DAR F21 Project Status Notebook Assignment 5

DeFi

Jason Podgorski (GitHub: podgoj)

10/17/2021

Contents

Summary of Work 1

GitHub Commits 1

Personal Contribution 1

Primary Findings 2

The Code 2
- Predicting if a User Will Liquidate . . . 2
- Who are the Bad Borrowers in AAVE? . . . 10

Conclusion 25

Summary of Work

For notebook 5, I wanted to focus on two areas. First, I wanted to continue my investigation of predicting if users will liquidate. I wanted to expand the data sample to more users and try additional features such as the number of deposits, redeems, and swaps the user has. Next, I wanted to quantify good vs. bad borrows in AAVE. I examined three stable coins: USDC, USDT and DAI. My goal was to see which users borrowed when interest rates were at their highest and see if those users were forced to liquidate shortly after.

GitHub Commits

Branch Name: dar-podgoj

Files on GitHub:

podgoj_assignment05.Rmd

podgoj_assignment05.pdf

podgoj_assignment05.html

Personal Contribution

All of the work in this notebook is my own.

Primary Findings

For predicting who will be liquidated in the future, my model needs a feature overhaul. User health factor should be the primary way of deciding if a user will be liquidated in the near future. I also believe I need to balance the number of users who have/haven't been liquidated in my sample. Decision variables with an extremely unbalanced ratio can lead to an overfit model.

By looking at bad borrowers and days with high interest rates, I was happy with my analysis of users that consistently made bad borrows. By focusing on a small population of users, I was able to compare transaction histories, borrowing habits, borrowing amounts, and liquidations. I was able to find superusers that borrowed a lot and repayed little. I also learned many of these borrowers have borrowed 10+ million USD. Since many of my teammates were looking into clustering, I thought my "manually clustering" was an interesting way to compare users with similar patterns and defined constraints.

The Code

```
# load Rds (binary version of csv file) into dataframe
df <- read_rds('../../Data/transactionsv2.rds')
tail(df, 2)
```

```
##        amount borrowRate borrowRateMode onBehalfOf         pool reserve
## 745611     NA         NA                        NA 1.034668e+48    WBTC
## 745612     NA         NA                        NA 1.034668e+48     CRV
##         timestamp         user       type reservePriceETH reservePriceUSD
## 745611 1619057823 1.251533e+48 collateral              NA              NA
## 745612 1610607551 4.198858e+47 collateral              NA              NA
##        amountUSD collateralAmount collateralReserve liquidator principalAmount
## 745611        NA               NA                           NA              NA
## 745612        NA               NA                           NA              NA
##        principalReserve reservePriceETHPrincipal reservePriceUSDPrincipal
## 745611                                        NA                       NA
## 745612                                        NA                       NA
##        reservePriceETHCollateral reservePriceUSDCollateral amountUSDPincipal
## 745611                        NA                        NA                NA
## 745612                        NA                        NA                NA
##        amountUSDCollateral borrowRateModeFrom borrowRateModeTo stableBorrowRate
## 745611                  NA                                                  NA
## 745612                  NA                                                  NA
##        variableBorrowRate fromState toState protocolContract     user_alias
## 745611                 NA      True   False            False Elizabeth Ruiz
## 745612                 NA      True   False            False  Shela Hazzard
##        onBehalfOf_alias            datetime
## 745611        John Dunn 2021-04-22 02:17:03
## 745612        John Dunn 2021-01-14 06:59:11
```

```
# 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)
```

Predicting if a User Will Liquidate

In my last notebook, I attempted to predict if a user has liquidated in their past. I used logistic regression and random forest models with the number of repays and borrows per user, as well as the interest rates they typically borrow at. The interest rates surprisingly had little significance compared to the borrow and repay features. With a smaller sample, the model showed promise. I am going to try to replicate the same models with a larger population of users and see if the results were as strong as previously.

Notebook 4: podgoj_assignment04.pdf

```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
# 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, deposit, redeem, repay, and swap data frames by user for df_all
liquid_borrows <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "borrow") %>%
  summarise(borrows = n())
liquid_deposits <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "deposit") %>%
  summarise(deposits = n())
liquid_redeems <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "redeem") %>%
  summarise(redeems = n())
liquid_repays <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "repay") %>%
  summarise(repays = n())
liquid_swaps <- liquidators_df %>%
  group_by(user) %>%
  filter(type == "swap") %>%
  summarise(swaps = n())
nonliquid_borrows <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "borrow") %>%
  summarise(borrows = n())
nonliquid_deposits <- nonliquidators_df %>%
```

```r
  group_by(user) %>%
  filter(type == "deposit") %>%
  summarise(deposits = n())
nonliquid_redeems <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "redeem") %>%
  summarise(redeems = n())
nonliquid_repays <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "repay") %>%
  summarise(repays = n())
nonliquid_swaps <- nonliquidators_df %>%
  group_by(user) %>%
  filter(type == "swaps") %>%
  summarise(swaps = n())
# merge liquid and nonliquid transactions
user_borrows <- rbind(liquid_borrows, nonliquid_borrows)
user_deposits <- rbind(liquid_deposits, nonliquid_deposits)
user_redeems <- rbind(liquid_redeems, nonliquid_redeems)
user_repays <- rbind(liquid_repays, nonliquid_repays)
user_swaps <- rbind(liquid_swaps, nonliquid_swaps)
head(user_borrows)
```

```
## # A tibble: 6 x 2
##      user borrows
##     <dbl>   <int>
## 1 1.33e44      16
## 2 3.86e44       4
## 3 5.89e44      24
## 4 2.08e45       1
## 5 2.16e45       2
## 6 2.88e45       5
```

```r
# add transaction dataframes to df_all
df_all <- merge(df_all, user_borrows, all = TRUE)
df_all <- merge(df_all, user_deposits, all = TRUE)
df_all <- merge(df_all, user_redeems, all = TRUE)
df_all <- merge(df_all, user_repays, all = TRUE)
df_all <- merge(df_all, user_swaps, all = TRUE)
# set na values to 0
df_all[is.na(df_all)] <- 0
head(df_all, 2)
```

```
##    user value         group borrows deposits redeems repays swaps
## 1     1     0 Nonliquidator       0        0       0      0     0
## 2 57005     0 Nonliquidator       0        0       0      0     0
```

```r
# eliminate users that have an outlier number of borrows (500+)
df_all <- df_all[df_all$borrows > 1 & df_all$borrows < 500,]
```

The first set of features I try in the prediction model are the number of borrows and repays per user. I later add the other transaction types to compare the effectiveness of the model.

```r
# separate data into training a testing sets
train_df <- df_all[1:(nrow(df_all) * .8),]
train_df <- train_df[c("value", "borrows", "deposits", "redeems", "repays", "swaps")]
```

```r
test_df <- df_all[((nrow(df_all) * .8) + 1):nrow(df_all),]
test_df <- test_df[c("value", "borrows", "deposits", "redeems", "repays", "swaps")]
```

Logistic Regression

```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                  308
## 2 TRUE                  1874
```

