The Wells Fargo Loyalty Program

Eliza Starr, Jason Boulware, Ashley Garrett

10/15/2017

Value and Implementation of the Loyalty Program

According to research, 81% of customers prefer financial institutions that offer some type of incentive for performing typical banking activities. One way that banks can cater to their customers' desires is through a loyalty or reward program where customers earn points that can be redeemed for products and/or services as well as discounts. In response to this research, we devised a Wells Fargo customer loyalty program that incentivizes customer loyalty and retention through a proposed badge and reward system.

Our loyalty algorithm ranks customers by factors that research shows are closely tied to customer retention: the number of checking and savings accounts, branch visits, and online transactions. For example, branch visits are taken into account because 61% of customers prefer full-service branches due to a preference for person-to-person interaction to gain assistance with major purchases, such as that of a home or car. Additionally, 49% of customers in general said that they are more likely to trust their banks when speaking to someone in person. Online transactions are significant because of the convenience offered to the 49% of Millennials claiming to be less interested in the physical branch bank format.

Customers can be given badges—silver, gold, and platinum, for example—to denote their ranking relative to other customers. Wells Fargo can then choose to reward these customers for their continued loyalty. Some forms of loyalty rewards already implemented by banks worldwide are as follows:

- Earning points based on monthly activities within the bank's network
- Free withdrawals from non-chain ATMS
- No-fee checking accounts
- Lower rates on certain types of loans
- Discounts on mortgages and HELOC

Furthermore, we are able to identify customers with loyalty potential by comparing their data with that of high-ranking customers. For example, research claims that younger generations tend to be less likely to switch financial institutions than older generations, and thereby have a higher retention rate. In support of this research, our algorithm shows a correlation between young-age and high customer loyalty as we have defined it. Wells Fargo can target these outlying customers with special offers that incentivize customers to increase their loyalty ranking.

Therefore, our loyalty system benefits both the customers and Wells Fargo; customers can receive loyalty rewards and offers from Wells Fargo, and Wells Fargo can better target their incentivizing efforts at outlying customers and their loyal customers.

Required Customer Inputs

Simplicity is not always the best way to approach a situation, but can be helpful when dealing with new concepts. In this case, Wells Fargo will not have to ask the customer to do more than they're already doing. All that is required for the loyalty algorithm to work is the continued collection of customer data that is already automized by Wells Fargo. The only further input customers are encouraged to give is to accept the rewards and offers offered to them by Wells Fargo.

Customer Interaction with Concept

Keeping the idea of simplicity strong, the customer should have a very positive, but small interaction with our idea. Customers may be able to see their rank online or in the mobile app. Using the app, the customer could receive hints on how to improve their ranking and see incentives based on reaching a higher level. If Wells Fargo decides to offer a deal based on the customer's rank, the customer will be notified.

Walking through the Algorithm

The loyalty algorithm first looks at one month and one variable at a time, ranking the customers against each other using checking_acct_ct, savings_acct_ct, online_bank_cnt, and branch_visit_cnt. For example, for the month of September, customer 1 could be ranked third in checking_acct_ct, but twelfth in savings_acct_ct. This number could fluctuate month-by-month based on the customer's increased or decreased interactions with the bank or with closing a checking account. Still one month at a time, we then averaged together the rankings of each variable for each customer. We then ranked the customers by this average to get each customer's overall ranking for each individual month.

Next, we averaged and ranked the monthly ranks to compute the overall loyalty ranking of each customer over the months of September through December. With this overall ranking, ranging from 1 to 50, we attempted to correlate Loyalty Rank with more variables which include: age, sav_balance_altered, check_bal_altered, and tenure_altered. Using the randomForest approach and various scatterplots, we discovered a slight correlation between a higher loyalty rank and age; younger customers scored high in loyalty based on our ranking system.

