DAR F21 Project Status Notebook Assignment 4

DeFi

Jason Podgorski (GitHub: podgoj)

10/04/2021

Contents

Summary of Work

For notebook 4, I wanted to take a deep dive into common patterns between two separate groups: users who have liquidated and users who haven't. I wanted to look at features that would be a good indicator if a user is likely to fall into a particular group and then run a machine learning model to assess the quality of the features I discovered.

GitHub Commits

Branch Name: dar-podgoj

Files on GitHub:

podgoj_assignment04.Rmd - https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/Assignment04/podgoj_assignment04.Rmd

podgoj_assignment04.pdf - https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/Assignment04/podgoj_assignment04.pdf

podgoj_assignment04.html - https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/Assignment04/podgoj_assignment04.html

Personal Contribution

All of the work in this notebook is my own.

Primary Findings

The goal of this notebook was to discover features that would distinguish characteristics between a user who has liquidated in the past and a user who hasn't. To accomplish the feature engineering, I did an exploratory data analysis with a series of charts that I thought would show a difference between the two groups. My initial set of features included the number of borrows per user, repays per user, and the average stable and variable rates on borrows of DAI, USDC, and USDT. I then split the data in 75% training and 25% testing to run a prediction using logistic regression and random forest models. When analyzing the features, I noticed that borrows and repays dominated the model. I ran both models using borrows and repays as the only features and had promising results. I had 94.7% accuracy with the random forest model and 73.7% accuracy with logistic regression when predicting the test data.

The Code

```r
# load Rds (binary version of csv file) into dataframe
df <- read_rds('../../Data/transactions.Rds')
head(df, 2)
```

```
##       amount borrowRate borrowRateMode   onBehalfOf         pool reserve
## 1   41501.63   6.274937       Variable 8.502518e+47 1.034668e+48     DAI
## 2 7000000.00   2.589628       Variable 4.635974e+47 1.034668e+48    USDT
##    timestamp         user   type reservePriceETH reservePriceUSD amountUSD
## 1 1621340435 8.502518e+47 borrow    2.852900e+14       0.9948044     41286
## 2 1622477822 4.635974e+47 borrow    3.812835e+14       1.0000000   7000000
##   collateralAmount collateralReserve principalAmount principalReserve
## 1               NA                                NA
## 2               NA                                NA
##   reservePriceETHPrincipal reservePriceUSDPrincipal reservePriceETHCollateral
## 1                       NA                       NA                        NA
## 2                       NA                       NA                        NA
##   reservePriceUSDCollateral amountUSDPincipal amountUSDCollateral
## 1                        NA                NA                  NA
## 2                        NA                NA                  NA
##   borrowRateModeFrom borrowRateModeTo stableBorrowRate variableBorrowRate
## 1                                                   NA                 NA
## 2                                                   NA                 NA
```

```r
# create a new column in date format using timestamp variable
df <- df[order(df$timestamp),]
posixt <- as.POSIXct(df$timestamp, origin = "1970-01-01")
df$date <- as.Date(posixt)
```

```r
# get pool of unique users who have liquidated in their history
liquidators <- df[df$type == "liquidation",]
liquidators <- unique(liquidators$user)
liquidators_df <- df[df$user %in% liquidators,]
```