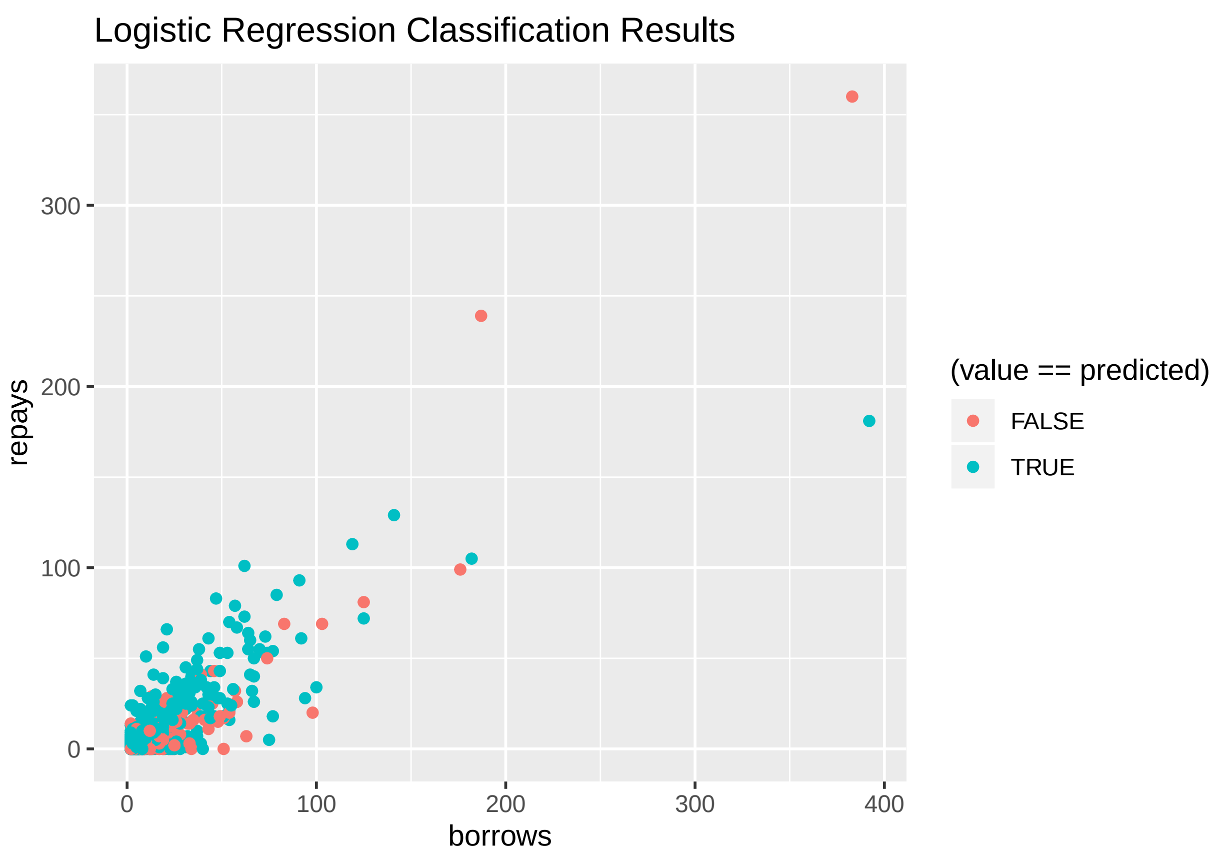
```r
confusionMatrix(table(logit_classify, test_df[[c(1)]]))
```

```
## Confusion Matrix and Statistics
##
##
## logit_classify    0    1
##              0 1848  290
##              1   18   26
##
##                Accuracy : 0.8588
##                  95% CI : (0.8435, 0.8732)
##     No Information Rate : 0.8552
##     P-Value [Acc > NIR] : 0.3262
##
##                   Kappa : 0.113
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99035
##             Specificity : 0.08228
##          Pos Pred Value : 0.86436
##          Neg Pred Value : 0.59091
##              Prevalence : 0.85518
##          Detection Rate : 0.84693
##    Detection Prevalence : 0.97984
##       Balanced Accuracy : 0.53632
##
##        'Positive' Class : 0
```

```
##
```

Looking at the confusion matrix, our balanced accuracy is 53.6% which is similar to straight up guessing. The overall accuracy is 85.9% which is extremely over-inflated because around 95% of the sample are users who haven't been liquidated yet. The model therefore assumes almost everyone is in that group.

```
# 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")
```



Random Forest

```
# train random forest model
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?
```

```
# run random forest prediction model and aggregate results
rf_prediction = predict(rf, newdata = test_df[c("borrows", "repays")])
```

```r
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                  309
## 2 TRUE                  1873
```

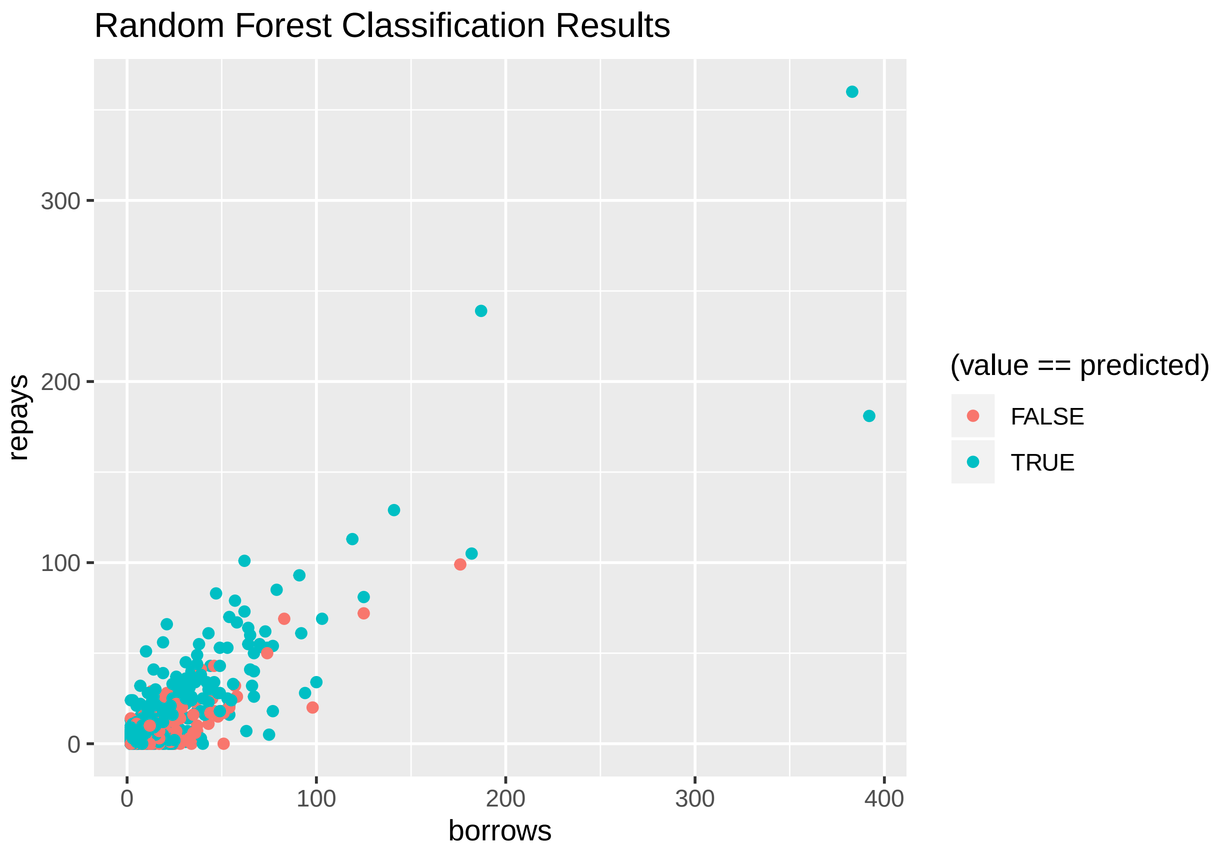
```r
confusionMatrix(table(rf_classify, test_df[[c(1)]]))
```

```
## Confusion Matrix and Statistics
##
##
## rf_classify    0    1
##           0 1837  280
##           1   29   36
##
##                Accuracy : 0.8584
##                  95% CI : (0.843, 0.8728)
##     No Information Rate : 0.8552
##     P-Value [Acc > NIR] : 0.3485
##
##                   Kappa : 0.1468
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9845
##             Specificity : 0.1139
##          Pos Pred Value : 0.8677
##          Neg Pred Value : 0.5538
##              Prevalence : 0.8552
##          Detection Rate : 0.8419
##    Detection Prevalence : 0.9702
##       Balanced Accuracy : 0.5492
##
##        'Positive' Class : 0
##
```

Similarly to the logistic regression model, the random forest model does no better than a coin flip despite the balanced accuracy being high.

```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,
```

```r
                                    colour = (value == predicted))) +
  ggtitle("Random Forest Classification Results")
```




```r
# run logistic regression on training data with all transaction types as a features
logit_all <- glm(value ~ borrows + deposits + redeems + repays + swaps, family = binomial, data = train_
```

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

```r
summary(logit_all)
```

```
##
## Call:
## glm(formula = value ~ borrows + deposits + redeems + repays +
##     swaps, family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.1224  -0.4973  -0.4568  -0.3966   4.3433
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.220965   0.041445 -53.588  < 2e-16 ***
## borrows       0.068301   0.005294  12.902  < 2e-16 ***
## deposits      0.037271   0.006027   6.184 6.25e-10 ***
## redeems      -0.055815   0.008859  -6.300 2.97e-10 ***
## repays       -0.086553   0.008205 -10.548  < 2e-16 ***
## swaps        19.056928 190.654372   0.100     0.92
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7216.9  on 8728  degrees of freedom
## Residual deviance: 5829.2  on 8723  degrees of freedom
## AIC: 5841.2
##
## Number of Fisher Scoring iterations: 18
```

```r
# predict logistic regression on testing data
# create data frame with predicted and expected values
logit_all_prediction <- predict(logit_all, newdata = test_df[c(-1)], type = "response")
logit_all_classify <- round(logit_all_prediction, digits = 0)
logit_all_df <- data.frame(
  predicted = logit_all_classify,
  actual = test_df[c(1)]
)
```

```r
# display results of logistic regression
logit_all_df %>%
  count(predicted == value)
```

```
## # A tibble: 2 x 2
##   `predicted == value`     n
##   <lgl>                <int>
## 1 FALSE                  258
## 2 TRUE                  1924
```