Documentation

Import Data

```
library(readxl)
month_end_balances <- read_excel("/usr/local/Learn2Mine-Main/galaxy-dist/lesson_datasets/Fake+Data+and+
    sheet = "Month end balances ", col_types = c("numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric",
        "numeric", "numeric", "numeric"))
```

Select month of December 2016

```
# create data frame and order it by masked_id and date
df <- data.frame(month_end_balances)
positions <- order(df$masked_id, df$asof_yyyymm)
df <- df[positions, ]

# separate months into different data frames
df_7 <- df[df$asof_yyyymm==201607,]
df_8 <- df[df$asof_yyyymm==201608,]
df_9 <- df[df$asof_yyyymm==201609,]
df_10 <- df[df$asof_yyyymm==201610,]
```

```r
df_11 <- df[df$asof_yyyymm==201611,]
df_12 <- df[df$asof_yyyymm==201612,]

#remove asof_yyyymm column
df_7$asof_yyyymm <- NULL
df_8$asof_yyyymm <- NULL
df_9$asof_yyyymm <- NULL
df_10$asof_yyyymm <- NULL
df_11$asof_yyyymm <- NULL
df_12$asof_yyyymm <- NULL

# Rank each variable for each customer and month
df_7$checking_acct_ct_rank <- rank(df_7$checking_acct_ct, ties.method= "average")
df_7$savings_acct_ct_rank <- rank(df_7$savings_acct_ct, ties.method= "average")
df_7$branch_visit_cnt_rank <- rank(df_7$branch_visit_cnt, ties.method= "average")
df_7$online_bank_cnt_rank <- rank(df_7$online_bank_cnt, ties.method= "average")

df_8$checking_acct_ct_rank <- rank(df_8$checking_acct_ct, ties.method= "average")
df_8$savings_acct_ct_rank <- rank(df_8$savings_acct_ct, ties.method= "average")
df_8$branch_visit_cnt_rank <- rank(df_8$branch_visit_cnt, ties.method= "average")
df_8$online_bank_cnt_rank <- rank(df_8$online_bank_cnt, ties.method= "average")

df_9$checking_acct_ct_rank <- rank(df_9$checking_acct_ct, ties.method= "average")
df_9$savings_acct_ct_rank <- rank(df_9$savings_acct_ct, ties.method= "average")
df_9$branch_visit_cnt_rank <- rank(df_9$branch_visit_cnt, ties.method= "average")
df_9$online_bank_cnt_rank <- rank(df_9$online_bank_cnt, ties.method= "average")

df_10$checking_acct_ct_rank <- rank(df_10$checking_acct_ct, ties.method= "average")
df_10$savings_acct_ct_rank <- rank(df_10$savings_acct_ct, ties.method= "average")
df_10$branch_visit_cnt_rank <- rank(df_10$branch_visit_cnt, ties.method= "average")
df_10$online_bank_cnt_rank <- rank(df_10$online_bank_cnt, ties.method= "average")

df_11$checking_acct_ct_rank <- rank(df_11$checking_acct_ct, ties.method= "average")
df_11$savings_acct_ct_rank <- rank(df_11$savings_acct_ct, ties.method= "average")
df_11$branch_visit_cnt_rank <- rank(df_11$branch_visit_cnt, ties.method= "average")
df_11$online_bank_cnt_rank <- rank(df_11$online_bank_cnt, ties.method= "average")

df_12$checking_acct_ct_rank <- rank(df_12$checking_acct_ct, ties.method= "average")
df_12$savings_acct_ct_rank <- rank(df_12$savings_acct_ct, ties.method= "average")
df_12$branch_visit_cnt_rank <- rank(df_12$branch_visit_cnt, ties.method= "average")
df_12$online_bank_cnt_rank <- rank(df_12$online_bank_cnt, ties.method= "average")

# Average the variable ranks of each customer for each month
df_7$loyalty_mean_7 <- rowMeans(df_7[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit_cn
df_8$loyalty_mean_8 <- rowMeans(df_8[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit_cn
df_9$loyalty_mean_9 <- rowMeans(df_9[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit_cn
df_10$loyalty_mean_10 <- rowMeans(df_10[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit
df_11$loyalty_mean_11 <- rowMeans(df_11[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit
df_12$loyalty_mean_12 <- rowMeans(df_12[,c("checking_acct_ct_rank","savings_acct_ct_rank","branch_visit

# Rank the customer's loyalty for each month
df_7$loyalty_rank_7 <- rank(df_7$loyalty_mean_7, ties.method="average")
df_8$loyalty_rank_8 <- rank(df_8$loyalty_mean_8, ties.method="average")
```