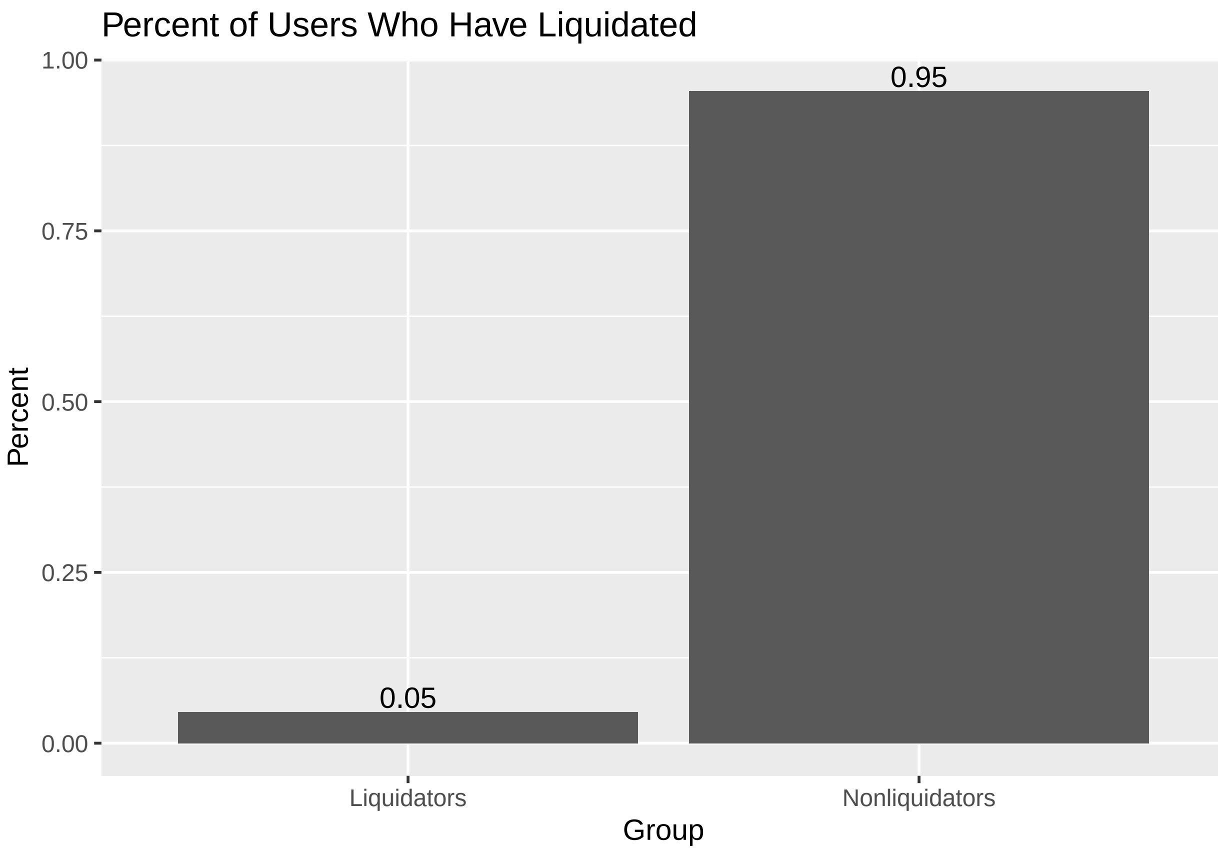
```r
# get pool of unique users who have not liquidated in their history
nonliquidators_df <- df[!(df$user %in% liquidators),]
nonliquidators <- unique(nonliquidators_df$user)
```

```r
# Compare the number of liquidating users with non-liquidating users in AAVE

# Create a new dataframe with the percentage of users who fall in each group
total_users <- length(unique(df$user))
```

```r
liquidator_counts_df <- data.frame(
  name = c("Liquidators", "Nonliquidators"),
  value = c(length(liquidators) / total_users, length(nonliquidators) / total_users)
  )

# Plot as a bar chart
ggplot(liquidator_counts_df, aes(x = name, y = value)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = round(value, digits = 2)), vjust = -0.2) +
  ylab("Percent") +
  xlab("Group") +
  ggtitle("Percent of Users Who Have Liquidated")
```



95% of unique users in AAVE have not liquidated yet. This includes users with 1 transaction and users with 300+ transactions. This means the number of frequent users who have liquidated is much greater than 5%.