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

```
## Confusion Matrix and Statistics
##
##
## logit_all_classify    0    1
##                  0 1854  246
##                  1   12   70
##
##                Accuracy : 0.8818
##                  95% CI : (0.8675, 0.895)
##     No Information Rate : 0.8552
##     P-Value [Acc > NIR] : 0.0001677
##
##                   Kappa : 0.3106
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9936
##             Specificity : 0.2215
##          Pos Pred Value : 0.8829
##          Neg Pred Value : 0.8537
##              Prevalence : 0.8552
##          Detection Rate : 0.8497
##    Detection Prevalence : 0.9624
##       Balanced Accuracy : 0.6075
##
```

```
##        'Positive' Class : 0
##
```

There is improved balanced accuracy but I believe I need to go in a different direction with feature creation and selection.

Who are the Bad Borrowers in AAVE?

```r
# load interest rates data
rates_df <- read_csv('../../Data/rates.csv')
```

```
## Parsed with column specification:
## cols(
##   liquidityRate = col_double(),
##   reserve = col_character(),
##   stableBorrowRate = col_double(),
##   timestamp = col_double(),
##   variableBorrowRate = col_double()
## )
```

```r
tail(rates_df, 2)
```

```
## # A tibble: 2 x 5
##   liquidityRate reserve stableBorrowRate  timestamp variableBorrowRate
##           <dbl> <chr>              <dbl>      <dbl>              <dbl>
## 1       4.54    USDC               12.5   1632924677              5.53
## 2       0.00598 WETH                3.31  1632924677              0.250
```

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

```r
# calculate median variable and stable rates for USDC
usdc_stableRates <- df %>%
  group_by(date) %>%
  filter(reserve == as.character("USDC") & borrowRateMode == "Stable") %>%
  summarize(stableRate = median(borrowRate))
usdc_variableRates <- df %>%
  group_by(date) %>%
  filter(reserve == as.character("USDC") & borrowRateMode == "Variable") %>%
  summarize(variableRate = median(borrowRate))
usdc_rates <- merge(usdc_stableRates, usdc_variableRates)
head(usdc_rates)
```

```
##         date stableRate variableRate
## 1 2020-12-03   5.333889     3.923204
## 2 2020-12-04   5.819460    11.284508
## 3 2020-12-05   5.855355     3.839515
## 4 2020-12-06   5.978114     3.998428
## 5 2020-12-07   5.675690     3.344663
## 6 2020-12-08   5.840489     3.897950
```

```r
# break date into month, week, and day for time series heatmap
usdc_rates$year <- format(usdc_rates$date, format = "%Y")
usdc_rates$month <- month.abb[month(usdc_rates$date)]
usdc_rates$monthweek <- ceiling(day(usdc_rates$date) / 7)
```

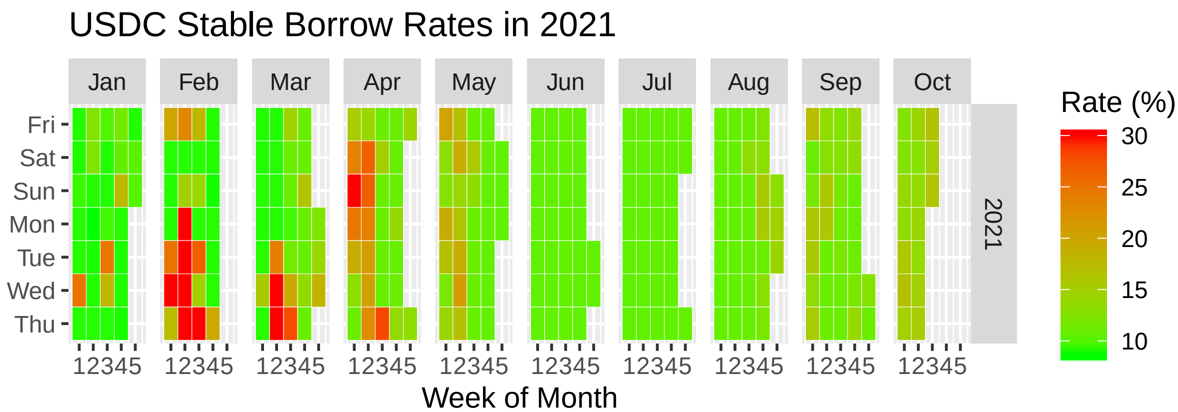
```r
usdc_rates$day <- c("Sun", "Mon", "Tue", "Wed", "Thu",
    "Fri", "Sat")[as.POSIXlt(usdc_rates$date)$wday + 1]
# assign outliers to 30 for heatmap scaling
usdc_rates$stableRate[usdc_rates$stableRate > 30] <- 30
usdc_rates$variableRate[usdc_rates$variableRate > 30] <- 30
usdc_rates <- usdc_rates[usdc_rates$date >= "2021-01-01",]
head(usdc_rates)
```

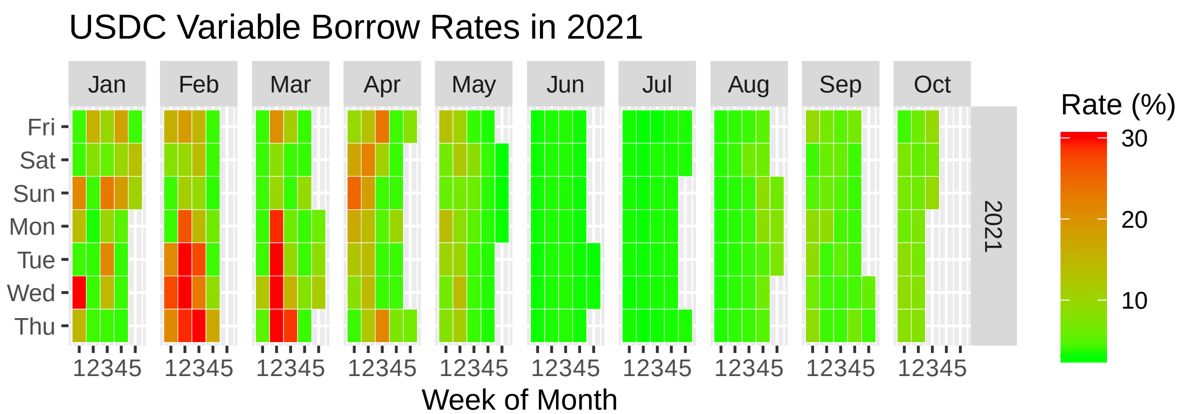
```
##          date stableRate variableRate year month monthweek day
## 30 2021-01-01   8.940408     3.890120 2021   Jan         1 Fri
## 31 2021-01-02   8.917192     3.940635 2021   Jan         1 Sat
## 32 2021-01-03   9.325847    21.221567 2021   Jan         1 Sun
## 33 2021-01-04   8.989028    13.841652 2021   Jan         1 Mon
## 34 2021-01-05   8.969490     3.921793 2021   Jan         1 Tue
## 35 2021-01-06  24.789498    30.000000 2021   Jan         1 Wed
```

```r
# create USDC stable rate time series heatmap
usdc_stable_plot <- ggplot(usdc_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mon"
  geom_tile(colour = "white") +
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="USDC Stable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
# create USDC variable rate time series heatmap
usdc_variable_plot <- ggplot(usdc_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mon
  geom_tile(colour = "white") +
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="USDC Variable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
# combine USDC heatmaps into one visualization
ggarrange(
  usdc_stable_plot, usdc_variable_plot
  )
```





The heatmaps show the variable and stable interest rates of USDC in calendar format. We see that higher rates occur earlier in 2021.

```
# calculate median variable and stable rates for USDT
usdt_stableRates <- df %>%
  group_by(date) %>%
  filter(reserve == as.character("USDT") & borrowRateMode == "Stable") %>%
  summarize(stableRate = median(borrowRate))
usdt_variableRates <- df %>%
  group_by(date) %>%
  filter(reserve == as.character("USDT") & borrowRateMode == "Variable") %>%
  summarize(variableRate = median(borrowRate))
usdt_rates <- merge(usdt_stableRates, usdt_variableRates)
```

