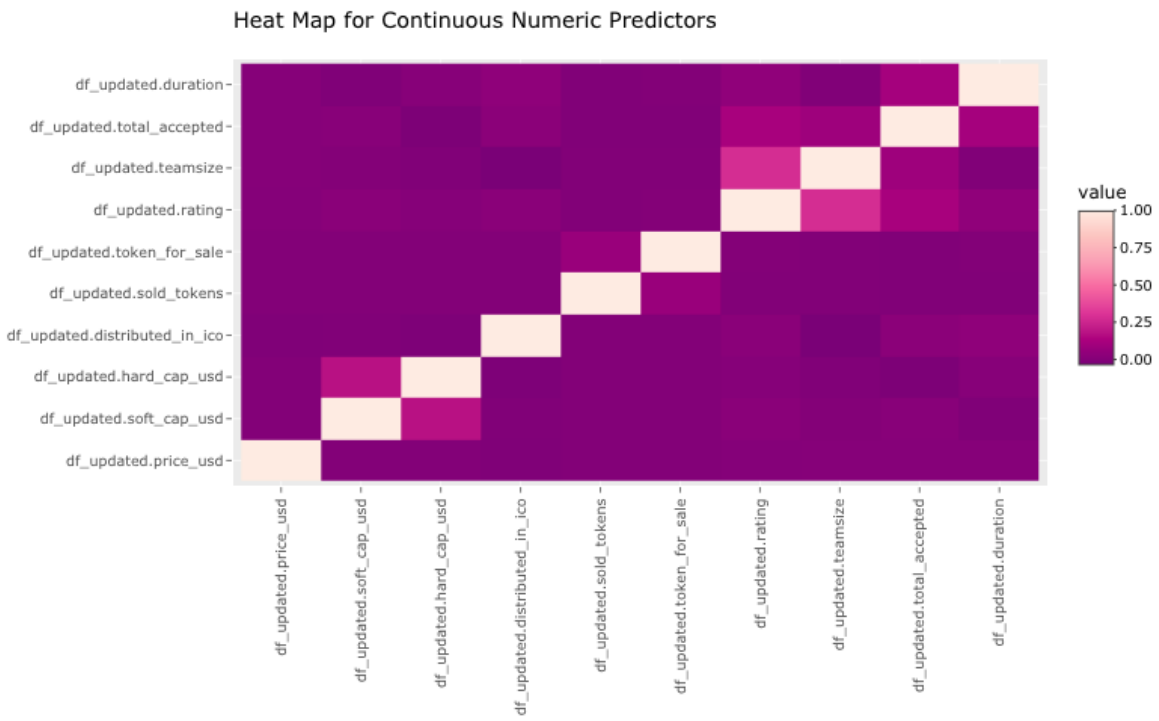
The following table is the output of the KNN confusion matrix, which is a table that shows how accurate your predictions are. As you can see, our model has 76.09% predictive accuracy with a 95% confidence interval from 73.93% to 78.16%. This will be our best predictive model for KNN as we have already done cross validation to minimize mean standard error.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1028  319
         1   65  194

               Accuracy : 0.7609
                 95% CI : (0.7393, 0.7816)
    No Information Rate : 0.6806
    P-Value [Acc > NIR] : 0.0000000000008651