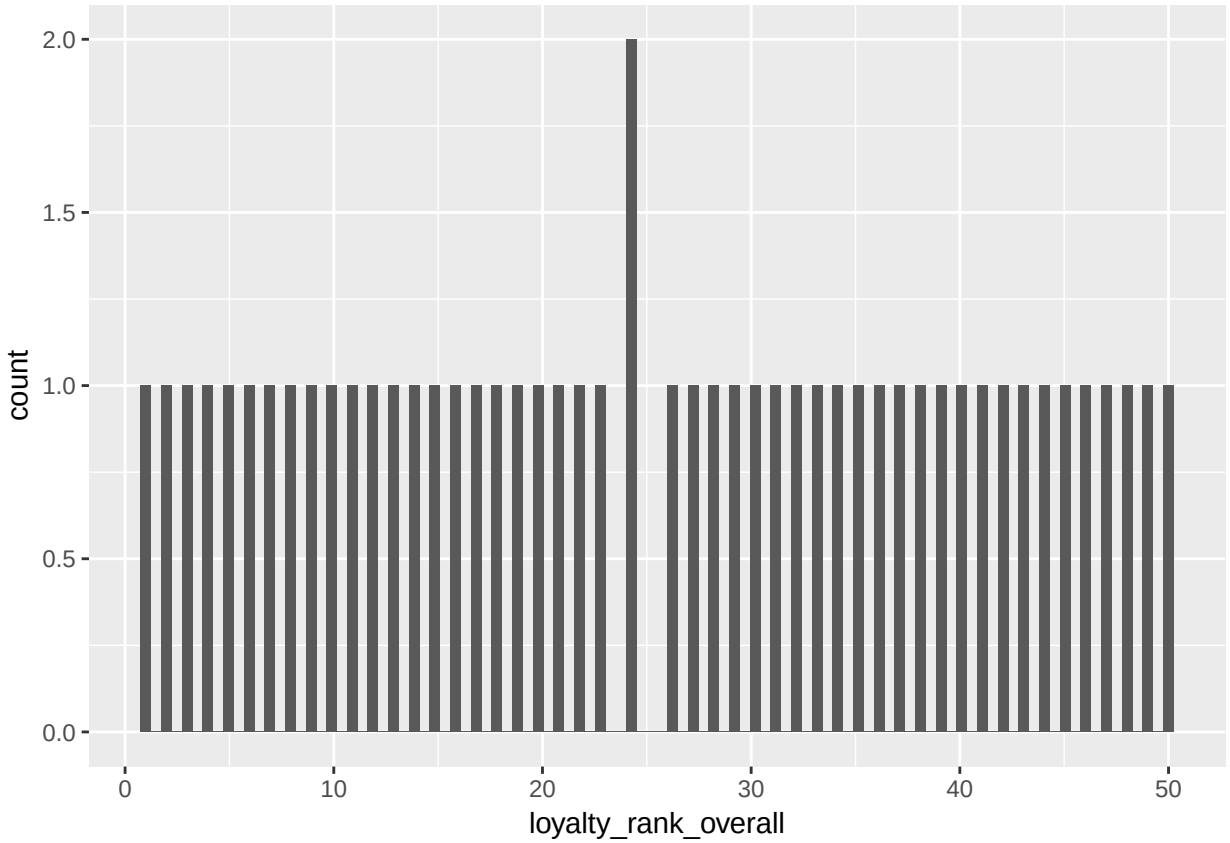
```r
df_9$loyalty_rank_9 <- rank(df_9$loyalty_mean_9, ties.method="average")
df_10$loyalty_rank_10 <- rank(df_10$loyalty_mean_10, ties.method="average")
df_11$loyalty_rank_11 <- rank(df_11$loyalty_mean_11, ties.method="average")
df_12$loyalty_rank_12 <- rank(df_12$loyalty_mean_12, ties.method="average")

# Join dataframes for each month
df_rank <- data.frame(df_7$masked_id, df_7$age, df_7$tenure_altered, df_7$check_bal_altered, df_7$sav_ba

# Rename columns
colnames(df_rank) <- c("id", "age","tenure_altered","check_bal_altered","sav_bal_altered","rank_7", "ra

# Average loyalty_means for each customer from each month
df_rank$loyalty_mean_overall <- rowMeans(df_rank[,c("rank_7", "rank_8", "rank_9", "rank_10", "rank_11",
# Find loyalty ranking for each customer from all of the months (from months 7 through 12)
df_rank$loyalty_rank_overall <- rank(df_rank$loyalty_mean_overall, ties.method="average")

# Want an even distribution and a rank range from 1 to 50
library(ggplot2)
ggplot(df_rank, aes(loyalty_rank_overall)) + geom_histogram(bins=100)
```