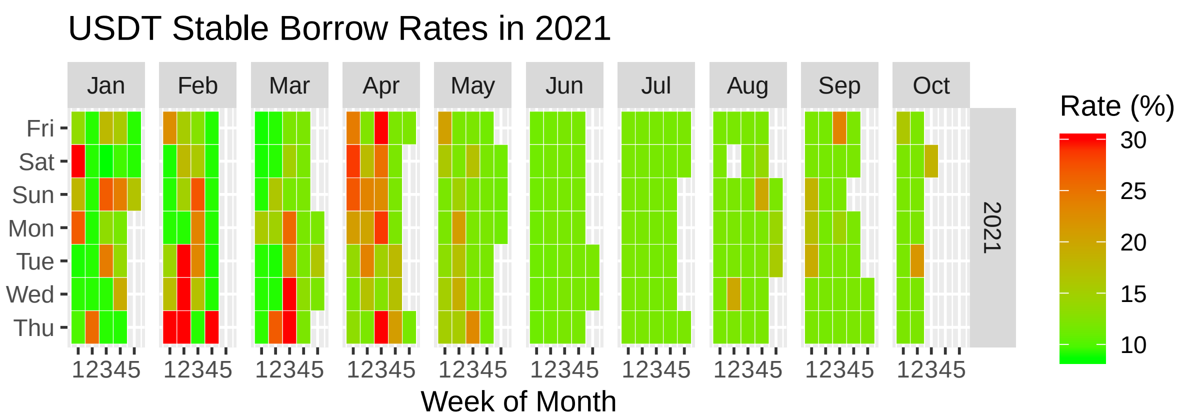
```
# break date into month, week, and day for time series heatmap
usdt_rates$year <- format(usdt_rates$date, format = "%Y")
usdt_rates$month <- month.abb[month(usdt_rates$date)]
usdt_rates$monthweek <- ceiling(day(usdt_rates$date) / 7)
usdt_rates$day <- c("Sun", "Mon", "Tue", "Wed", "Thu",
    "Fri", "Sat")[as.POSIXlt(usdt_rates$date)$wday + 1]
# assign outliers to 30 for heatmap scaling
usdt_rates$stableRate[usdt_rates$stableRate > 30] <- 30
usdt_rates$variableRate[usdt_rates$variableRate > 30] <- 30
usdt_rates <- usdt_rates[usdt_rates$date >= "2021-01-01",]
```

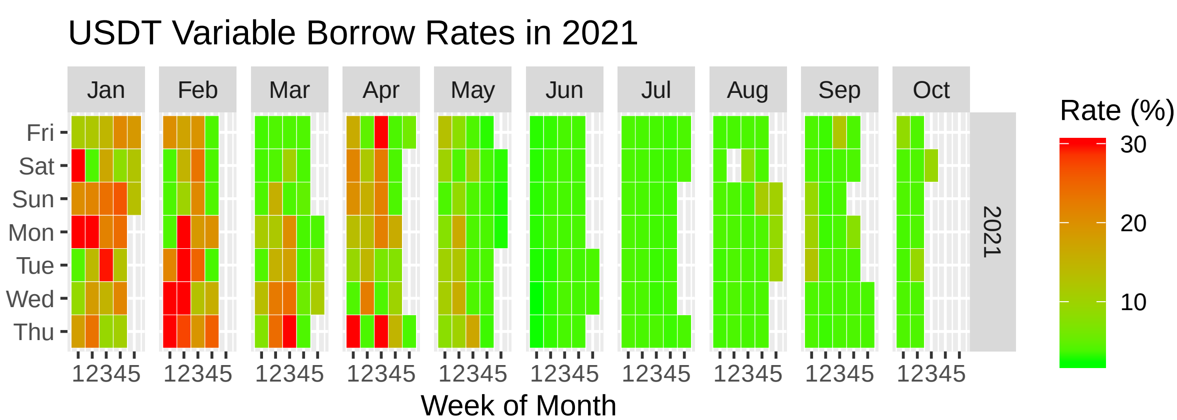
```
# create USDT stable rate time series heatmap
usdt_stable_plot <- ggplot(usdt_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mon"
  geom_tile(colour = "white") +
```

```r
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Se
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="USDT Stable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
# create USDT variable rate time series heatmap
usdt_variable_plot <- ggplot(usdt_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mo
  geom_tile(colour = "white") +
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Se
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="USDT Variable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
# combine USDT heatmaps into one visualization
ggarrange(
  usdt_stable_plot, usdt_variable_plot
)
```





Similarly to USDC, USDT had higher rates in the first few months of 2021.

```r
# calculate median variable and stable rates for DAI
dai_stableRates <- df %>%
```

```r
  group_by(date) %>%
  filter(reserve == as.character("DAI") & borrowRateMode == "Stable") %>%
  summarize(stableRate = median(borrowRate))
dai_variableRates <- df %>%
  group_by(date) %>%
  filter(reserve == as.character("DAI") & borrowRateMode == "Variable") %>%
  summarize(variableRate = median(borrowRate))
dai_rates <- merge(dai_stableRates, dai_variableRates)
head(dai_rates)
```

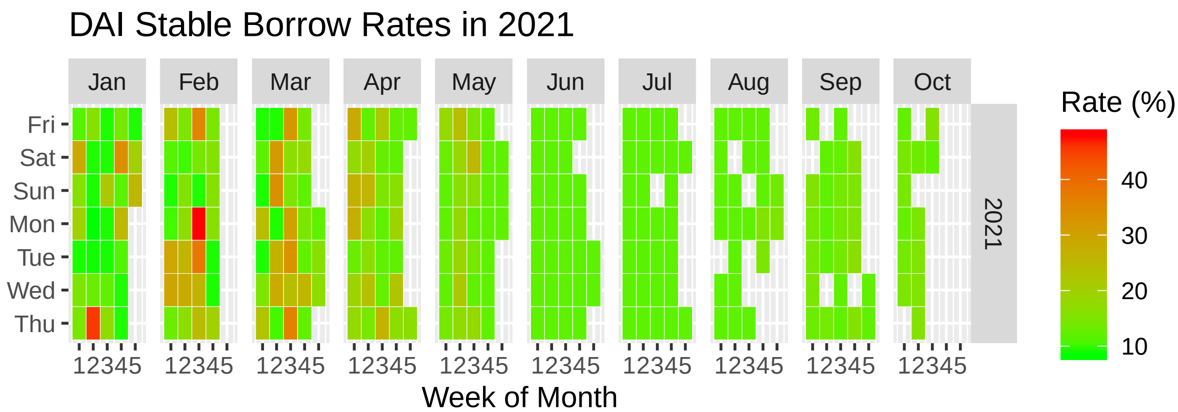
```
##         date stableRate variableRate
## 1 2020-12-03   5.388793     2.807834
## 2 2020-12-04   5.423849     3.557081
## 3 2020-12-05   5.772982     3.636086
## 4 2020-12-06   5.709240     3.854518
## 5 2020-12-07  17.974825    20.615696
## 6 2020-12-08   6.123975    24.348272
```