                  Kappa : 0.3669

 Mcnemar's Test P-Value : < 0.00000000000000022
```

4.3. Regularization

4.3.1 Ridge

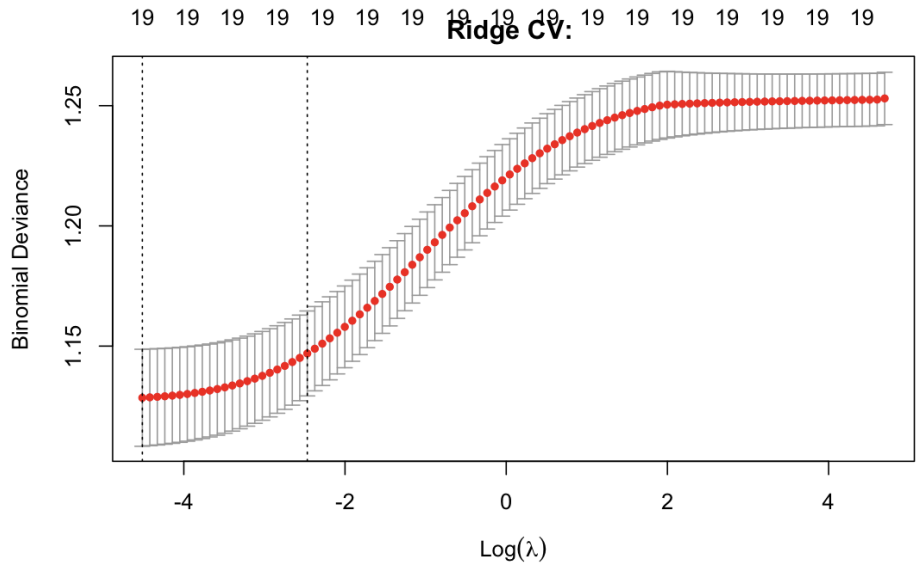
Regularization methods aim to reduce noise within data and eliminate covariates. We split our scaled data into 75/25 train/test subsets and began to assess potential sources of covariation that might show up in our regressions. One way that we visualized correlation between variables is with the following heat map:



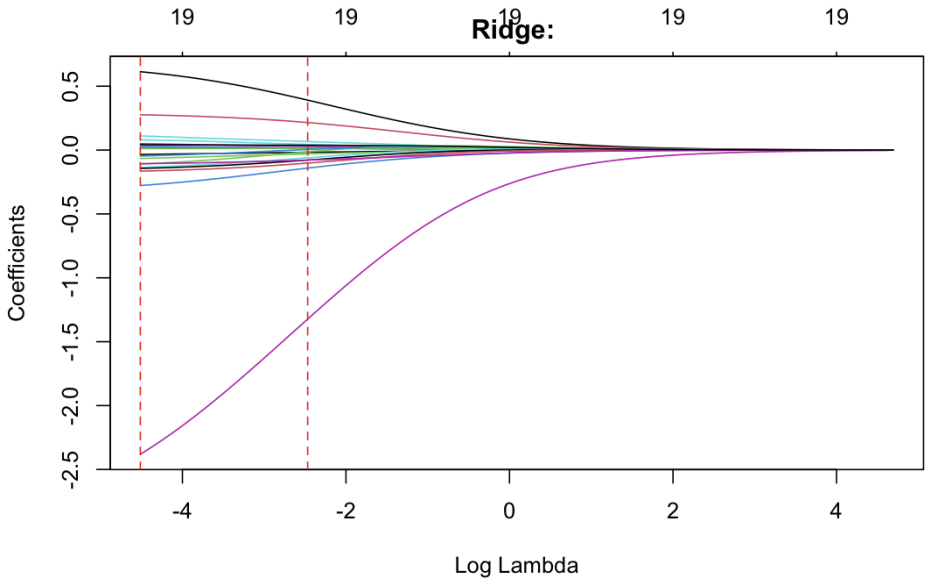
We notice that most have a rather low correlation, indicating that our data is unlikely to contain significant covariants. This suggests that we do not need to worry about violations of our linear regression assumptions before conducting our regularization. Before assessing any models, let's consider the strengths and usage cases for ridge and LASSO, which are distinguished by α level. The ridge regression never eliminates unnecessary variables, because it has an α level of 0. For this reason, ridge is a good choice for smaller datasets where there may not be any need to eliminate variables entirely. Conversely, LASSO is better at eliminating noise and will output a simpler model. Although ridge appears to be a more appropriate choice, we will also conduct a LASSO to provide us with an alternative framework and test the possibility of a small, interpretable model for our data.