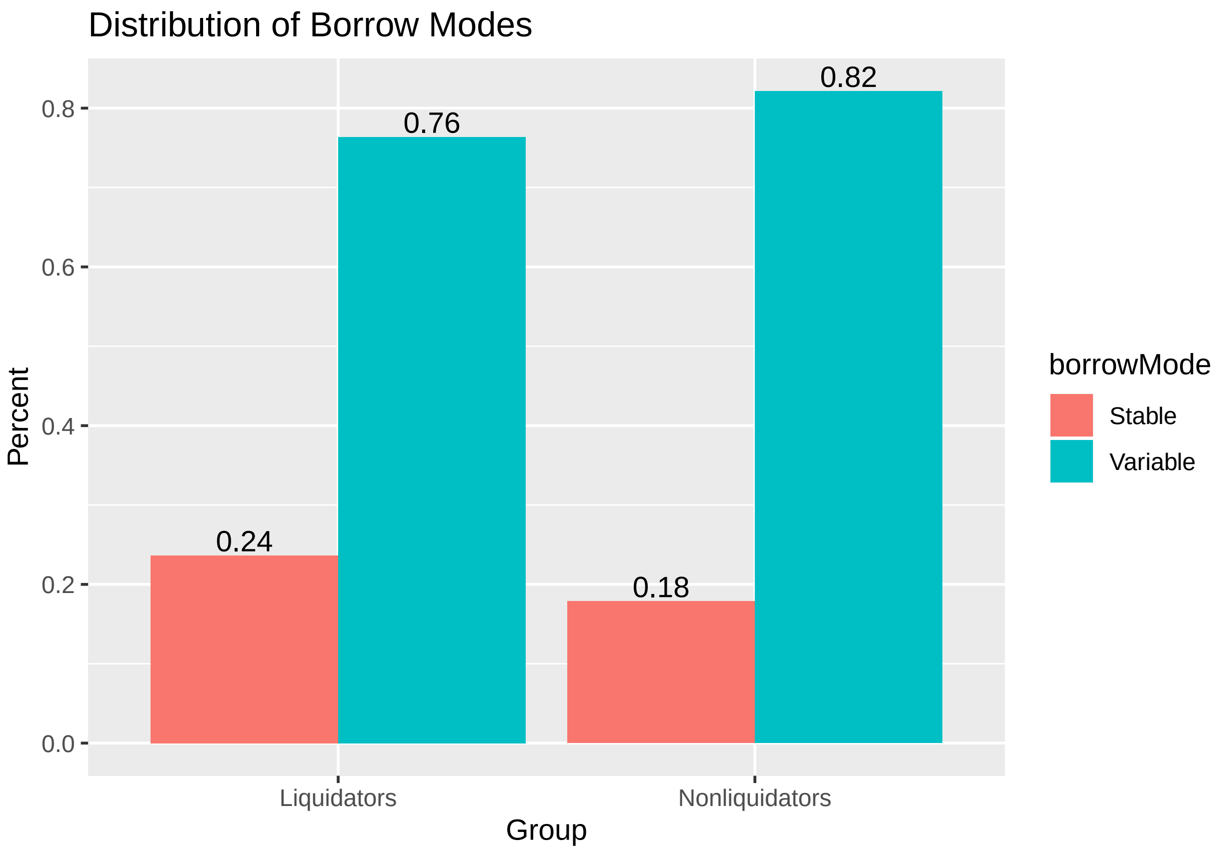
```r
# Compare the borrow mode of liquidators and non-liquidators

# Number of users for each option (Ex. borrower who's a liquidator and chose a stable rate)
stable_liquid <- nrow(liquidators_df[liquidators_df$borrowRateMode == "Stable",])
variable_liquid <- nrow(liquidators_df[liquidators_df$borrowRateMode == "Variable",])
stable_nonliquid <- nrow(nonliquidators_df[nonliquidators_df$borrowRateMode == "Stable",])
variable_nonliquid <- nrow(nonliquidators_df[nonliquidators_df$borrowRateMode == "Variable",])

sum_liquid <- stable_liquid + variable_liquid
sum_nonliquid <- stable_nonliquid + variable_nonliquid
```

```r
# Create a dataframe for the distribution of borrow modes between the two groups
borrowing_df <- data.frame(
  name = c("Liquidators", "Liquidators", "Nonliquidators", "Nonliquidators"),
  borrowMode = c("Stable", "Variable", "Stable", "Variable"),
  value = c(stable_liquid/sum_liquid, variable_liquid/sum_liquid, stable_nonliquid/sum_nonliquid, varia
  )

# Plot as a bar chart
ggplot(borrowing_df, aes(factor(name), value, fill = borrowMode)) +
  geom_bar(stat="identity", position = "dodge") +
  geom_text(aes(label = round(value, digits = 2)), vjust = -0.2,
            position = position_dodge(0.9)) +
  xlab("Group") +
  ylab("Percent") +
  ggtitle("Distribution of Borrow Modes")
```

