Analysis of User Activity as of 08-05-2019

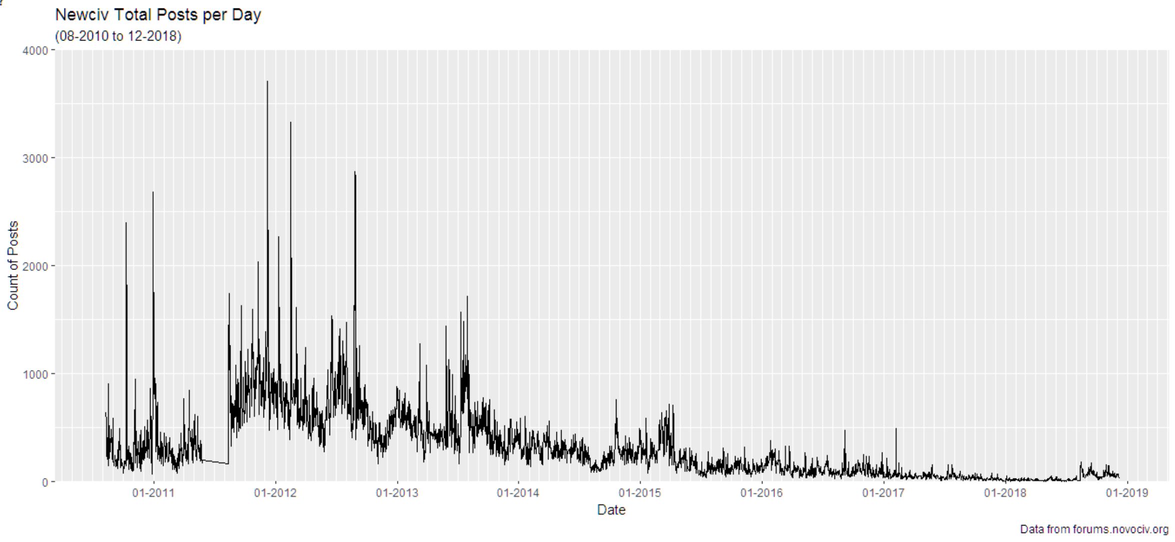
Big Data – Jared Crivello 83472 Final Project

Context

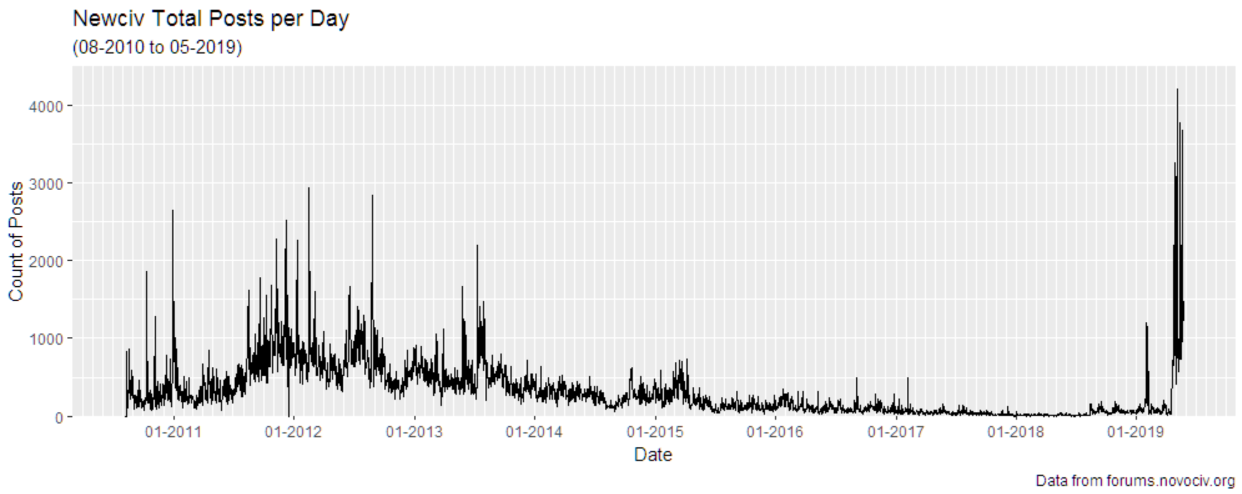
I have managed a website named NovoCiv (https://forums.novociv.org) with discussion forums since August 2010. Over the past 8.5 years the forums have amassed almost 1,000,000 posts. There are 1,135 users, and each user has a unique row in the user table, with more than 200 columns describing anything from post count to birthday. The user dataset was created by simply exporting the user table straight from the database.

In Data Visualization, I showed a graph of user activity over time. This graph has been reproduced below.

Figure ??



As I mentioned at the time, the activity of the website had declined to almost nothing. However, there was a small uptick in activity right before my presentation, and I had expressed hope that the activity would rebound. Next, I show the updated graph as of May 24th, 2019.



Because of various projects, I have managed to raise activity on the website to the highest its ever been.