Regularization methods have a shrinkage term λ, which is the feature that penalizes unnecessary or noisy data. To identify the optimal λ for our ridge regression, we cross-validate

our training data and return a minimum λ of 0.01094549. One standard error above this value is λ = 0.08474698. We add a visual component to cross validation with the graph below.



After cross-validating, we train our minimum-λ ridge model using glmnet and fit it to the test data. Because we want to make a binomial prediction, we classify our ridge regression's predictions into success (> .5) or failure (<= .5) . The following plot illustrates the ridge model's shrinkage mechanism in action.

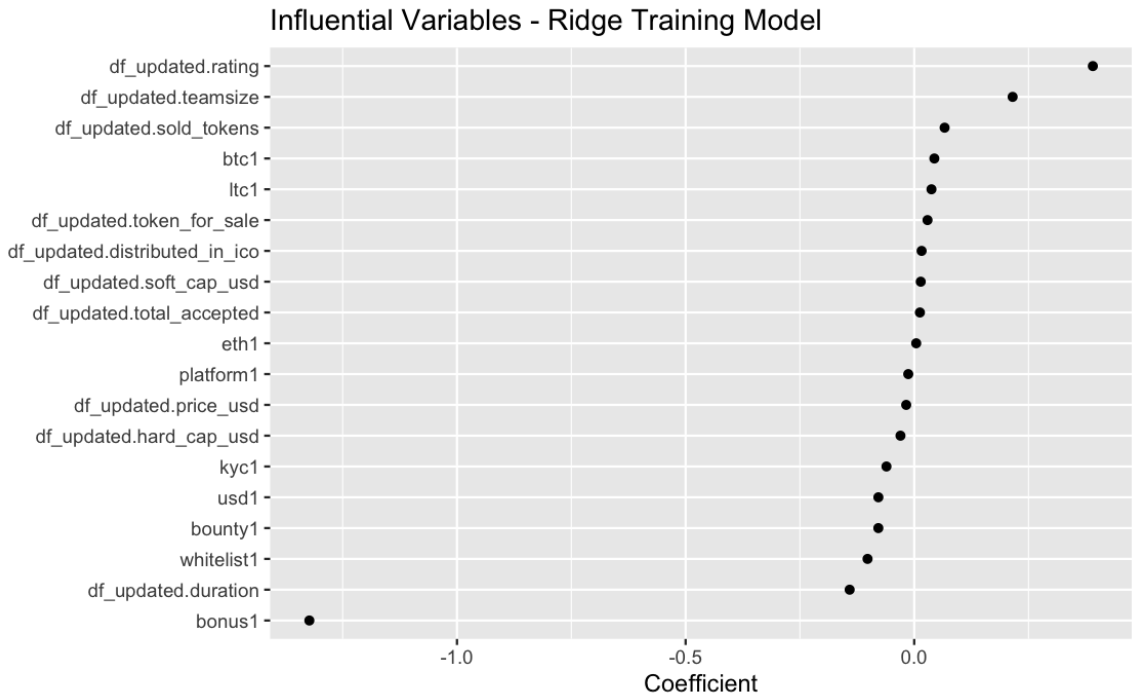


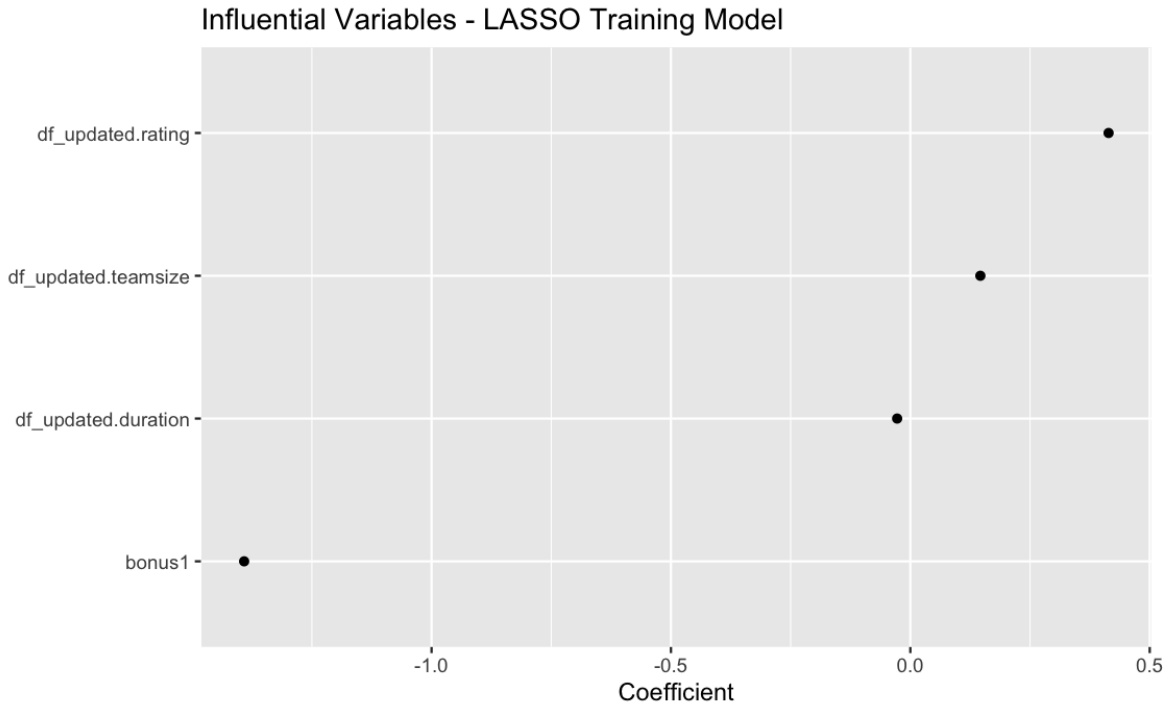
To measure the performance of our model, we put the ridge regression's predictions into a confusion matrix and evaluated them against the test data. Notice a formidable accuracy, but a high frequency of false negatives.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1084  485
         1   10   29

               Accuracy : 0.6922
                 95% CI : (0.669, 0.7147)
    No Information Rate : 0.6803
    P-Value [Acc > NIR] : 0.1613

                  Kappa : 0.0626

 Mcnemar's Test P-Value : <0.0000000000000002
```