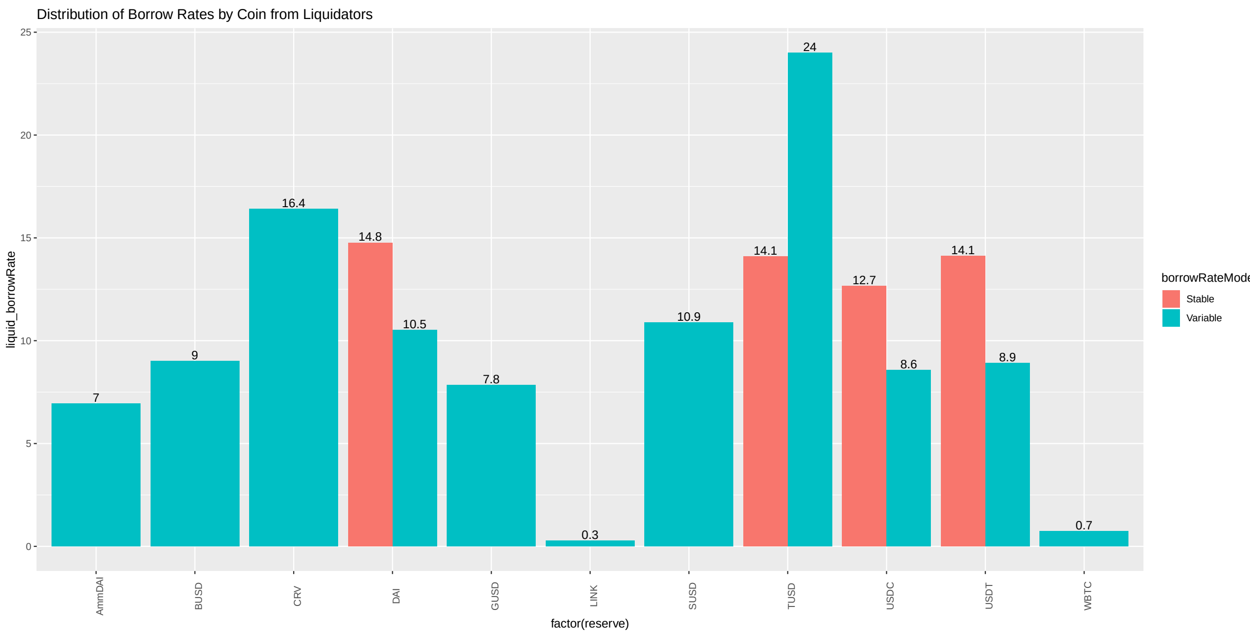


Liquidating users use stable rates slightly more than nonliquidating users do. I don't believe it's significant enough to be a feature in a prediction model.

```r
# Get all borrow rows from liquidator dataframe
liquidators_borrows_df <- liquidators_df[liquidators_df$type == "borrow",]

# Calculate mean borrow rate of coins with more than 100 borrows
liquidators_borrows_df <- liquidators_borrows_df %>%
  group_by(reserve, borrowRateMode) %>%
  filter(n() > 100) %>%
  summarize(liquid_borrowRate = mean(borrowRate))
```

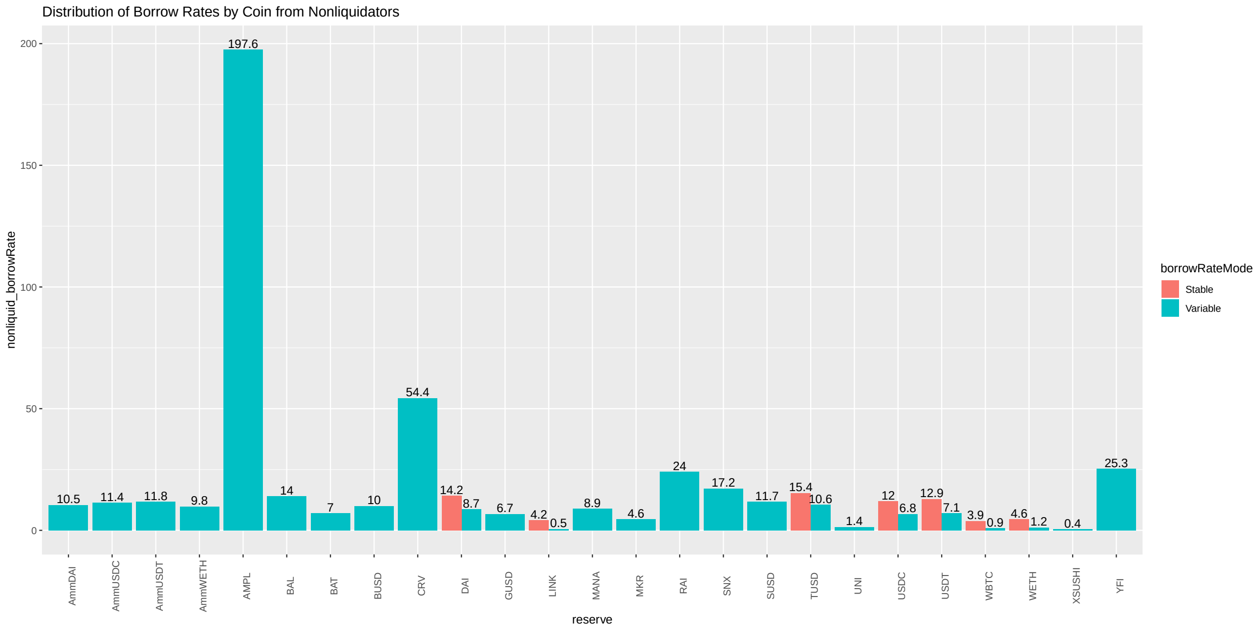
```r
# Plot borrow rates by coin of liquidators
ggplot(liquidators_borrows_df, aes(factor(reserve), liquid_borrowRate, fill = borrowRateMode)) +
  geom_bar(stat="identity", position = "dodge") +
  geom_text(aes(label = round(liquid_borrowRate, digits = 1)), vjust = -0.2,
            position = position_dodge(0.9)) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Distribution of Borrow Rates by Coin from Liquidators")
```




```r
# Get all borrow rows from nonliquidator dataframe
nonliquidators_borrows_df <- nonliquidators_df[nonliquidators_df$type == "borrow",]

# Calculate mean borrow rate of coins with more than 100 borrows
nonliquidators_borrows_df <- nonliquidators_borrows_df %>%
  group_by(reserve, borrowRateMode) %>%
  filter(n() > 100) %>%
  summarize(nonliquid_borrowRate = mean(borrowRate))

# Plot borrow rates by coin of liquidators
ggplot(nonliquidators_borrows_df, aes(factor(reserve), nonliquid_borrowRate, fill = borrowRateMode)) +
  geom_bar(stat="identity", position = "dodge") +
  geom_text(aes(label = round(nonliquid_borrowRate, digits = 1)), vjust = -0.2,
            position = position_dodge(0.9)) +
  theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("Distribution of Borrow Rates by Coin from Nonliquidators") +
  xlab("reserve")
```




```r
borrows_df <- merge(liquidators_borrows_df, nonliquidators_borrows_df)
borrows_df
```

```
##    reserve borrowRateMode liquid_borrowRate nonliquid_borrowRate
## 1   AmmDAI       Variable         6.9513928           10.4665914
## 2     BUSD       Variable         9.0032064           10.0361790
## 3      CRV       Variable        16.4032715           54.3942738
## 4      DAI         Stable        14.7689565           14.2044563
## 5      DAI       Variable        10.5307445            8.6595154
## 6     GUSD       Variable         7.8359176            6.7182073
## 7     LINK       Variable         0.2823775            0.5386345
## 8     SUSD       Variable        10.8875389           11.7072866
## 9     TUSD         Stable        14.0890245           15.3568824
## 10    TUSD       Variable        23.9958667           10.5712956
## 11    USDC         Stable        12.6737056           11.9604741
## 12    USDC       Variable         8.5751221            6.7600944
## 13    USDT         Stable        14.1261681           12.9144506
## 14    USDT       Variable         8.9131298            7.0503096
## 15    WBTC       Variable         0.7466791            0.9185590
```

