Capstone

Agatha North

December 3, 2017

Employee Attrition

An Under Recognized Problem

Workplace attrition is a universal factor in any form of employment. Attrition has many forms, both voluntary and involuntary, and its effects on a business can be felt both fiscally and within the workplace community. Because even valued employees are affected by attrition it is often viewed as a necessary evil in the workplace, something that is outside individual control.

Employee morale tends to be the biggest loss when attrition is concerned. After all, no one likes being fired, no one enjoys seeing their friends and co-workers being let go, and no one wants to train their friends' replacement. At the same time, businesses don't want their best workers leaving to seek opportunity elsewhere. Skilled workers can be considered a commodity like any other. The investment that goes into them; the hours of training, the workplace efficiency that familiarity helps foster, and the saved expenses on training, interviewing, and advertising, are all significant factors that companies need to consider. New hires mean risk where reliable and experienced employees can be considered stable.

Preventing unnecessary attrition in the workplace is an effective means of reducing expenses, and increasing employee engagement. Finding sustainable and realistic ways to increase employee satisfaction is something most companies can benefit and learn from.

The Data

This analysis is possible due to 37 anonymous companies that used the company feedback app Happyforce (found at https://www.myhappyforce.com/en/). These companies provided their data for this set, which was then broken down into 4 dataframes. This includes churn data, employee comments, employee likes/dislikes, and self reported happiness votes. The dataset can be found at: https://www.kaggle.com/harriken/employeeturnover

Data Wrangling

Cleaning this data was relatively simple. The data is straightforward in its purpose and format, so the chief concerns in cleaning it were to make the dataframes with the same variables match, to increase term clarity, and to simplify the dates.

My code:

Cleaning the dates using a combination of the gsub function and lubridate.

```
levels(votes$companyAlias) <- c("C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8",
    "C9", "C10", "C11", "C12", "C13", "C14", "C15", "C16", "C17", "C18", "C19",
    "C20", "C21", "C22", "C23", "C24", "C25", "C26", "C27", "C28", "C29", "C30",
    "C31", "C32", "C33", "C34", "C35", "C36", "C37")
levels(churn$companyAlias) <- c("C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8",
    "C9", "C10", "C11", "C12", "C13", "C14", "C15", "C16", "C17", "C18", "C19",
    "C20", "C21", "C22", "C23", "C24", "C25", "C26", "C27", "C28", "C29", "C30",
    "C31", "C32", "C33", "C34", "C35", "C36", "C37")
```

```
levels(comments$companyAlias) <- c("C1", "C2", "C3", "C4", "C5", "C6", "C7",
    "C8", "C9", "C10", "C11", "C12", "C13", "C14", "C15", "C16", "C17", "C18",
    "C19", "C20", "C21", "C22", "C23", "C24", "C25", "C26", "C27", "C28", "C29",
    "C30", "C31", "C32", "C33", "C34", "C35", "C36")
levels(interaction$companyAlias) <- c("C1", "C2", "C3", "C4", "C5", "C6", "C7",
    "C8", "C9", "C10", "C11", "C12", "C13", "C14", "C15", "C16", "C17", "C18",
    "C19", "C20", "C21", "C22", "C23", "C24", "C25", "C26", "C27", "C28", "C29",
    "C30", "C31", "C32", "C33", "C34", "C35")

colnames(churn)[4] <- c("lastPostDate")
```

Cleaning up the anonymous companies into clearly labeled aliases prior to deciding which companies will be removed for the machine learning process. Also shortening and simplifying the column name within churn.

```
table(votes$companyAlias)
```

```
##
##    C1    C2    C3    C4    C5    C6    C7    C8    C9   C10   C11   C12
##  6286  4680    46   442 50479   351 23023    28  3602   153    42 26756
##   C13   C14   C15   C16   C17   C18   C19   C20   C21   C22   C23   C24
##  5347  3827     8   764 24902  6427 16886   722  5847  6450  3322  2437
##   C25   C26   C27   C28   C29   C30   C31   C32   C33   C34   C35   C36
##  5497   191 11734    53    97  1497  1200  4156  3648    92   162    15
##   C37
##    63
```

```
votes <- votes %>% group_by(companyAlias) %>% filter(n() >= 100)
```