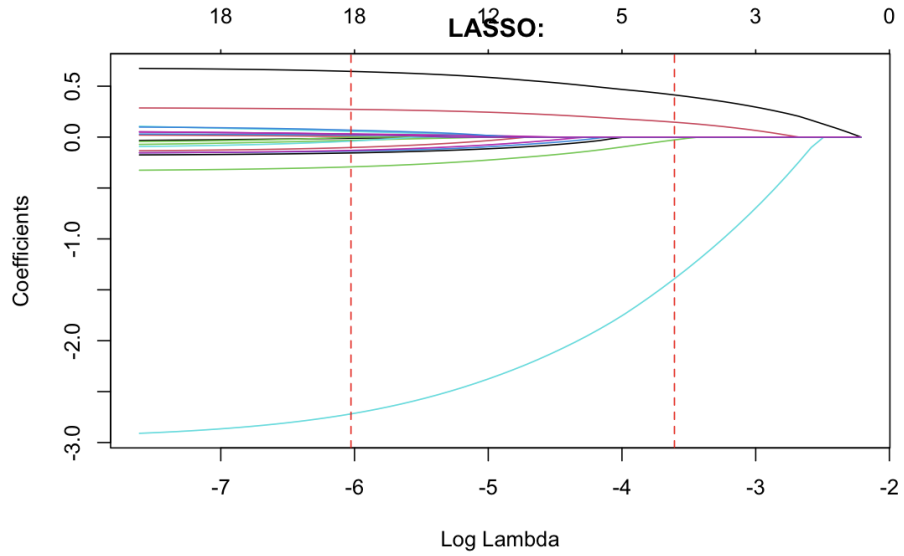
4.3.2 LASSO

For our LASSO model, we follow the exact same procedure as we did for ridge, but set α to 1. The LASSO shrinkage plot and performance report are shown below; in addition, we visually represent the influential variables for ridge and LASSO to demonstrate how our LASSO regression took the 4 most influential variables and eliminated the rest. The ridge has many variables with coefficients nearly at zero, while the LASSO shrunk those exact variables to zero to remove them from the model.


Influential Variables - Ridge Training Model
df_updated.rating
df_updated.teamsize
df_updated.sold_tokens
btc1
ltc1
df_updated.token_for_sale
df_updated.distributed_in_ico
df_updated.soft_cap_usd
df_updated.total_accepted
eth1
platform1
df_updated.price_usd
df_updated.hard_cap_usd
kyc1
usd1
bounty1
whitelist1
df_updated.duration
bonus1
-1.0
-0.5
0.0
Coefficient


Influential Variables - LASSO Training Model
df_updated.rating
df_updated.teamsize
df_updated.duration
bonus1
-1.0
-0.5
0.0
0.5
Coefficient

We present here the LASSO plot and confusion matrix results, which are similar to those of the ridge regression with some slight deviations from each other, parameters count aside.