I find the bar plots tough to compare visually. Looking at the dataframe, I notice borrow rates for liquidating users are higher for the three most popular coins (DAI, USDC, USDT). It makes sense for liquidating users to take on higher rates and I will consider using borrow rates for those coins as features in the prediction model.

```r
# Create dataframe to be used in machine learning models

# Add column for unqiue users in liquidator and nonliquidator groups
df_all <- data.frame(
  user = c(liquidators, nonliquidators)
)
# Create column labeling the group the user is associated with (string and binary)
df_all$value <- ifelse(df_all$user %in% liquidators, 1, 0)
df_all$group <- ifelse(df_all$user %in% liquidators, "Liquidator", "Nonliquidator")
head(df_all, 2)
```

```
##           user value      group
## 1 9.434081e+47     1 Liquidator
## 2 8.824112e+47     1 Liquidator
```

```r
# Create borrow and repay dataframes by user for df_all
liquid_borrows <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "borrow") %>%
  summarise(borrows = n())
liquid_repays <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "repay") %>%
  summarise(repays = n())
nonliquid_borrows <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "borrow") %>%
  summarise(borrows = n())
nonliquid_repays <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "repay") %>%
  summarise(repays = n())
user_borrows <- rbind(liquid_borrows, nonliquid_borrows)
user_repays <- rbind(liquid_repays, nonliquid_repays)
```

```r
# Add user repay and borrows to df_all
df_all <- merge(df_all, user_borrows, all = TRUE)
df_all <- merge(df_all, user_repays, all = TRUE)
# Set na values to 0
df_all[is.na(df_all)] <- 0
head(df_all, 2)
```

```
##           user value         group borrows repays
## 1 2.577533e+33     0 Nonliquidator       0      0
## 2 6.663597e+34     0 Nonliquidator       2      2
```

```r
# Create variable and stable borrow rate dataframes for users
# Coins selected were DAI, USDC, USDT
stable_rate_dai <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Stable" & reserve == as.character("DAI")) %>%
  summarize(mean_stable_rate_dai = mean(borrowRate))
stable_rate_usdc <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Stable" & reserve == as.character("USDC")) %>%
  summarize(mean_stable_rate_usdc = mean(borrowRate))
stable_rate_usdt <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Stable" & reserve == as.character("USDT")) %>%
  summarize(mean_stable_rate_usdt = mean(borrowRate))
variable_rate_dai <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Variable" & reserve == as.character("DAI")) %>%
  summarize(mean_variable_rate_dai = mean(borrowRate))
variable_rate_usdc <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Variable" & reserve == as.character("USDC")) %>%
```

```r
  summarize(mean_variable_rate_usdc = mean(borrowRate))
variable_rate_usdt <- df %>%
  group_by(user) %>%
  filter(type == "borrow" & borrowRateMode == "Variable" & reserve == as.character("USDT")) %>%
  summarize(mean_variable_rate_usdt = mean(borrowRate))
```

```r
# Join dataframes with borrow rates as columns in df_all
df_all <- merge(df_all, stable_rate_dai, all = TRUE)
df_all <- merge(df_all, variable_rate_dai, all = TRUE)
df_all <- merge(df_all, stable_rate_usdc, all = TRUE)
df_all <- merge(df_all, variable_rate_usdc, all = TRUE)
df_all <- merge(df_all, stable_rate_usdt, all = TRUE)
df_all <- merge(df_all, variable_rate_usdt, all = TRUE)
# Omit users who haven't borrowed DAI, USDC, USDT
df_all <- na.omit(df_all)
head(df_all, 2)
```

```
##             user value         group borrows repays mean_stable_rate_dai
## 382 1.148328e+46     0 Nonliquidator      35     52            11.934031
## 657 2.025163e+46     0 Nonliquidator      10      6             9.898803
##     mean_variable_rate_dai mean_stable_rate_usdc mean_variable_rate_usdc
## 382               3.993179             10.697962                3.154049
## 657              19.081225              8.921656               14.971844
##     mean_stable_rate_usdt mean_variable_rate_usdt
## 382             11.655556                3.211726
## 657              8.992857                7.814606
```

```r
# Omit user 71
df_all <- df_all[-c(71),]
```