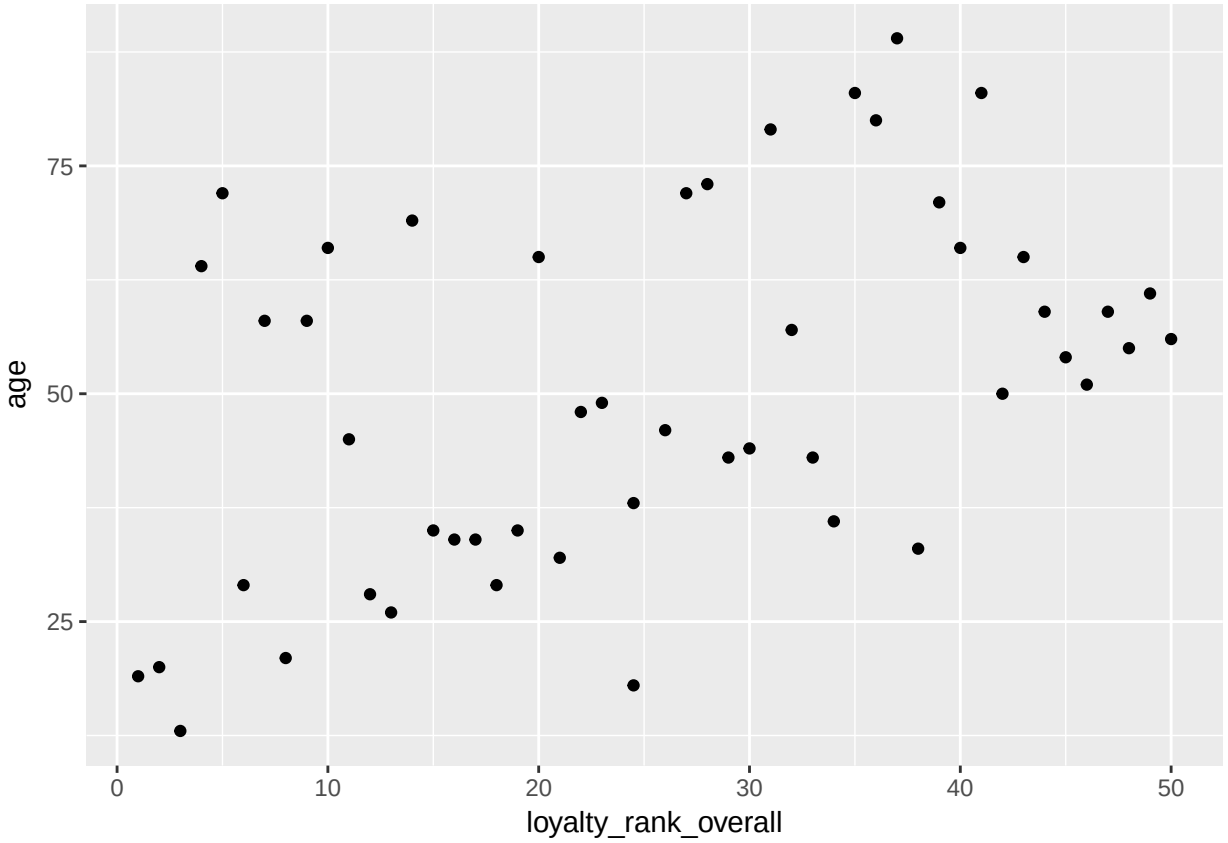


Correlation Scatterplots

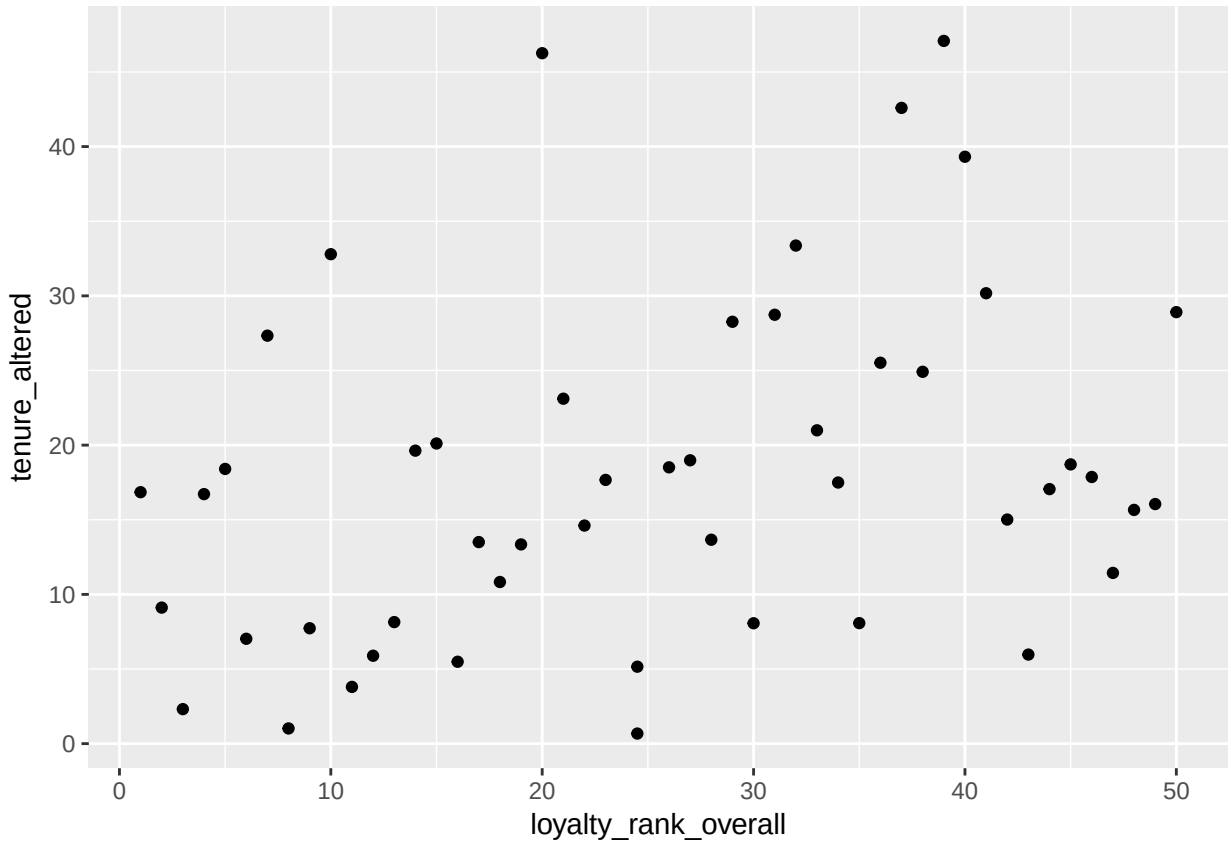
These scatterplots are intended to show correlations between a customer's rank and other variables, if a correlation exists.

Visit https://ashleyg.shinyapps.io/loyalty_app/ to view the following scatterplots interactively.