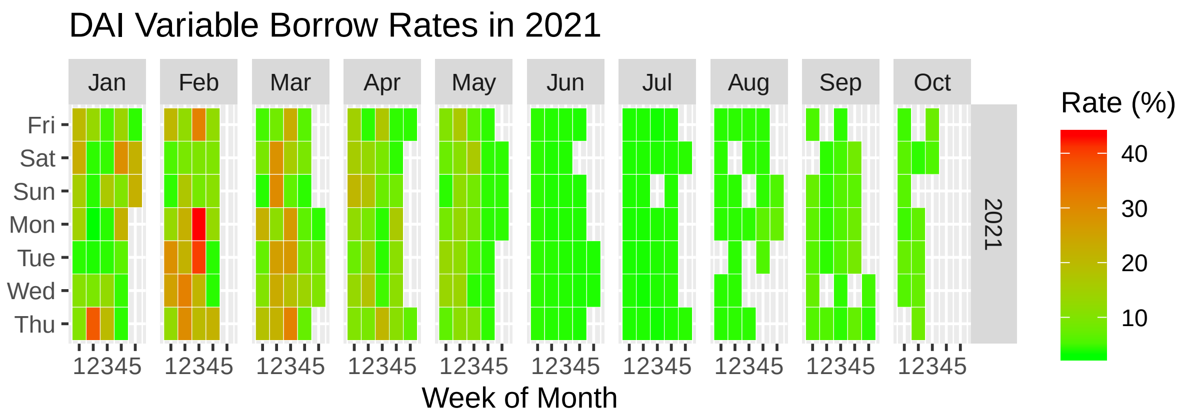
```r
# break date into month, week, and day for time series heatmap
dai_rates$year <- format(dai_rates$date, format = "%Y")
dai_rates$month <- month.abb[month(dai_rates$date)]
dai_rates$monthweek <- ceiling(day(dai_rates$date) / 7)
dai_rates$day <- c("Sun", "Mon", "Tue", "Wed", "Thu",
    "Fri", "Sat")[as.POSIXlt(dai_rates$date)$wday + 1]
dai_rates <- dai_rates[dai_rates$date >= "2021-01-01",]
```

```r
# create DAI stable rate time series heatmap
dai_stable_plot <- ggplot(dai_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mon",
  geom_tile(colour = "white") +
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Se
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="DAI Stable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
# create DAI variable rate time series heatmap
dai_variable_plot <- ggplot(dai_rates, aes(monthweek, factor(day, levels = c("Thu", "Wed", "Tue", "Mon"
  geom_tile(colour = "white") +
  facet_grid(year ~ factor(month, levels = c("Jan", "Feb","Mar", "Apr", "May", "Jun", "Jul", "Aug", "Se
  scale_fill_gradient(low="green", high="red") +
  labs(x="Week of Month",
       y="",
       title="DAI Variable Borrow Rates in 2021",
       fill="Rate (%)") +
  scale_colour_manual(values = NA)
```

```r
ggarrange(
  dai_stable_plot, dai_variable_plot
)
```





Since variable borrow rates can fluctuate greatly each day, it is difficult to classify the variable borrows as "bad" because it can recover the next day. Borrowing at higher stable rates has greater potential to get the user in trouble.

```
# put outliers back in data set, only care about percentiles
usdc_rates[is.na(usdc_rates)] <- 30
# summarize USDC stable borrow rate statistics
summary(usdc_rates$stableRate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.675  10.571  10.942  13.119  14.166  30.000
```

```
# put outliers back in data set, only care about percentiles
usdt_rates[is.na(usdt_rates)] <- 30
# summarize USDT stable borrow rate statistics
summary(usdt_rates$stableRate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.66   11.72   11.91   13.99   14.89   30.00
```

```
# put outliers back in data set, only care about percentiles
dai_rates[is.na(dai_rates)] <- 30
# summarize DAI stable borrow rate statistics
summary(dai_rates$stableRate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.585  11.846  11.995  15.206  16.182  47.936
```

I will define my statistic for a "good borrow day" as any day when the interest rate is between the min and first quartile. I will define a "bad borrow day" as any day when the interest rate is between the third quartile and the max.

```
# get dates where the stable rate is greater than the third quartile
usdc_b_borrow_days <- usdc_rates[usdc_rates$stableRate > 14.166,]$date
usdt_b_borrow_days <- usdt_rates[usdt_rates$stableRate > 14.89,]$date
dai_b_borrow_days <- dai_rates[dai_rates$stableRate > 16.182,]$date
```

```
# filter out stable borrows from the main transaction dataset
usdc_bad_borrows <- df %>%
  group_by(date) %>%
  filter(borrowRateMode == "Stable" & reserve == as.character("USDC"))
usdt_bad_borrows <- df %>%
  group_by(date) %>%
  filter(borrowRateMode == "Stable" & reserve == as.character("USDT"))
dai_bad_borrows <- df %>%
  group_by(date) %>%
  filter(borrowRateMode == "Stable" & reserve == as.character("DAI"))
```

```
# get borrows that occur on bad borrow days
usdc_bad_borrows <- usdc_bad_borrows[usdc_bad_borrows$date %in% usdc_b_borrow_days,]
usdt_bad_borrows <- usdt_bad_borrows[usdt_bad_borrows$date %in% usdt_b_borrow_days,]
dai_bad_borrows <- dai_bad_borrows[dai_bad_borrows$date %in% dai_b_borrow_days,]
```

```
# Percentage of bad USDC borrows in 2021
pct_usdc_b_borrows <- nrow(usdc_bad_borrows) /
nrow(df[df$borrowRateMode == "Stable" &
        df$reserve == as.character("USDC") &
        df$date >= "2021-01-01",])
pct_usdc_b_borrows
```

```
## [1] 0.1607717
```

```
# Percentage of bad USDT borrows in 2021
pct_usdt_b_borrows <- nrow(usdt_bad_borrows) /
nrow(df[df$borrowRateMode == "Stable" &
        df$reserve == as.character("USDT") &
        df$date >= "2021-01-01",])
pct_usdt_b_borrows
```

```
## [1] 0.2027577
```

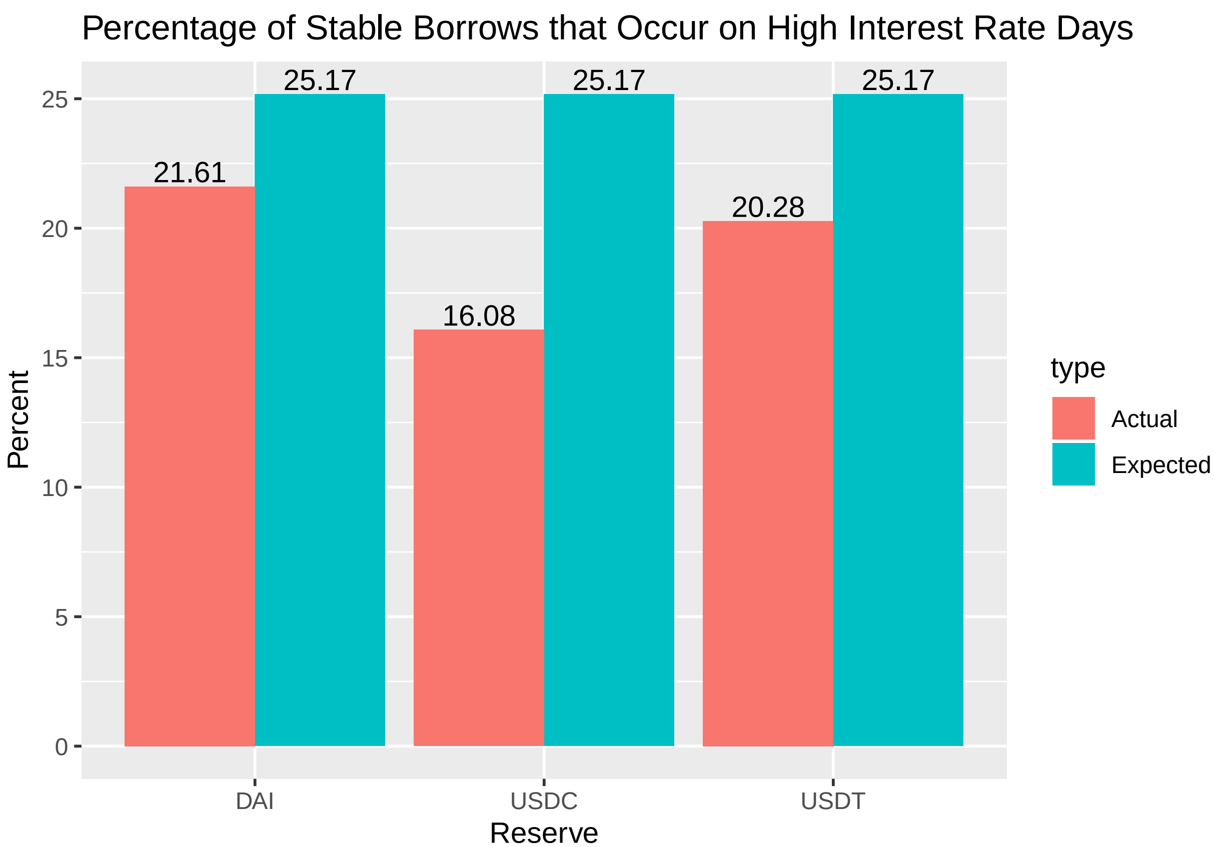
```
# Percentage of bad DAI borrows in 2021
pct_dai_b_borrows <- nrow(dai_bad_borrows) /
nrow(df[df$borrowRateMode == "Stable" &
        df$reserve == as.character("DAI") &
        df$date >= "2021-01-01",])
pct_dai_b_borrows
```

```
## [1] 0.2161401
```

```
# create dataframe to use in bar chart
bad_borrows_comparison_df <- data.frame(
  reserve = c("USDC", "USDT", "DAI", "USDC", "USDT", "DAI"),
  type = c("Actual", "Actual", "Actual", "Expected", "Expected", "Expected"),
  value = c(pct_usdc_b_borrows * 100,
            pct_usdt_b_borrows * 100,
            pct_dai_b_borrows * 100,
            length(usdc_b_borrow_days) / nrow(usdc_rates) * 100,
```

```r
             length(usdt_b_borrow_days) / nrow(usdt_rates) * 100,
             length(usdc_b_borrow_days) / nrow(usdc_rates) * 100)
)

# Plot as a bar chart
ggplot(bad_borrows_comparison_df, aes(factor(reserve), value, fill = type)) +
  geom_bar(stat="identity", position = "dodge") +
  geom_text(aes(label = round(value, digits = 2)), vjust = -0.2,
            position = position_dodge(0.9)) +
  xlab("Reserve") +
  ylab("Percent") +
  ggtitle("Percentage of Stable Borrows that Occur on High Interest Rate Days")
```