After running the logistic regression model, we get very skewed results so I removed user 71 to handle the outlier.

```r
# Separate data into training a testing sets
train_df <- df_all[1:(nrow(df_all) * .75),]
train_df <- train_df[c("value", "borrows", "repays", "mean_stable_rate_dai",
                       "mean_variable_rate_dai", "mean_stable_rate_usdc",
                       "mean_variable_rate_usdc", "mean_stable_rate_usdt",
                       "mean_variable_rate_usdt")]
test_df <- df_all[((nrow(df_all) * .75) + 1):nrow(df_all),]
test_df <- test_df[c("value", "borrows", "repays", "mean_stable_rate_dai",
                       "mean_variable_rate_dai", "mean_stable_rate_usdc",
                       "mean_variable_rate_usdc", "mean_stable_rate_usdt",
                       "mean_variable_rate_usdt")]
```

I chose a 75-25 split because the sample I have isn't very large. For my next notebook, I'll increase the size of the sample to help with evaluating the model. I originally shrunk the sample to accommodate more features but those ended up not being needed.

```r
logit_all <- glm(value ~ borrows + repays + mean_stable_rate_dai +
                       mean_variable_rate_dai + mean_stable_rate_usdc +
                       mean_variable_rate_usdc + mean_stable_rate_usdt +
                       mean_variable_rate_usdt, family = binomial, data = train_df)
summary(logit_all)
```

```
##
```

```
## Call:
## glm(formula = value ~ borrows + repays + mean_stable_rate_dai +
##     mean_variable_rate_dai + mean_stable_rate_usdc + mean_variable_rate_usdc +
##     mean_stable_rate_usdt + mean_variable_rate_usdt, family = binomial,
##     data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2628  -0.7103  -0.2595   0.7783   2.2589
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              0.851392   2.756477   0.309  0.75742
## borrows                  0.080312   0.025627   3.134  0.00173 **
## repays                  -0.058520   0.019647  -2.979  0.00290 **
## mean_stable_rate_dai    -0.038288   0.102172  -0.375  0.70786
## mean_variable_rate_dai  -0.043972   0.056719  -0.775  0.43818
## mean_stable_rate_usdc   -0.320195   0.289236  -1.107  0.26828
## mean_variable_rate_usdc  0.114067   0.067900   1.680  0.09297 .
## mean_stable_rate_usdt   -0.001658   0.109795  -0.015  0.98795
## mean_variable_rate_usdt  0.026426   0.052222   0.506  0.61284
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 78.580  on 56  degrees of freedom
## Residual deviance: 53.062  on 48  degrees of freedom
## AIC: 71.062
##
## Number of Fisher Scoring iterations: 5
```

As shown from the looking at the significance of the z scores, borrows and repays make the greatest impact in the model by a wide margin. That's why I decided to just use those two features in my final logical regression and random forest models.

```r
# Run logistic regression on training data
logit <- glm(value ~ borrows + repays, family = binomial, data = train_df)
```

```r
# Predict logistic regression on testing data
# Created dataframe with predicted and expected values
logit_prediction <- predict(logit, newdata = test_df[c(-1)], type = "response")
logit_classify <- round(logit_prediction, digits = 0)
logit_df <- data.frame(
  predicted = logit_classify,
  actual = test_df[c(1)]
)
```

```r
# Display results of logistic regression
logit_df %>%
  count(predicted == value)
```

```
## # A tibble: 2 x 2
##   `predicted == value`     n
##   <lgl>                <int>
## 1 FALSE                    5
```

```
## 2 TRUE                    14
```