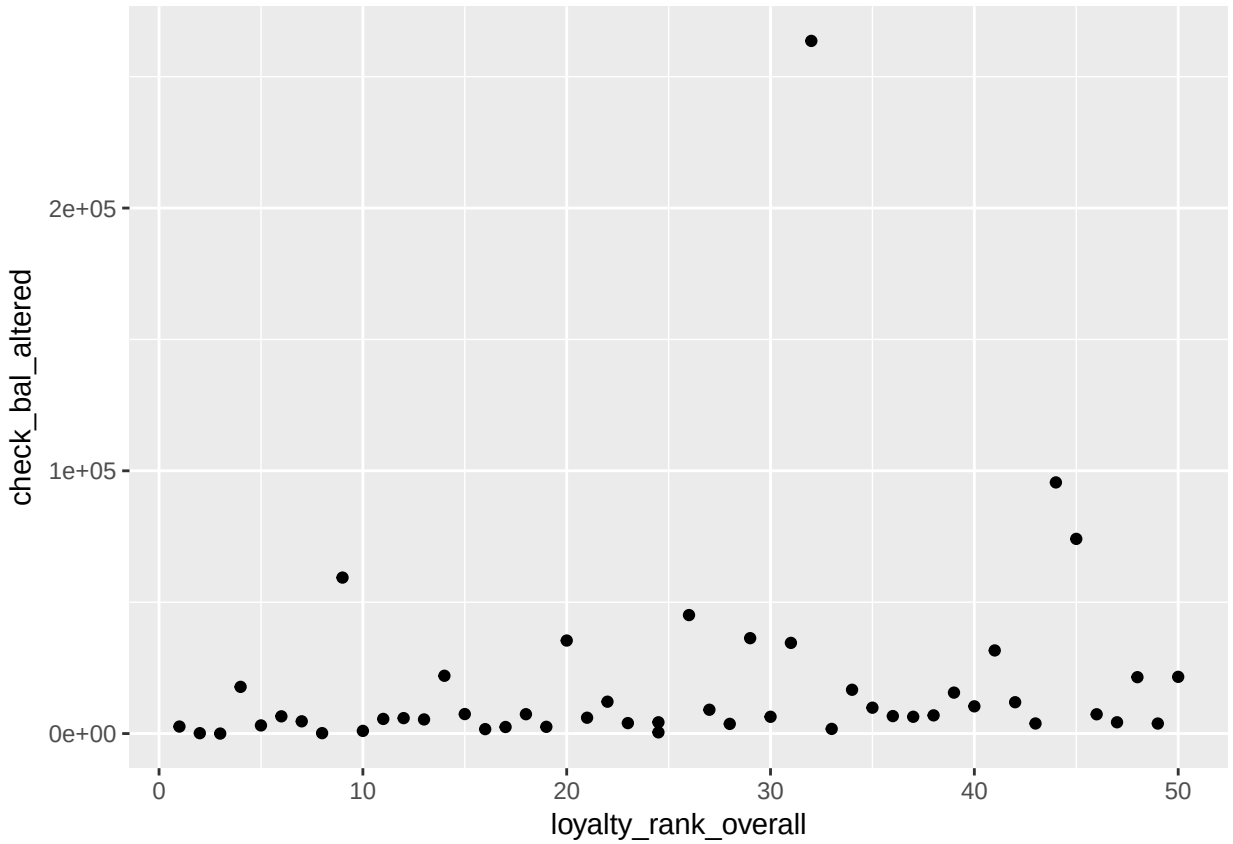
```
library(ggplot2)
ggplot(df_rank, aes(x=loyalty_rank_overall, y=age)) + geom_point()
```