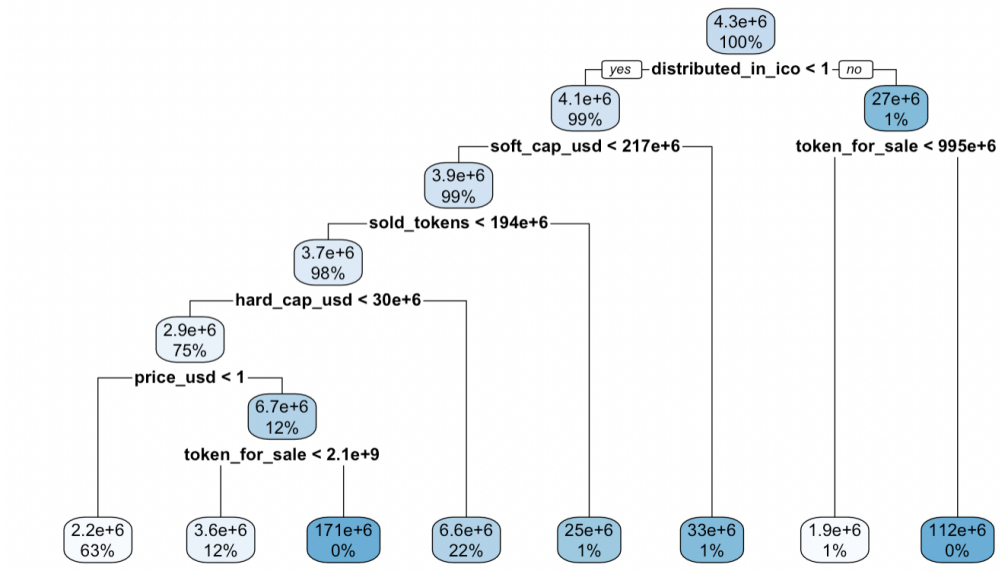

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1079  478
         1   15   36

               Accuracy : 0.6934
                 95% CI : (0.6702, 0.7159)
    No Information Rate : 0.6803
    P-Value [Acc > NIR] : 0.1363

                  Kappa : 0.074

 Mcnemar's Test P-Value : <0.0000000000000002
```

LASSO and ridge are both relatively accurate overall, with approximately 69% accuracy each, but we notice that both are very poor at correctly identifying successful ICOs. Less than 10% of successful ICOs were correctly identified by these models, but we believe that there is still much insight to gain from their results. Both ridge and LASSO serve a purpose here–since the LASSO

took away so many similar variables, it may have been doing so erratically. Therefore, we learn from LASSO that it is not worth trying to eliminate variables from subsequent models (in part because our dataset is so sparse). That said, the LASSO still serves as a proof-of-concept because it maintains a similar level of accuracy to the ridge without using any ex post facto variables (such as tokens distributed in ICO), suggesting that our general framework could be applied somewhere as a true predictive model.

4.4. Regression Trees

Regression trees are created by partitioning the data into smaller and smaller categories to make predictions on the outcome of the data. In our regression tree model we attempted to see which variables affect the amount of raised US dollars in an ICO. To start out, we split the data into training and testing subsets, 70 and 30 percent respectively. Then, using R's 'Caret' machine learning library, we were able to perform cross validation with all of our quantitative variables to find the optimal regression tree. The output is displayed below.



As you can see from the output, the first partition made was the proportion of the coin distributed in the ICO. If 100% of the coins were distributed in the ICO, then the other variable used to predict outcome was the amount of tokens for sale. If less than 100% of the coins were distributed in the ICO then there were many more branches used to determine the amount of raised USD. The significant variables in determining USD raised according to this model were the proportion distributed in the ICO, the soft cap, sold tokens, hard cap, price of coin and total coins for sale.

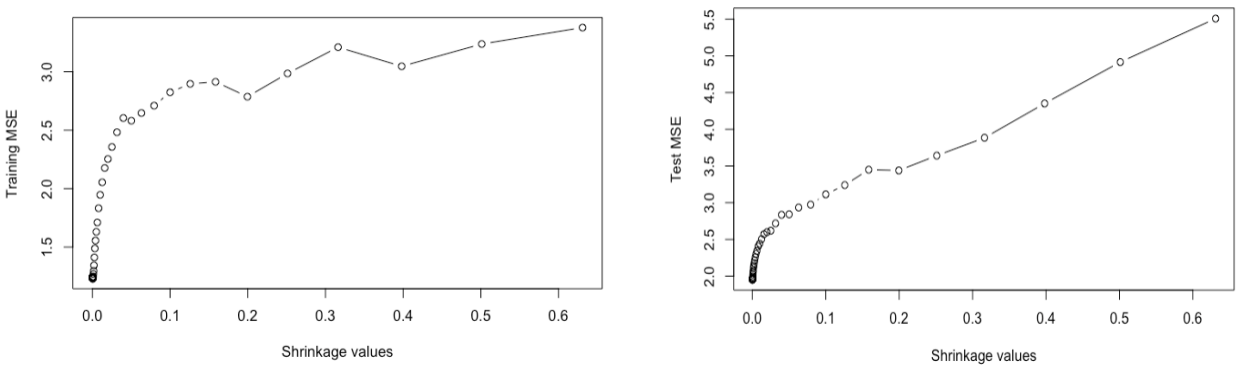
4.5. Boosting

Boosting is a process in which trees are grown sequentially where each tree is grown using information from previously grown trees. For this section, we first boosted our data with n=1000 trees and the distribution set to “bernoulli”. Our summary results for our best 10 variables are shown as:

	var <chr>	rel.inf <dbl>
rating	rating	18.1778428
price_usd	price_usd	14.2744876
token_for_sale	token_for_sale	13.8753361
soft_cap_usd	soft_cap_usd	13.5666190
hard_cap_usd	hard_cap_usd	12.2593713
distributed_in_ico	distributed_in_ico	12.0185702
teamsize	teamsize	10.0376488
bounty	bounty	1.2443886
accepting_usd	accepting_usd	1.1041384
kyc	kyc	0.9411405

The variables with the most influence were rating, price_usd, soft_cap_usd, and token_for_sale. This is not surprising as the initial rating of the coin as well as the price of the coin should play a key role in the success of the ICO. Next, we seeked to find the optimal λ that decreased our training and testing MSE. We built a for loop to test λ values through the boosted data. Our

results for the training and testing data are shown below, revealing that test MSE was minimized at 1.95 with a λ value of .000126.



4.6. Random Forests

Finally, we elected to use a random forest to predict the success of an ICO as a supplementary procedure to both our tree-based methods and our regularization methods. The decision tree that we grew earlier serves as a visual benchmark for our prediction process, and the boosting algorithm extrapolates that process into simulated magnitude. Because boosting uses shallow trees, which have a high bias and a low variance, our boosting algorithm does its best to reduce bias. The boosting algorithm grows trees sequentially; conversely, the random forest algorithm is made to reduce variance by growing all trees simultaneously. We chose to grow a random forest to explore the other side of the bias-variance trade-off because it uses deeper decision trees, which have low bias and high variance.

For any tree-based method, it is important to distinguish categorical and numeric data, so we used caTools to prepare the data because it makes it easy to designate data classes within matrices. We did not transform the scale of any columns because a tree-based method does not involve regression coefficients, or other features in which proportionality is essential. We

conducted a 75/25 training/test split and used the training set to grow a forest with the randomForest package. We used that forest to predict successes and failures and compare that to the actual successes in the test data in a confusion matrix, returning this output.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 978 311
         1 116 202

               Accuracy : 0.7343
                 95% CI : (0.712, 0.7558)
    No Information Rate : 0.6808
    P-Value [Acc > NIR] : 0.000001682

                  Kappa : 0.32

 Mcnemar's Test P-Value : < 0.00000000000000022
```