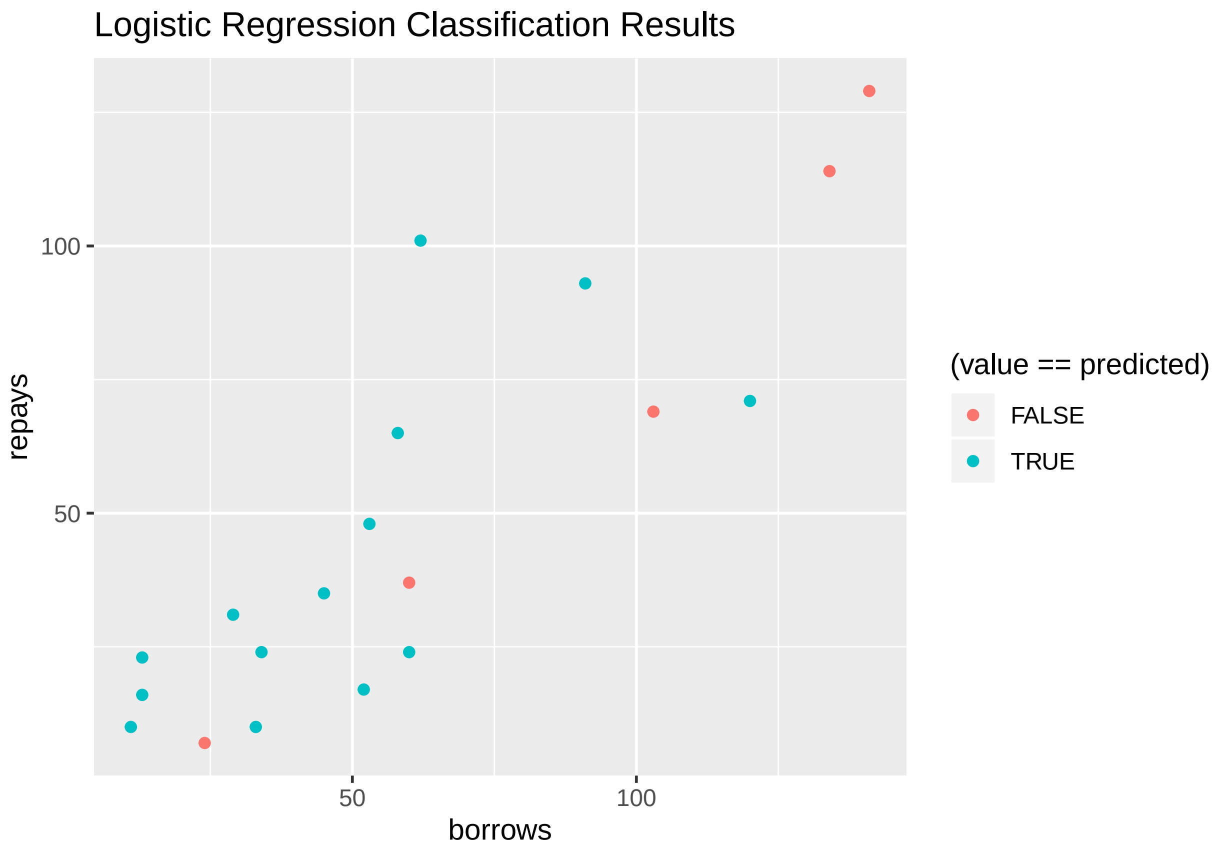
The success rate for logistic regression is 14/19 (73.7%).

```r
confusionMatrix(table(logit_classify, test_df[[c(1)]]))
```

```
## Confusion Matrix and Statistics
##
##
## logit_classify  0  1
##              0 10  1
##              1  4  4
##
##                Accuracy : 0.7368
##                  95% CI : (0.488, 0.9085)
##     No Information Rate : 0.7368
##     P-Value [Acc > NIR] : 0.6173
##
##                   Kappa : 0.4311
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.7143
##             Specificity : 0.8000
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7368
##          Detection Rate : 0.5263
##    Detection Prevalence : 0.5789
##       Balanced Accuracy : 0.7571
##
##        'Positive' Class : 0
##
```

Looking at the confusion matrix, 10 users who haven't been liquidated and 4 users who have been liquidated were successfully predicted. There was 1 liquidator who was predicted as a nonliquidator. Alternatively, there were 4 nonliquidators predicted as liquidators.

```r
# Plot the results of the logistic regression model
logit_results <- test_df
logit_results$predicted <- logit_classify
ggplot() +
  geom_point(data = logit_results,
             mapping = aes(x = borrows,
                           y = repays,
                           colour = (value == predicted))) +
  ggtitle("Logistic Regression Classification Results")
```



This plot visualizes the users who were predicted incorrectly. It's interesting to see the points follow a trendline closely if it were drawn.

```r
# Run random forest prediction model on same data and aggregate results
rf = randomForest(x = train_df[c("borrows", "repays")],
                        y = train_df$value,
                        ntree = 500, random_state = 0)
```

```
## Warning in randomForest.default(x = train_df[c("borrows", "repays")], y =
## train_df$value, : The response has five or fewer unique values. Are you sure you
## want to do regression?
```

```r
rf_prediction = predict(rf, newdata = test_df[c("borrows", "repays")])
rf_classify = round(rf_prediction, digits = 0)
rf_df <- data.frame(
  predicted = rf_classify,
  actual = test_df[c(1)]
)
```

```r
# Display results of random forest prediction
rf_df %>%
  count(predicted == value)
```

```
## # A tibble: 2 x 2
##   `predicted == value`     n
##   <lgl>                <int>
## 1 FALSE                    1
## 2 TRUE                    18
```