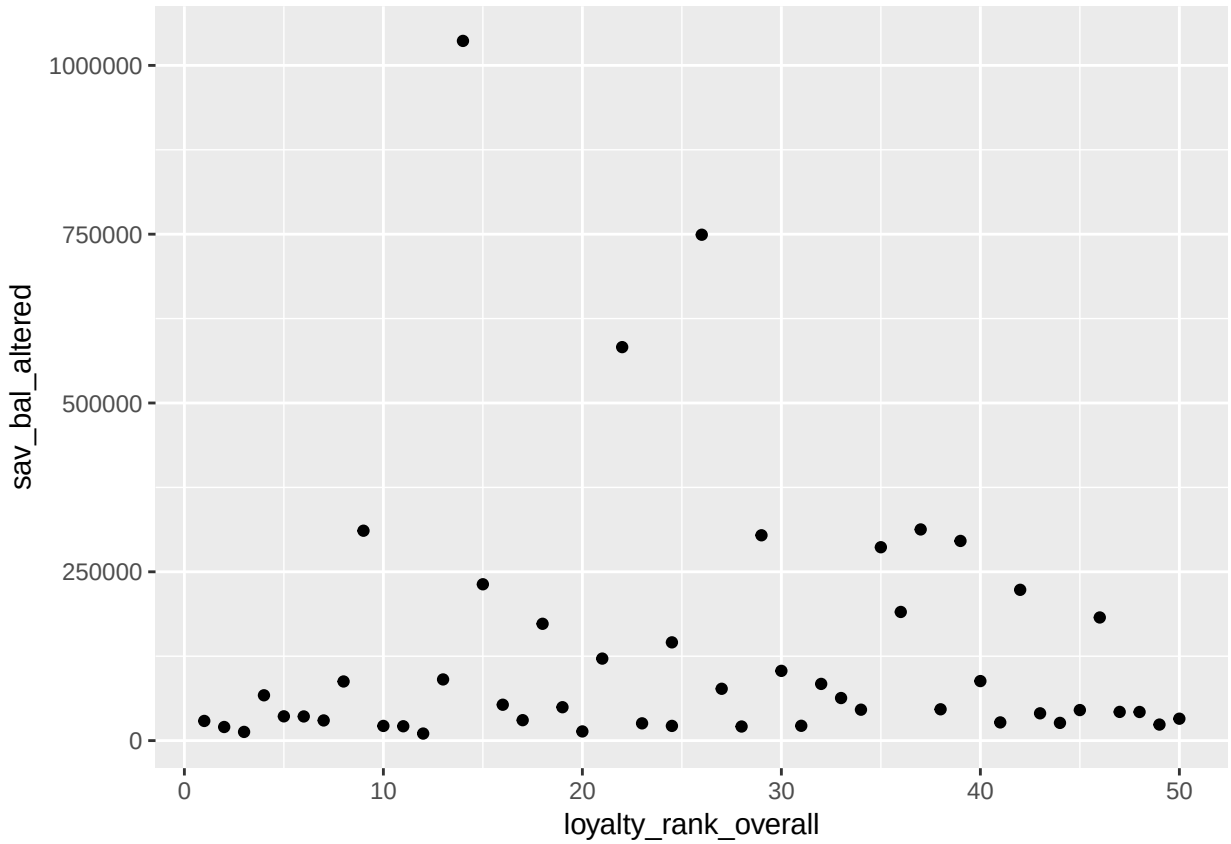

```
ggplot(df_rank, aes(x=loyalty_rank_overall, y=tenure_altered)) + geom_point()
```




```
ggplot(df_rank, aes(x=loyalty_rank_overall, y=check_bal_altered)) + geom_point()
```




```
ggplot(df_rank, aes(x=loyalty_rank_overall, y=sav_bal_altered)) + geom_point()
```



Prediction

Error rate of 38%, % variables explained of 18.75

```
library(randomForest)
```

```
## randomForest 4.6-10
```

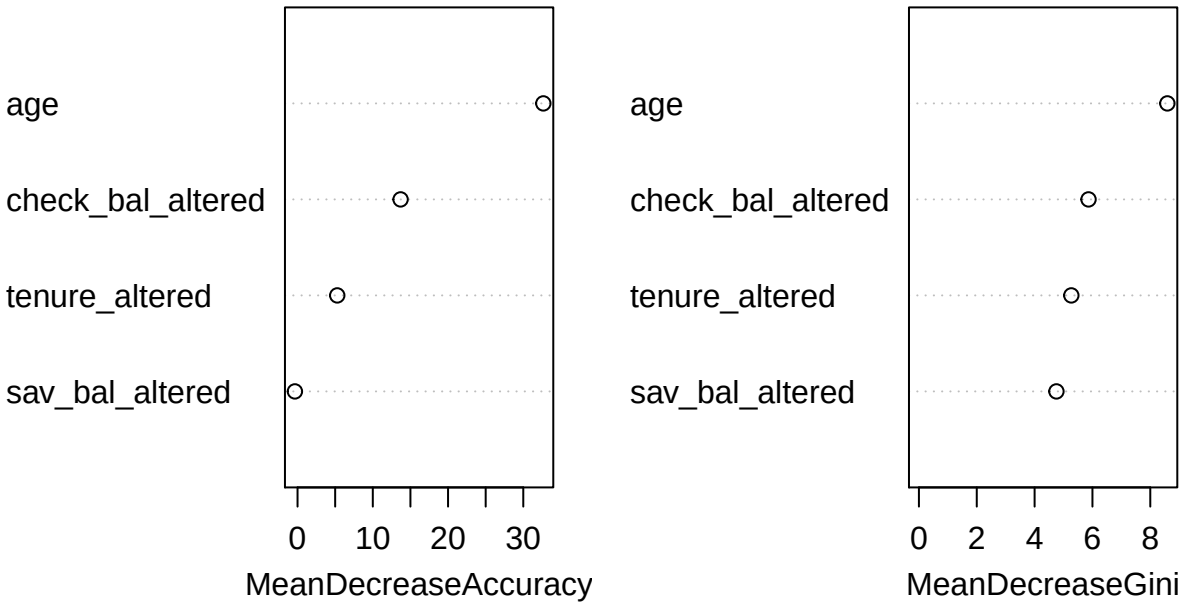
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
df_rank$loyal_top = as.factor(df_rank$loyalty_rank_overall<=25)
fit <- randomForest(loyal_top ~ age + tenure_altered + check_bal_altered + sav_bal_altered,
                    data=df_rank,
                    importance=TRUE,
                    ntree=2000)