Briefly checking the votes data to determine which companies may be worth removing. I used votes as the basis for this, rather than comments and interactions, because voting was the baseline function of the App and therefore most relevant on the overall data. After inspecting the data, and consulting plots, it seemed best to remove any companies with 100 instances of voting or less. These companies were clearly either too small, or their engagement was minimal, and would skew the results.

```
churn <- churn[churn$companyAlias != "C3", ]
churn <- churn[churn$companyAlias != "C8", ]
churn <- churn[churn$companyAlias != "C11", ]
churn <- churn[churn$companyAlias != "C15", ]
churn <- churn[churn$companyAlias != "C28", ]
churn <- churn[churn$companyAlias != "C29", ]
churn <- churn[churn$companyAlias != "C34", ]
churn <- churn[churn$companyAlias != "C36", ]
churn <- churn[churn$companyAlias != "C37", ]
churn <- droplevels(churn)

comments <- comments[comments$companyAlias != "C3", ]
comments <- comments[comments$companyAlias != "C8", ]
comments <- comments[comments$companyAlias != "C11", ]
comments <- comments[comments$companyAlias != "C15", ]
comments <- comments[comments$companyAlias != "C28", ]
comments <- comments[comments$companyAlias != "C29", ]
comments <- comments[comments$companyAlias != "C34", ]
comments <- comments[comments$companyAlias != "C36", ]
comments <- comments[comments$companyAlias != "C37", ]
comments <- droplevels(comments)
```

```
interaction <- interaction[interaction$companyAlias != "C3", ]
interaction <- interaction[interaction$companyAlias != "C8", ]
interaction <- interaction[interaction$companyAlias != "C11", ]
interaction <- interaction[interaction$companyAlias != "C15", ]
interaction <- interaction[interaction$companyAlias != "C28", ]
interaction <- interaction[interaction$companyAlias != "C29", ]
interaction <- interaction[interaction$companyAlias != "C34", ]
interaction <- interaction[interaction$companyAlias != "C36", ]
interaction <- interaction[interaction$companyAlias != "C37", ]
interaction <- droplevels(interaction)
```

Moving on to the other data, I used the results of the filter on votes to determine which companies needed to be removed from the rest of the sets.

```
votes$EmpID <- paste0(votes$employee, votes$companyAlias)
comments$EmpID <- paste0(comments$employee, comments$companyAlias)
interaction$EmpID <- paste0(interaction$employee, interaction$companyAlias)
churn$EmpID <- paste0(churn$employee, churn$companyAlias)

comments$txt <- as.character(comments$txt)
comments$LengthTxt <- nchar(comments$txt)

EmpVoteCount <- votes %>% group_by(EmpID) %>% dplyr::summarize(VoteCount = n(),
    MeanVote = mean(vote))

EmpComments <- comments %>% group_by(EmpID) %>% dplyr::summarize(AvgLength = mean(LengthTxt),
    AvgLikes = mean(likes), AvgDislikes = mean(dislikes))

EmpCombined <- left_join(EmpVoteCount, EmpComments)
```

```
## Joining, by = "EmpID"
```

```
EmpCombined[is.na(EmpCombined)] <- 0

churn <- churn %>% dplyr::select(EmpID, stillExists, companyAlias)
EmpCombined <- left_join(EmpCombined, churn)
```

```
## Joining, by = "EmpID"
```