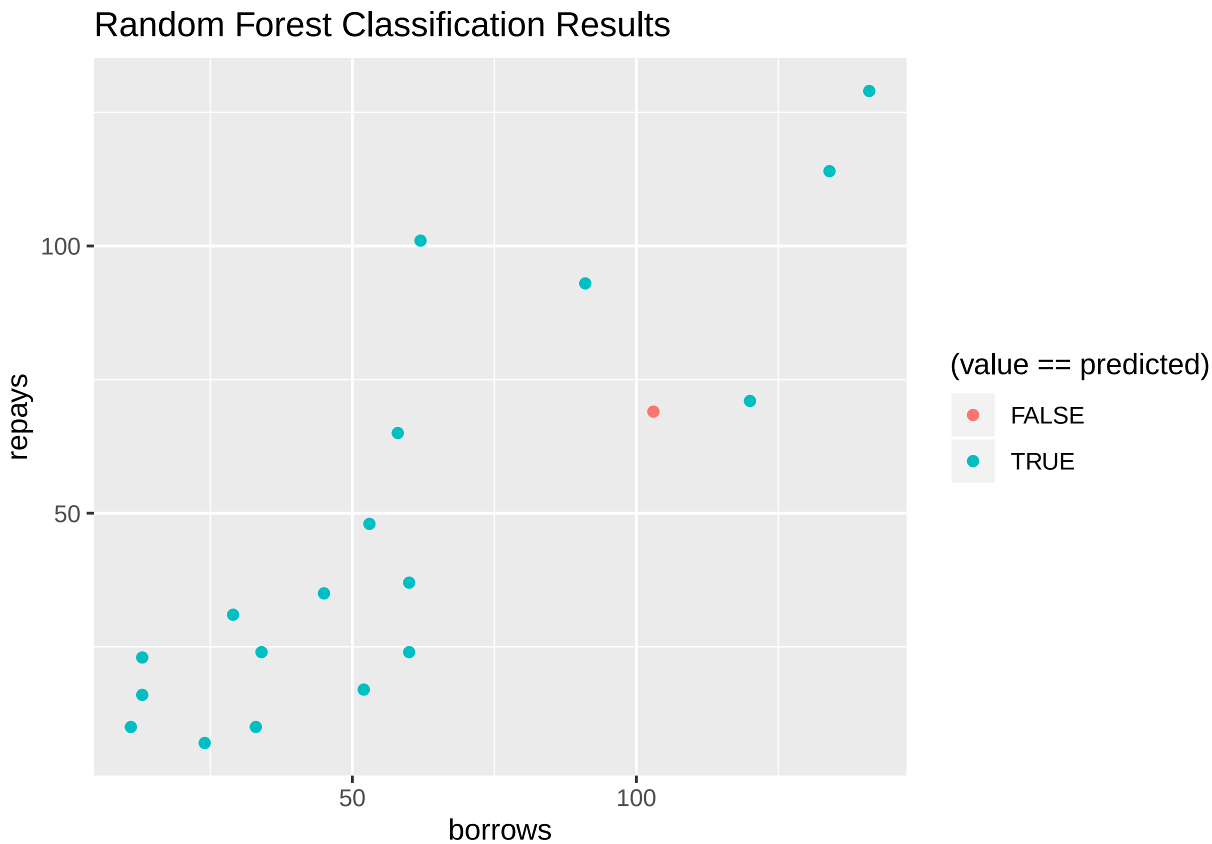
The success rate for random forest is 18/19 (94.7%).

```r
confusionMatrix(table(rf_classify, test_df[[c(1)]]))
```

```
## Confusion Matrix and Statistics
##
##
## rf_classify  0  1
##           0 13  0
##           1  1  5
##
##                Accuracy : 0.9474
##                  95% CI : (0.7397, 0.9987)
##     No Information Rate : 0.7368
##     P-Value [Acc > NIR] : 0.02352
##
##                   Kappa : 0.8725
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.9286
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8333
##              Prevalence : 0.7368
##          Detection Rate : 0.6842
##    Detection Prevalence : 0.6842
##       Balanced Accuracy : 0.9643
##
##        'Positive' Class : 0
##
```

1 nonliquidator was predicted to be a liquidator.

```r
# Plot the results of the random forest prediction model
rf_results <- test_df
rf_results$predicted <- rf_classify
ggplot() +
  geom_point(data = rf_results,
             mapping = aes(x = borrows,
                                     y = repays,
                                     colour = (value == predicted))) +
  ggtitle("Random Forest Classification Results")
```



The falsely predicted value was also predicted incorrectly in the logistic regression model.

Conclusion

Overall, this notebook is a good first step in solving a complicated problem. From my exploratory data analysis, I thought borrow rates would have greater importance in my models. Accommodating these features made the sample of users small. The sample used was certainly credible because it contained active users that borrowed with three different coins using stable and variable borrow rates. My next notebook will use an expanded sample. I will look into evaluating user histories to declare them fit for prediction. Borrows and repays were strong features and I look forward to seeing how they perform on more data. There is likely more feature engineering that will be needed. Ultimately, a 94.7% success rate with the random forest model is an encouraging start to solving this problem. I hope to further quantify the reliability of this model with my next notebook.