Since the third quartile to the max stable borrow rates is expected to account for 25% of users, this bar plot shows the actual percentage of borrows that occur on bad borrow days. We see lower percentages of borrows on high interest rate days which is exactly what we should expect. However, the difference isn't as great I would expect. Large numbers of borrows are happening at extraordinarily high stable rates.

```r
# of USDC bad borrowers, calculate bad borrow percentage and frequency of USDC stable borrows
usdc_bad_stable_borrowers <- usdc_bad_borrows %>%
  group_by(onBehalfOf_alias) %>%
  summarize(usdc = n())
usdc_stable_borrowers <- df %>%
  group_by(onBehalfOf_alias) %>%
  filter(borrowRateMode == "Stable" &
         reserve == as.character("USDC") &
         date >= "2021-01-01") %>%
```

```
  summarize(usdc_pct = n())
usdc_bad_stable_borrowers <- merge(usdc_bad_stable_borrowers, usdc_stable_borrowers)
usdc_bad_stable_borrowers$usdc_pct <- round(usdc_bad_stable_borrowers$usdc / usdc_bad_stable_borrowers$u
head(usdc_bad_stable_borrowers)
```

```
##   onBehalfOf_alias usdc usdc_pct
## 1       Aaron Diaz    1     0.33
## 2   Aaron Mitchell    2     0.06
## 3   Abdul Gabaldon    1     1.00
## 4     Adam Burwell    1     0.25
## 5       Adam Freed    1     0.50
## 6    Addie Donahue    1     0.50
```

```
# of USDT bad borrowers, calculate bad borrow percentage and frequency of USDT stable borrows
usdt_bad_stable_borrowers <- usdt_bad_borrows %>%
  group_by(onBehalfOf_alias) %>%
  summarize(usdt = n())
usdt_stable_borrowers <- df %>%
  group_by(onBehalfOf_alias) %>%
  filter(borrowRateMode == "Stable" &
         reserve == as.character("USDT") &
         date >= "2021-01-01") %>%
  summarize(usdt_pct = n())
usdt_bad_stable_borrowers <- merge(usdt_bad_stable_borrowers, usdt_stable_borrowers)
usdt_bad_stable_borrowers$usdt_pct <- round(usdt_bad_stable_borrowers$usdt / usdt_bad_stable_borrowers$u
head(usdt_bad_stable_borrowers)
```

```
##    onBehalfOf_alias usdt usdt_pct
## 1 Abigail Benedetti    1     0.11
## 2   Abigail Nichols    1     1.00
## 3       Alan Guzman    1     0.50
## 4     Alba Campbell    1     0.17
## 5          Alex Mui    1     0.50
## 6 Alexander Hopwood    1     1.00
```

```
# of DAI bad borrowers, calculate bad borrow percentage and frequency of DAI stable borrows
dai_bad_stable_borrowers <- dai_bad_borrows %>%
  group_by(onBehalfOf_alias) %>%
  summarize(dai = n())
dai_stable_borrowers <- df %>%
  group_by(onBehalfOf_alias) %>%
  filter(borrowRateMode == "Stable" &
         reserve == as.character("DAI") &
         date >= "2021-01-01") %>%
  summarize(dai_pct = n())
dai_bad_stable_borrowers <- merge(dai_bad_stable_borrowers, dai_stable_borrowers)
dai_bad_stable_borrowers$dai_pct <- round(dai_bad_stable_borrowers$dai / dai_bad_stable_borrowers$dai_pc
head(dai_bad_stable_borrowers)
```

```
##    onBehalfOf_alias dai dai_pct
## 1    Aaron Mitchell   1    0.33
## 2 Abigail Benedetti   2    0.33
## 3       Adam Howell   1    0.33
## 4     Adele Handley   2    0.67
## 5 Agripina Robinson   1    0.20
```

```
## 6     Alba Campbell   1    0.33
```

```r
# merge dataframes into one for all bad borrowers of Stable USDC, USDT, and DAI
bad_stable_borrowers <- merge(usdc_bad_stable_borrowers, usdt_bad_stable_borrowers, all = TRUE)
bad_stable_borrowers <- merge(bad_stable_borrowers, dai_bad_stable_borrowers, all = TRUE)
bad_stable_borrowers[is.na(bad_stable_borrowers)] <- 0
head(bad_stable_borrowers)
```

```
##    onBehalfOf_alias usdc usdc_pct usdt usdt_pct dai dai_pct
## 1        Aaron Diaz    1     0.33    0     0.00   0    0.00
## 2    Aaron Mitchell    2     0.06    0     0.00   1    0.33
## 3    Abdul Gabaldon    1     1.00    0     0.00   0    0.00
## 4 Abigail Benedetti    0     0.00    1     0.11   2    0.33
## 5   Abigail Nichols    0     0.00    1     1.00   0    0.00
## 6      Adam Burwell    1     0.25    0     0.00   0    0.00
```

```r
# interactive and filterable dataframe
reactable(bad_stable_borrowers)
```

onBehalfOf_	usdc	usdc_pct	usdt	usdt_pct
Aaron Diaz	1	0.33	0	0
Aaron Mitchell	2	0.06	0	0
Abdul Gabaldon	1	1	0	0
Abigail Benedetti	0	0	1	0.11
Abigail Nichols	0	0	1	1
Adam Burwell	1	0.25	0	0
Adam Freed	1	0.5	0	0
Adam Howell	0	0	0	0
Addie Donahue	1	0.5	0	0
Adele Handley	0	0	0	0

1–10 of 1242 rows Previous 1 2 3 4 5 ... 125 Ne

Kathy Lorenz, Herman Arno, Dorthy Thomas, Beatrice Rodriguez, etc. appear to be the most conistent bad borrowers.

```
# find bad borrowers who have made at least 10 bad borrows
worst_borrowers <- bad_stable_borrowers[bad_stable_borrowers$usdc >= 10 |
                                        bad_stable_borrowers$usdt >= 10 |
                                        bad_stable_borrowers$dai >= 10,]
unique(worst_borrowers$onBehalfOf_alias)
```

```
##   [1] Benjamin Richters Bernice Rodriguez Carl Briant        Christopher Cape
```

```
##  [5] Daniel Lynch      Darrell Kingston  Dorthy Thomas     Florence Johnson
##  [9] Frank Lennon      Frederick Stull   Herman Arno       Ida Murphy
## [13] Joel Moody        Joyce Dickson     Kathleen Kistner  Kathy Lorenz
## [17] Kay Reel          Lucio Hargis      Patricia Eatmon   Reba Funk
## [21] Roland Tufano     Sandra Freeman    Sharon West       Thomas Knight
## [25] Timothy Markert
## 50705 Levels: Aaron Adams Aaron Anderson Aaron Armendariz ... Zula Neyman
```

```
# of the 25 bad borrowers, find the ones who have liquidated
worst_borrowers_df <- df[df$onBehalfOf_alias %in% unique(worst_borrowers$onBehalfOf_alias),]
worst_liquidators_df <- df[df$user_alias %in% unique(worst_borrowers$onBehalfOf_alias),]
worst_liquidators_df <- worst_liquidators_df[worst_liquidators_df$type == "liquidation",]
bad_liquidator_users <- unique(worst_liquidators_df$user_alias)
bad_liquidator_users
```

```
##  [1] Benjamin Richters Frederick Stull   Patricia Eatmon   Timothy Markert
##  [5] Roland Tufano     Sharon West       Joyce Dickson     Thomas Knight
##  [9] Dorthy Thomas     Florence Johnson  Herman Arno       Ida Murphy
## [13] Sandra Freeman    Reba Funk         Kathleen Kistner
## 51421 Levels: Aaron Adams Aaron Anderson Aaron Armendariz ... Zula Neyman
```

Of the 25 frequent bad borrowers, 15 have already had at least one liquidation.