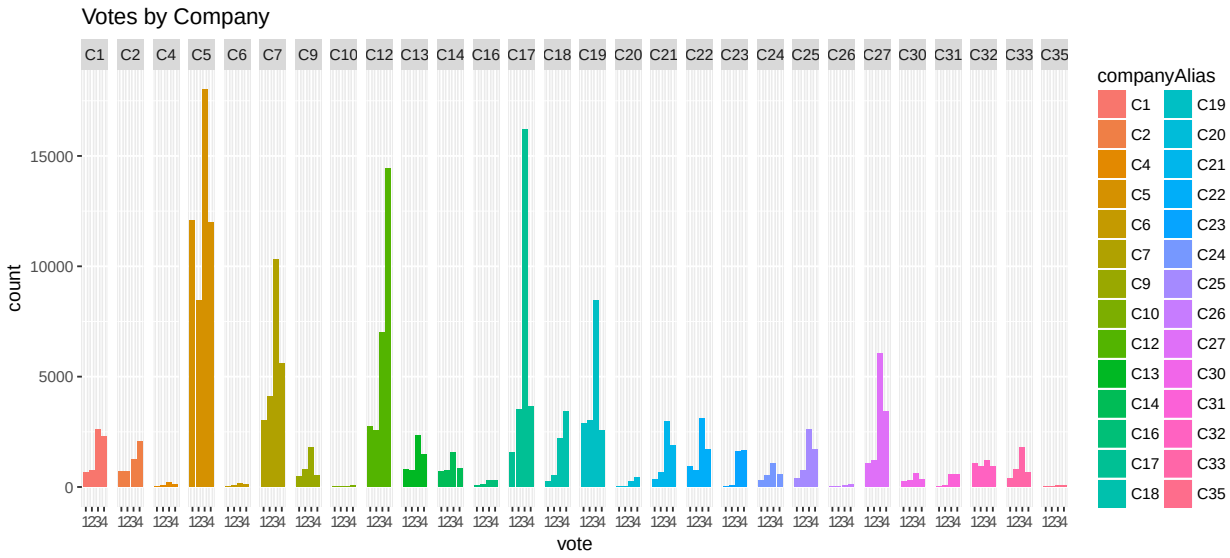
To get the code into the state it needed to be in for analysis and machine learning I created a new frame with individual employees and their personal stats. Their usage of the App was far more apparent through these combined frames than it would be otherwise.

To deal with instances of NA, it seemed best to replace all of those missing values with 0, indicating disengagement with that particular potion of the App.

Exploration of the Data

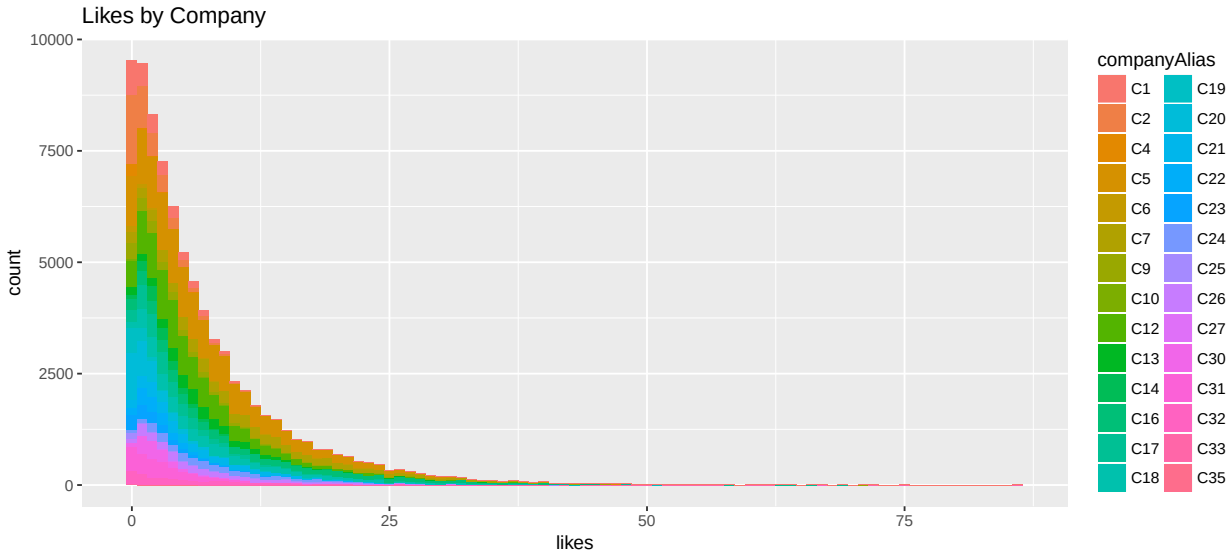
When digging into the data certain things quickly became apparent. First and foremost was that the size of the companies involved in the data varied, and as a result, the volume of their votes, likes, dislikes, comments, and other data of relevance.

```
ggplot(votes, aes(vote, fill = companyAlias)) + geom_bar() + facet_grid(~votes$companyAlias) +
    labs(title = "Votes by Company")
```