The accuracy is roughly the same as the rest of our models, but the significance of this random forest is in its capacity to predict successful ICOs. Our ridge, LASSO, and logistic regressions had ~70% accuracy, but all had much higher rates of false negatives. Given the vast majority of ICOs fail, our regularization and regression models may retain comparable levels of accuracy due to being rather pessimistic. Nonetheless, it is apparent that our random forest is better able to identify ICOs that will eventually succeed.

5. Conclusion

The models that we created throughout this process give us great insight into how to make predictions on ICO success. According to our confusion matrix for our linear regression, our model had 74.63% predictive accuracy which is very solid, and also gave us a good picture on which variables are significant. Our ridge and LASSO models performed worse than a basic logistic regression in measuring ICO success, but our LASSO was able to fit a model of the same accuracy as ridge containing only four variables–none being post-factum–indicating that our method has real-world predictive potential. Our most accurate model for predicting ICO success was the K-nearest neighbors, which had an accuracy rate of 76.09%. The KNN model dramatically outperformed our regression models in its ability to identify ICOs that went on to be successful. It was able to correctly predict nearly 40% of eventual successes, matched only by our random forest model. Given the encouraging performance of these two models, we can confidently recommend further exploration into other nonlinear classifiers should further research be conducted on this dataset.

From our analysis, we can see that there are variables significant in predicting ICO success that are present within the data we have chosen to assess. We are encouraged by promising model simplicity in our LASSO and the predictive accuracy of our nonlinear classifiers, lending us strong conviction that it is possible to accurately forecast the success of ICOs given a similar set of parameters.

6. References

https://www.paulmomtaz.com/data/tord

https://www.sciencedirect.com/science/article/pii/S0929119920301231

https://towardsdatascience.com/icoomen-using-machine-learning-to-predict-ico-prices-29fa4cec6d86