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

Objective

Overall website activity is driven by the activity of the users. I want to understand what causes a user to go inactive, so that I can reduce these forces and increase the likelihood of retaining active members. In a way, this is similar to a churn analysis in a company. My aim Is to build predictive models and test their accuracy. Once I have my best model, I can use it to predict the probability of any given user to be active. By filtering the dataset to users who are currently active and sorting by the lowest probability of inactivity, I will then be able to identify which users are most at risk of going inactive.

Data Management

The code related to Data Management is found in 'Data_Prep.R'. I have to prepare my data before I can analyze it. My starting point is the \`user\` table in a MYSQL database. I begin by exporting the entire user table to a .csv file, which is then imported into R with the function read.csv().

Before any data manipulation, I must remove the observation pertaining to the website's robot. This is not a real user, and this observation would likely be influential and harmful to the model.

Next, I want to gather some additional variables about a user before performing the analysis. The dataset was created at unix time of 1557330540 (May 8th, 2019), so the new variables are based on this time.

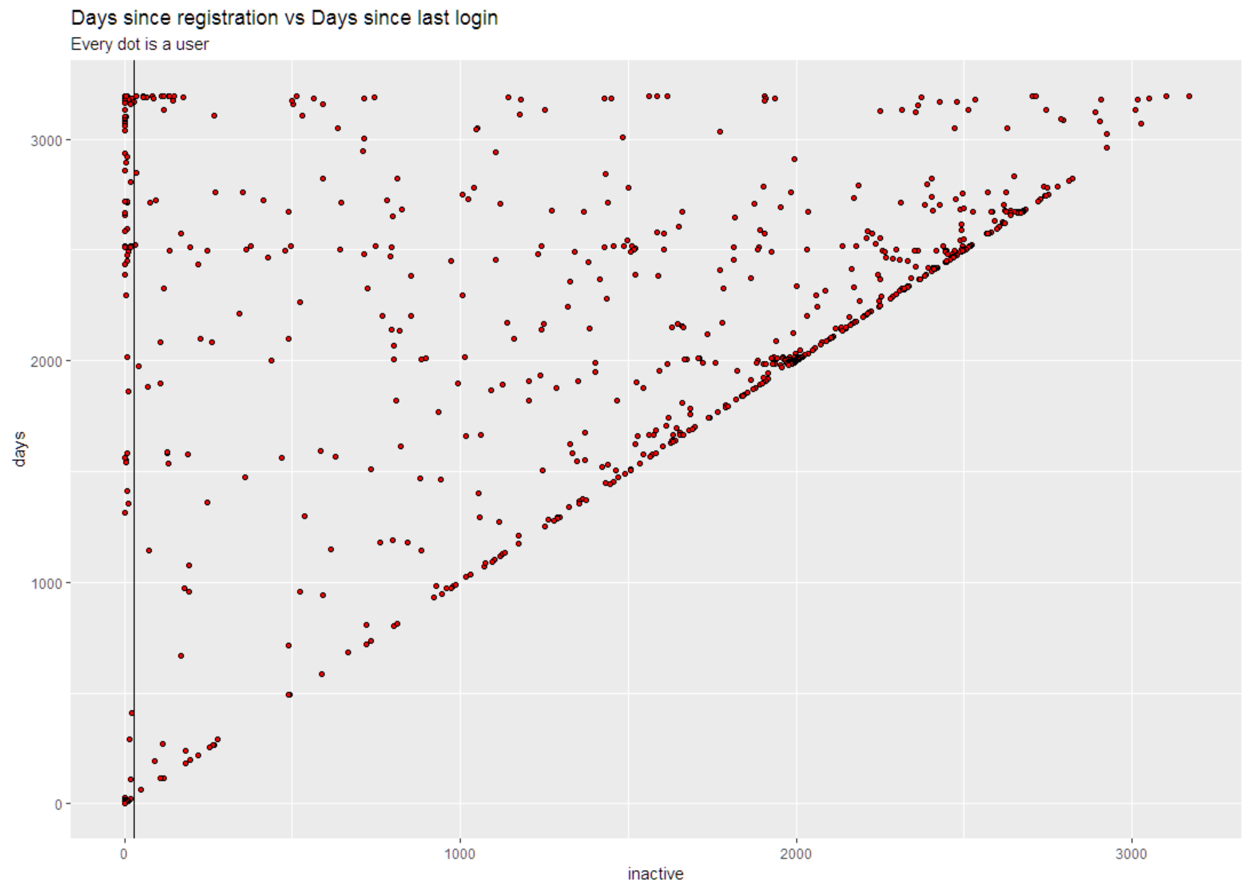
- user$active will be the binary flag for all of the future analysis. I will use all the techniques I've learned in this course to build models which predict if a user is active, given the other information about the user.
- user$inactive is the number of days since the user has last logged on.
- user$days is the number of days since the user created their account.

Although the inactive variable will not be used in any predictive models, it is useful in visualizing the state of the dataset. Below is a graph which plots inactive against days. The x axis (inactive) is the number of days since last login, and the y axis (days) is the number of days since account creation. The vertical black line at 30 days indicates the cutoff for the active flag. The cluster along the diagonal down the middle of the graph represents users who went inactive immediately or shortly after account creation, while the cluster in the top left indicates users who have been on the website since the beginning, and are still active.