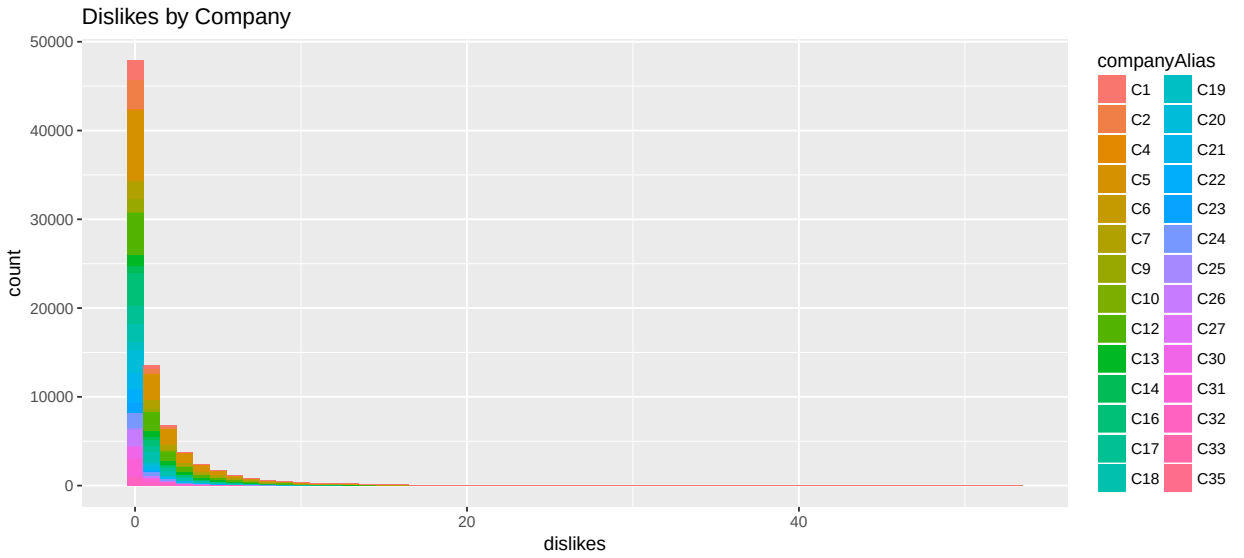


Some things to note from the plot above, companies 2, 12, and 18 are particularly happy, with the strong majority of their votes marked as 4. Companies 1, 23, and 31 also stand out in terms of overall happiness compared to unhappiness. On the otherhand company 33 stands out as the most evenly distributed between negative and positve votes, which incidentally also makes it the least happy of the batch. Looking at this data, I resolved to include companyAlias as a variable later in the machine learning. It was clear that the data lacked an important element, the company's individual firing policy, so retaining the actual alias as a variable would help make up for that.

```
ggplot(comments, aes(likes, fill = companyAlias)) + geom_histogram(binwidth = 1) +
    labs(title = "Likes by Company")
```