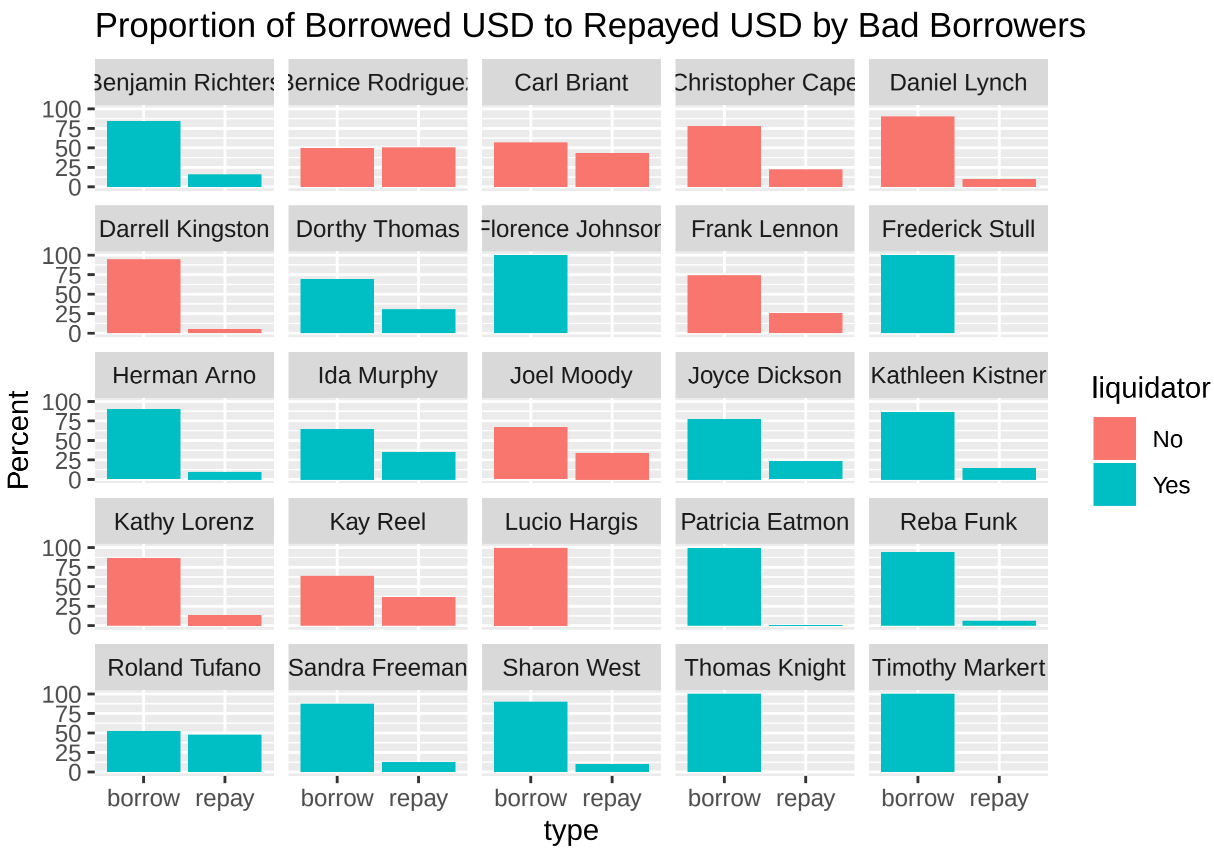
```
borrowers_br <- worst_borrowers_df %>%
  select("onBehalfOf_alias", "type", "amountUSD") %>%
  filter(type == "borrow" | type == "repay") %>%
  group_by(onBehalfOf_alias, type) %>%
  summarize_all(sum)
borrowers_br <- borrowers_br %>%
  group_by(onBehalfOf_alias) %>%
  mutate(amountUSD_prop = 100 * amountUSD / sum(amountUSD))
borrowers_br$liquidator <- ifelse(borrowers_br$onBehalfOf_alias %in% bad_liquidator_users, "Yes", "No")
borrowers_br
```

```
## # A tibble: 45 x 5
## # Groups:   onBehalfOf_alias [25]
##    onBehalfOf_alias  type   amountUSD amountUSD_prop liquidator
##    <fct>             <fct>      <dbl>          <dbl> <chr>
##  1 Benjamin Richters borrow   394936.           84.3 Yes
##  2 Benjamin Richters repay     73565.           15.7 Yes
##  3 Bernice Rodriguez borrow  3566220.           49.9 No
##  4 Bernice Rodriguez repay   3583655.           50.1 No
##  5 Carl Briant       borrow   875241.           56.6 No
##  6 Carl Briant       repay    670059.           43.4 No
##  7 Christopher Cape  borrow  2168169.           77.6 No
##  8 Christopher Cape  repay    624764.           22.4 No
##  9 Daniel Lynch      borrow   432463.           90.0 No
## 10 Daniel Lynch      repay     48065.           10.0 No
## # ... with 35 more rows
```

I want to compare the amount of USD borrowed versus the amount repayed and see the difference between users who have been liquidated and users who haven't.

```
# bar chart to compare the USD borrowed versus repayed
ggplot(borrowers_br) +
  geom_bar(mapping = aes(x = type, y = amountUSD_prop, fill = liquidator), stat = "identity") +
  facet_wrap(~ onBehalfOf_alias, ncol = 5) +
```

```r
  ggtitle("Proportion of Borrowed USD to Repayed USD by Bad Borrowers") +
  ylab("Percent")
```



It makes sense for the people who have borrowed a lot and repayed very little to have been liquidated in their past. For people like Lucio Hargis, I'm curious if they've only borrowed very recently.

```r
worst_borrowers_df %>%
  filter(type == "borrow") %>%
  group_by(onBehalfOf_alias) %>%
  filter(row_number() == n()) %>%
  select("onBehalfOf_alias", "date") %>%
  arrange(onBehalfOf_alias)
```

```
## # A tibble: 25 x 2
## # Groups:   onBehalfOf_alias [25]
##    onBehalfOf_alias  date
##    <fct>             <date>
##  1 Benjamin Richters 2021-02-26
##  2 Bernice Rodriguez 2021-05-19
##  3 Carl Briant       2021-10-15
##  4 Christopher Cape  2021-04-24
##  5 Daniel Lynch      2021-03-02
##  6 Darrell Kingston  2021-03-10
##  7 Dorthy Thomas     2021-05-16
##  8 Florence Johnson  2021-05-03
##  9 Frank Lennon      2021-10-14
## 10 Frederick Stull   2021-02-18
## # ... with 15 more rows
```

```r
# dates that Lucio Hargis borrowed
worst_borrowers_df[worst_borrowers_df$onBehalfOf_alias == "Lucio Hargis" &
                     worst_borrowers_df$type == "borrow",]$date
```

```
##  [1] "2021-07-30" "2021-07-31" "2021-08-04" "2021-08-04" "2021-08-05"
##  [6] "2021-08-06" "2021-08-07" "2021-08-10" "2021-08-11" "2021-08-16"
## [11] "2021-08-21" "2021-08-27" "2021-08-27" "2021-09-02" "2021-09-03"
## [16] "2021-09-06" "2021-09-06" "2021-09-08" "2021-09-11" "2021-09-13"
## [21] "2021-09-14" "2021-09-14" "2021-10-03" "2021-10-05" "2021-10-06"
## [26] "2021-10-06" "2021-10-06" "2021-10-07" "2021-10-12" "2021-10-14"
## [31] "2021-10-14" "2021-10-15" "2021-10-15" "2021-10-16"
```

As expected, Lucio has made a lot of borrows in the past couple months. I'm sure Lucio's health factor is worsening and could be on the verge of liquidation.

```r
# dates that Kathy Lorenz borrwed
kathy_dates <- worst_borrowers_df[worst_borrowers_df$onBehalfOf_alias == "Kathy Lorenz" &
                     worst_borrowers_df$type == "borrow",]$date
kathy_dates[1]
```

```
## [1] "2020-12-11"
```