varImpPlot(fit)
```

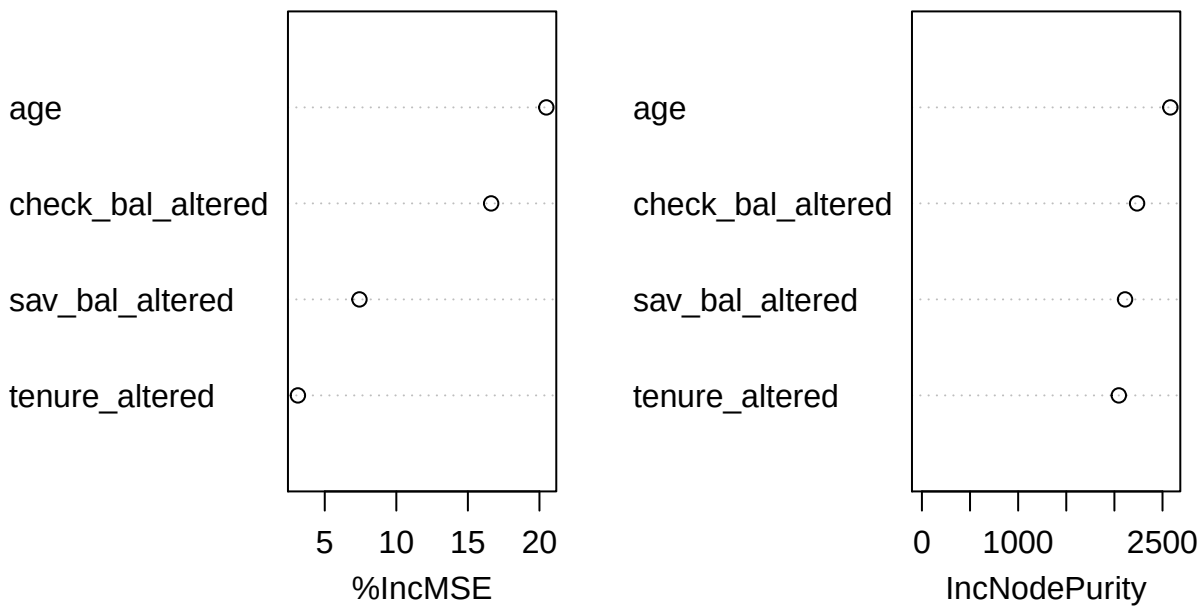



```
print(fit)
```

```
##
## Call:
##  randomForest(formula = loyal_top ~ age + tenure_altered + check_bal_altered +      sav_bal_altered,
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 38%
## Confusion matrix:
##       FALSE TRUE class.error
## FALSE    17    8        0.32
## TRUE     11   14        0.44
```

```
df_rank$loyal_top = df_rank$loyalty_rank_overall
fit <- randomForest(loyal_top ~ age + tenure_altered + check_bal_altered + sav_bal_altered,
                    data=df_rank,
                    importance=TRUE,
                    ntree=2000)

varImpPlot(fit)
```




```
print(fit)
```

```
##
## Call:
##  randomForest(formula = loyal_top ~ age + tenure_altered + check_bal_altered +      sav_bal_altered,
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 170.2561
##                     % Var explained: 18.24
```