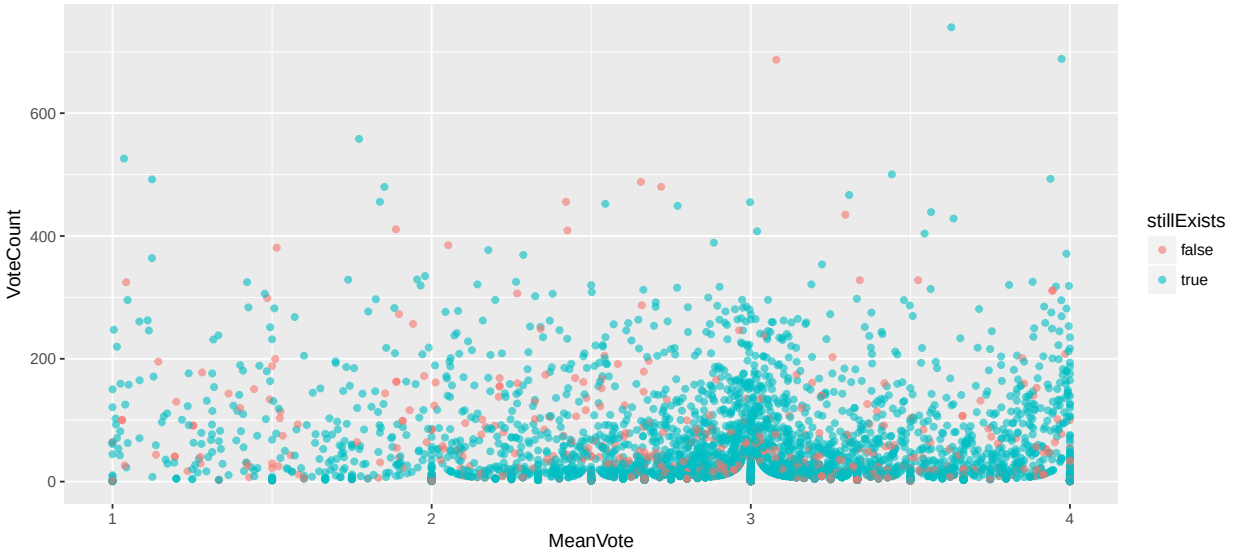



```
ggplot(comments, aes(dislikes, fill = companyAlias)) + geom_histogram(binwidth = 1) +
    labs(title = "Dislikes by Company")
```



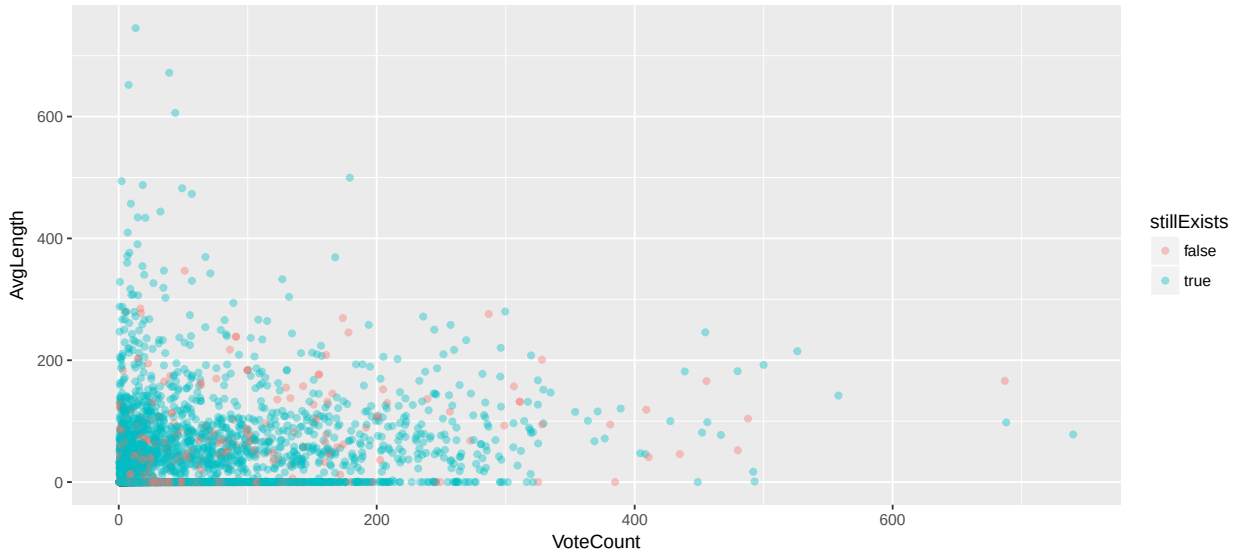
It was far more common for comments to receive likes than dislikes in every company. These slopes were overall unsurpising.

```
ggplot(EmpCombined, aes(EmpID, col = stillExists, x = MeanVote, y = VoteCount)) +
    geom_point(position = "jitter", alpha = 0.6)
```