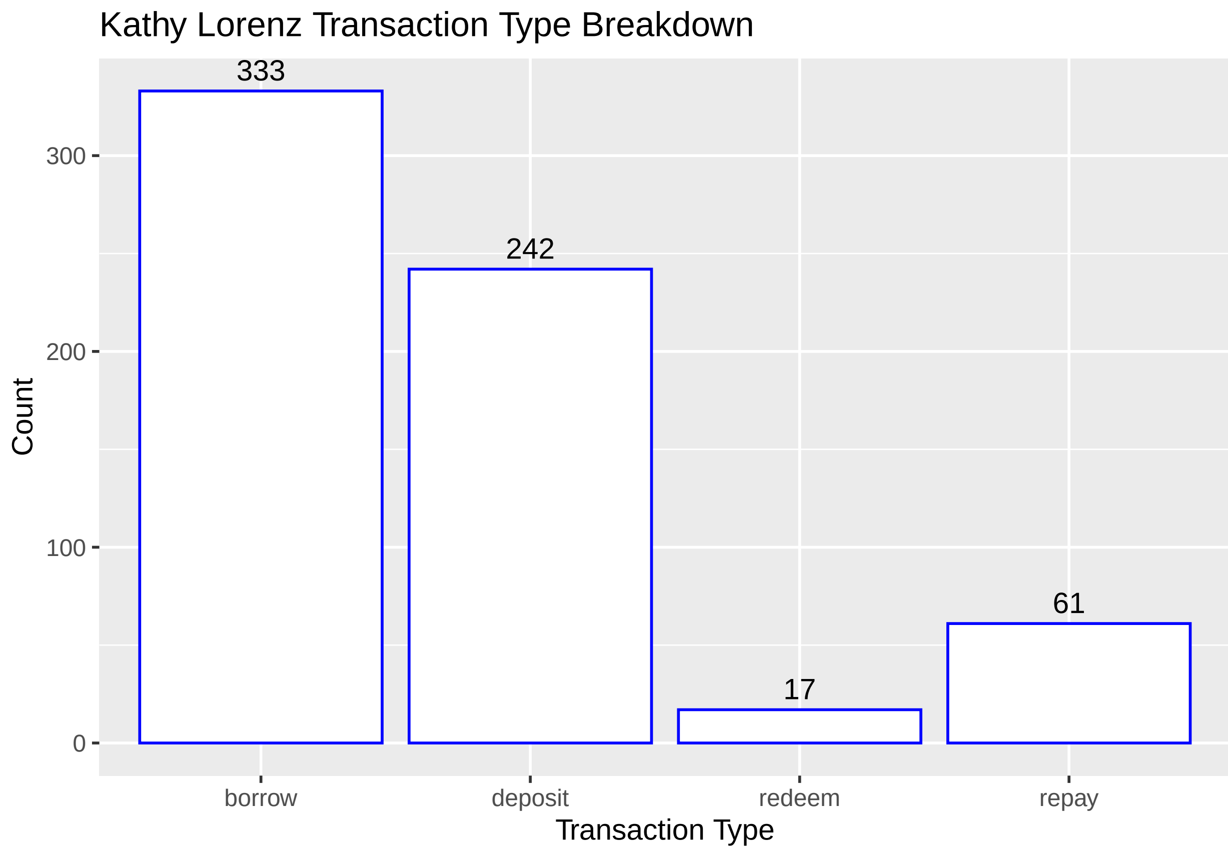
```r
kathy_dates[length(kathy_dates)]
```

```
## [1] "2021-09-07"
```

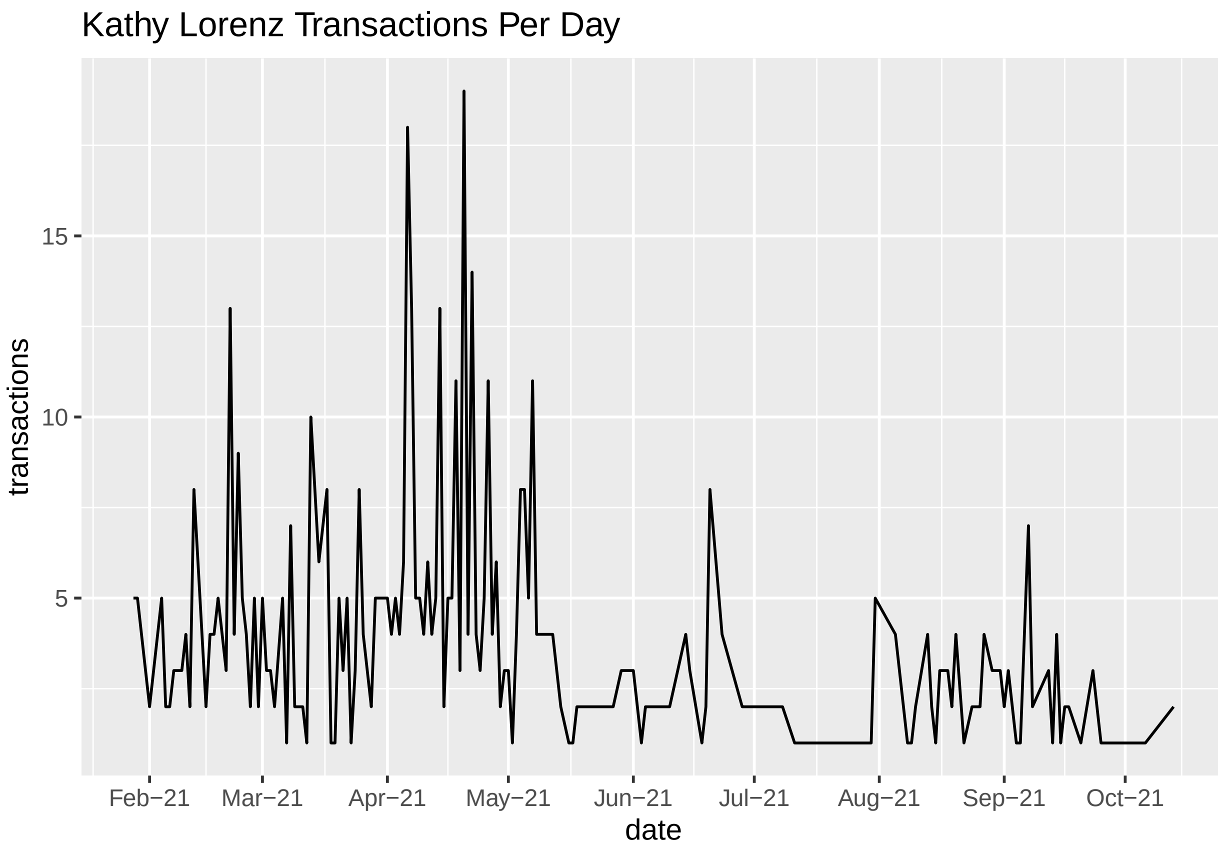
Kathy has made borrows from the very beginning and is quite active almost every day. Next notebook I'd like to understand how health factors are calculated. I'm surprised Kathy hasn't been liquidated.

```r
# transaction breakdown of Kathy Lorenz
kathy_df <- df[df$onBehalfOf_alias == "Kathy Lorenz",]
kathy_distribution <- kathy_df %>% count(type)

ggplot(data = kathy_distribution, aes(x = type, y = n)) +
  geom_bar(stat="identity", color = "blue", fill = "white") +
  xlab("Transaction Type") +
  ylab("Count") +
  ggtitle("Kathy Lorenz Transaction Type Breakdown") +
  geom_text(aes(label = n, vjust = -.5))
```




```r
# plot Kathy's transaction history
kathy_transactions <- kathy_df %>%
  filter(date >= "2021-01-01") %>%
  group_by(date) %>%
  summarize(transactions = n())
ggplot(kathy_transactions, aes(x = date, y = transactions)) +
    geom_line() +
    ggtitle("Kathy Lorenz Transactions Per Day") +
    scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")
```



Kathy appears to be a super user by making transactions almost every day. Looking back to one of my previous data frames, she's borrowed 13 million USD and repayed 2 million USD. While that seems like a lot, there are bad borrowers who have borrowed much more. I'm not sure how to characterize these users but is definitely something to look into.

Conclusion

Overall, much of this analysis would benefit from knowing the health factor of the user. Starting with predicting who will be a liquidator, I would've liked to try to even out the decision variables to minimize over-fitting. I think more thought has to go into the feature engineering. Next notebook I want to try to calculate health factor and use that as the main feature driving the model.

From my bad borrow analysis, I liked being able to breakdown a subset of users to try to understand their patterns and activities. In general, most people who have frequently made bad stable borrows have either been liquidated or are a new user in AAVE. I will continue to think more why some users like Kathy Lorenz have not been liquidated yet. I also want to look for good borrowers. I hypothesize that most of the yield farmers will be good borrowers. I also want to look at some of my teammate's survival plots more closely to help find bad variable borrowers. I need the survival plots because I'd like to know the average time most people take to pay back a loan so I can average out variable rates over that period of time.