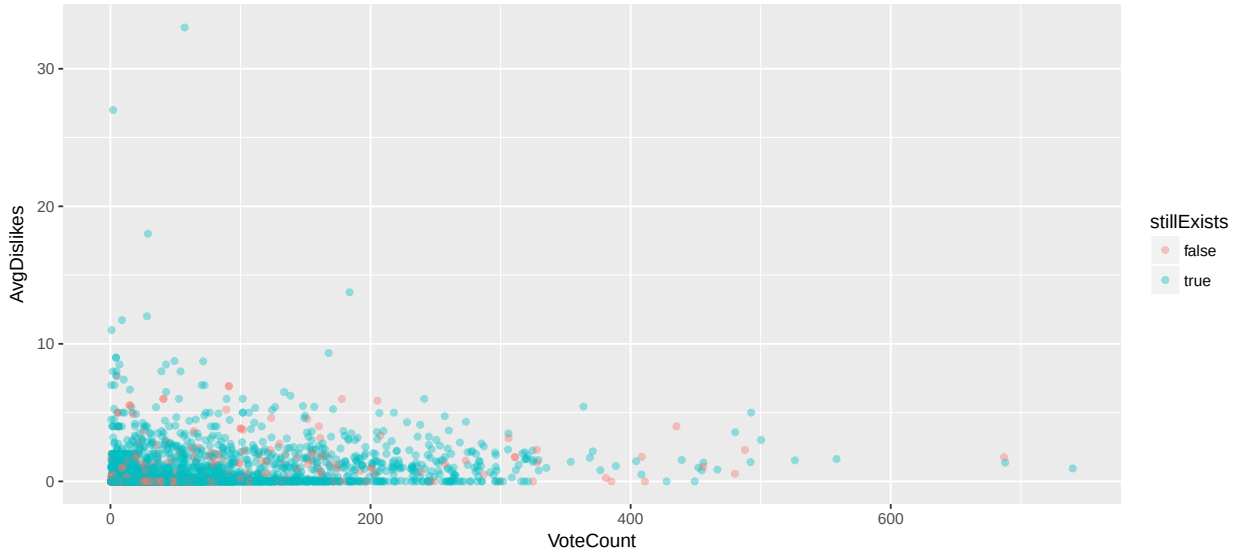


Investigating the individual votes when compared to how often an employee voted was quite fruitful. The distribution of votes for those who churned was extremely even across the mean vote, but the density of all votes trended towards the higher end. However, I did not expect to see individuals who's mean vote was roughly 1 have anywhere near the same vote count as individuals with a higher level of day to day happiness. Clearly day to day happiness wasn't all that strong of an indicator on its own, which proved useful later. It's facinating to think that there were some individuals who publically voted their unhappiness for over half a year and still remained at their company.

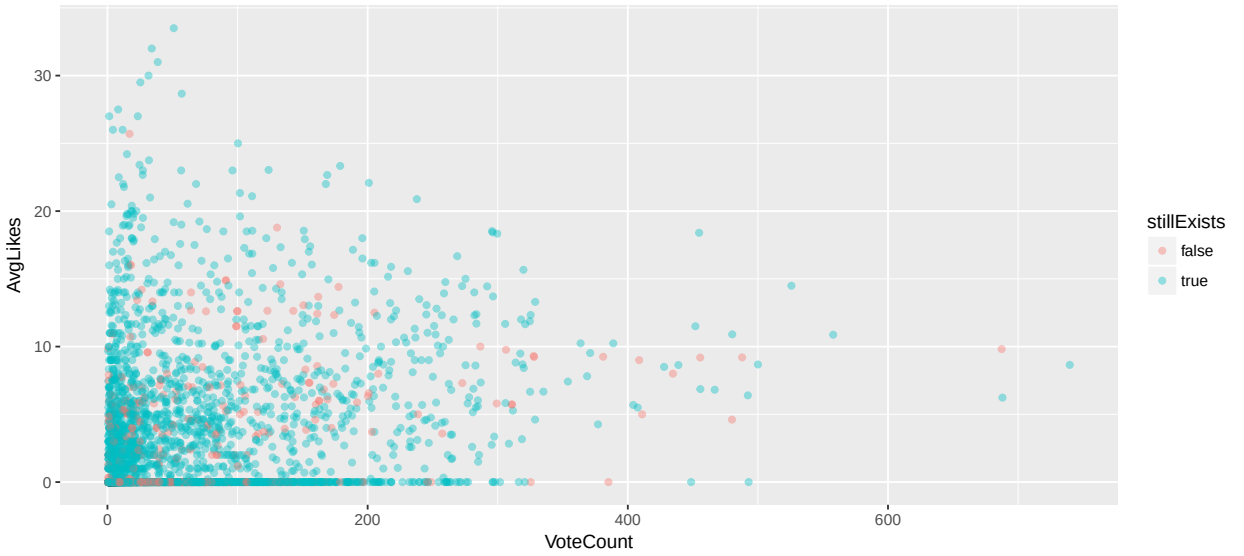
```
ggplot(EmpCombined, aes(EmpID, col = stillExists, x = VoteCount, y = AvgLength)) +
    geom_point(position = "jitter", alpha = 0.4)
```




```
ggplot(EmpCombined, aes(EmpID, col = stillExists, x = VoteCount, y = AvgDislikes)) +
    geom_point(position = "jitter", alpha = 0.4)
```




```
ggplot(EmpCombined, aes(EmpID, col = stillExists, x = VoteCount, y = AvgLikes)) +
    geom_point(position = "jitter", alpha = 0.4)
```



Looking at the average length, average dislikes, and average likes of individuals compared to their vote count, it's immediately clear that these three elements of engagement were strong indicators of investment in the app, if not the company as well. Interestingly there were few employees that seemed to be engaged by both the votes and also the comments.

Random Forest

```
##
## Call:
##  randomForest(formula = stillExists ~ ., data = EmpCom_train,      ntree = 500, mtry = 2, importance
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 10.98%
## Confusion matrix:
##       false true class.error
## false   294  237  0.44632768
## true     97 2413  0.03864542
```

This is the base model with nothing done to augment the results. It's only a bit better than a 50 50 split, so far from desirable. To proceed, the goal was to get the predictive power closer to 85% for churned employees. To achieve this I decided to use both ROSE, to help accomodate for the unbalanced nature of the data, and stacking with the caret, and caretEnsemble packages.

```
data_rose <- ROSE(stillExists ~ ., data = EmpCom_train, seed = 237)$data
table(data_rose$stillExists)
```

```
##
##  true false
##  1505  1536
```

```
modelrose <- randomForest(stillExists ~ ., data = data_rose, ntree = 1000, mtry = 2,
    importance = TRUE, na.action = na.roughfix, replace = FALSE)
modelrose
```

```
##
```

```
## Call:
##  randomForest(formula = stillExists ~ ., data = data_rose, ntree = 1000,      mtry = 2, importance =
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 20.59%
## Confusion matrix:
##       true false class.error
## true  1198   307   0.2039867
## false  319  1217   0.2076823
```

Running the data through the ROSE package was already a marked improvement. The synthetic additions to the under represented class in stillExists closed the gap towards my intended 85% or higher goal as shown here. Stacking, the process of creating additional predictor variables using the results of a variety of machine learning methods, would now be applied to the entire data set before making any split.

```
data_rosestack <- ROSE(stillExists ~ ., data = EmpCombMod, seed = 237)$data
table(data_rosestack$stillExists)
```

```
##
##  true false
##  2121  2137
```

```
data_rosestack$stillExists <- ifelse(data_rosestack$stillExists == "true", 1,
    0)

splitStack <- sample.split(data_rosestack, SplitRatio = 0.67)
StackS <- data_rosestack[splitStack, ]
testingData <- data_rosestack[!splitStack, ]

split2 <- sample.split(StackS, SplitRatio = 0.5)
ensembleData <- StackS[split2, ]
blenderData <- StackS[!split2, ]
```

After using ROSE on the entire set, I convert stillExists from true/false to a binary 1 or 0. Then I split the set roughly into thirds. This will allow me to run multiple methods on different parts of the data and still have a sizable portion left over for testing later.

```
labelName <- "stillExists"
predictors <- names(ensembleData)[names(ensembleData) != labelName]

myControl <- trainControl(method = "cv", number = 5, returnResamp = "none")

testSplitModel <- train(blenderData[, predictors], blenderData[, labelName],
    method = "gbm", trControl = myControl)

model_gbm <- train(ensembleData[, predictors], ensembleData[, labelName], method = "gbm",
    trControl = myControl)

model_rpart <- train(ensembleData[, predictors], ensembleData[, labelName],
    method = "rpart", trControl = myControl)

model_rf <- train(ensembleData[, predictors], ensembleData[, labelName], method = "rf",
    trControl = myControl)
```

I establish the label and the predictors as seperate entites for clarity going forward, and then set my Control

for the caretEnsemble package as cross validation. Afterwards I create models for three different methods.

```
blenderData$gbm_PROB <- predict(object = model_gbm, blenderData[, predictors])
blenderData$rpart_PROB <- predict(object = model_rpart, blenderData[, predictors])
blenderData$rf_PROB <- predict(object = model_rf, blenderData[, predictors])

testingData$gbm_PROB <- predict(object = model_gbm, testingData[, predictors])
testingData$rpart_PROB <- predict(object = model_rpart, testingData[, predictors])
testingData$rf_PROB <- predict(object = model_rf, testingData[, predictors])
```

The results of these models are added on as new predictors for my final set.

```
predictors <- names(blenderData)[names(blenderData) != labelName]
final_blender_model <- train(blenderData[, predictors], blenderData[, labelName],
    method = "rf", trControl = myControl)
```

After completeing the final model it was time to test the improvement.

```
predictions <- predict(object = final_blender_model, testingData[, predictors])

StackAuc <- roc(testingData[, labelName], predictions)
print(StackAuc$auc)
```

```
## Area under the curve: 0.8805
```

Stacking turned out to be a reletively simple but clearly powerful tool for improving the accuracy of my model, even with the limitations of the data set.

Questions, Conclusions, and Recommendations for Next Steps

After concluding my work on this data, I've come to realize that the questions I asked were ultimately the wrong sort of question to be asking.

User engagement with the app, while significant enough to use in our model, was less impactful than testing against the individual companies. In each criteria of engagement we saw highly 'engaged' individuals who churned, but the exact number of them is irrelevant because they were all from different companies with different company cultures and firing polices. From the information we have, there isn't a great deal of significance we can glean from common trends as a result.

So, with this in mind, we did learn some interesting details. First and foremost, of those who did churn there was a noticibly even spread across the votes. Employees with high job satisfaction and low job satisfaction, high app engagement and low app engagement. The simple conclusion to draw is that there is most likely a better set of possible data that could be gathered from these companies.

On that note, it's worth mentioning that this set of data would be more difficult to gather from a run of the mill company. Most companies aren't using an internal App that records satisfaction, it's as simple as that. However, most companies do have access to far more relevant data, such as employee tardiness, frequency of calling out of work, past performance reviews, or even simple data like how long the employee has been at the company. It is for that reason that I stand by the idea of using data science in an effort to predict and better maintain employee retention. The more intimate company data will likely have a larger impact and be able to identify company specific trends in their employee attrition.

What you should take away from this work is that, while broad stroke data from an app like this can be used for the purposes of identifying employees at risk of churning, it should be noted that results are going to be better when viewed on a specific company level. The app is more of a shortcut to what a more company specific data set could provide, and there is a great deal of room for error because of the potential for employees to lie about their vote results. Even still this model turned out to be about 88% effective, and if I

were to continue with this topic down the line I would seek out a specific company's data in an attempt to create a real model based on more practical resources.

Finally, identifying universally availible data is the most important thing that someone going forward with this topic could acquire. The appeal of this idea is how it can be employed by most businesses equally, so data that could be found in the majority of companies is key to its viability.

Acknowledgements

https://www.myhappyforce.com/en/ For providing the App

http://amunategui.github.io/blending-models/ For their Stacking insight

https://www.kaggle.com/harriken/how-many-unlikes-it-takes-to-get-fired/data For providing the Data

And Dhiraq Khanna for his expert assistance