Just looking at the graph, I can already see some trends in the data. For example, the large number of observations on the diagonal line indicate that many users go inactive immediately or shortly after registration. It should be easy to classify such users, as they will undoubtedly have low postcounts and other variables will be standard.

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project



With a good visualization of the data, I can move on to preparing it for analysis.

The dataset is very messy to begin with. I need to get rid of a lot of useless variables, deal with many missing values, and consolidate some groups before converting to factors. By using the BCA library, I have access to a useful function variable.summary(). I start with 133 variables.

```
market_threadcolor          factor    5.696757
ipaddress                   factor    7.449606
usertags                    factor   69.938650
homepage                    factor   71.516214
icq                         factor   72.567923
membergroupids              factor   73.619632
assetposthash               factor   86.415425
skype                       factor   97.195443
market_username_color       factor   97.458370
tvforums                    factor   97.721297
market_ct_color             factor   98.159509
aim                         factor   98.860649
msn                         factor   98.860649
market_username_glow        factor   98.948291
market_ct_glow              factor   99.561788
yahoo                       factor   99.649430
parentemail                logical  100.000000
infractiongroupids         logical  100.000000
fbuserid                   logical  100.000000
fbname                     logical  100.000000
fbaccesstoken              logical  100.000000
fbprofilepicurl            logical  100.000000
```

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

Checking the summary, I can immediately eliminate several variables from the dataset. The %NA jumps from 7% to 69%, so I decide to remove all variables which have NA greater than 10%. Although it is sometimes useful to extrapolate or impute missing data, I have verified that it would not be useful in these cases. Only four of the variables have missing values between 8% and 86%. All variables with 86% or more missing are far too empty to be useful.

Of the remaining four, I can eliminate them one by one. Usertags is a user specific string which is has encoded data. It would not be useful for analysis, even if it had no missing values. Homepage is a useless profile option, and there are other variables which will tell me the same information (that a user was modifying their profile). Icq follows the same logic as homepage. Finally, membergroupids has the same problems as usertags.

Next, I eliminate the variables which are unique to every user, such as password hash, salt, email and etc. The unique variables that I keep are:

- Userid – Easy way to track the specific user in case of investigating outliers
- Username – quick reference to the user being observed
- Ipaddress – can be used to extrapolate geographic location

Next, I remove those variables which would predict activity with 100% accuracy. If I can see the date of the last time a user was active, then of course I can extrapolate whether or not they are active today. I also remove joindate, as "days" has already used this variable to calculate a more useable value.

Next, I count the number of zeroes in each column, in order to identify variables which have no useful information. I then eliminate those which are 100% or nearly 100%.

Next I remove some variables which I know to be redundant or irrelevant.

- 'Usertitle' is the actual usertitle of a user, and less useful than 'Customtitle', which is a simple 3 level factor variable that says usertitle is default, custom set, or admin set
- 'preposts' is related to 'posts'
- 'reputationlevelid' is related to reputation function, which is unused
- 'birthday_search' is just 'birthday' but formatted different
- 'languageid' is irrelevant as there is only one language option
- 'logintype' only has one option
- 'invites' and 'caninvite' relate to a functionality that was only available for a limited time
- 'hurt_heal' only has the value of 1
- 'hh_time' is related to 'hurt_heal'
- 'tcg_free' is related to a feature that was only available for a limited time
- 'partyid' is related to a feature that was only available for a limited time
- 'pokeballs' and 'poke_team' relate to a feature that was added within the past month. Any activity on these variables would indicate with 100% certainty that a user is active.
- 'options' is a coded string used to specify specific user options
- 'timezoneoffset' would only be useful as a factor variable, but there are 24 levels. It will be better to use something like geolocation from IP addresses.

Finally, I check if any columns have a variance of 0 (implying all values are the same) and verify that it's fine. My resulting dataset has 48 columns, down from the starting 133.

Of the remaining variables, I still need to clean the data further. A summary of the recoded factor levels is listed below:

- Usergroupid – 2 (registered user), 15 (full user), all else
- Styleid – 0 (default), all else
- Showvbcode – 1 (default), all else
- Showbirthday – 0 (default), all else
- Customtitle – 0 (no title), all else
- Pmpopup – 0 (default), all else
- Maxposts – -1 (default), all else
- Startofweek – -1 (default), all else
- Referrerid – 0 (default), all else
- Market_donate_history – factor
- Market_steal_history – factor
- Market_gift_access – factor
- Market_gambling – factor

Next, I will deal with IP addresses. For some reason, 85 users have no ip address listed in the user table. I attempt to repair this table with data from posts. I grab the userids of those with no ip address, and then I paste them into a comma delimited string. This will let me quickly query the post table to find relevant ip addresses.

```
select distinct ipaddress, userid
from post
where ipaddress <> "" and userid in(<list of ids>)
group by userid
ORDER BY `post`.`userid` ASC
```

This query supplied me with ip addresses for 67 additional users. After adding these IPs to the table, I am left with 18 users which have no ip address.

At this point, I used the package “rgeolocate” with the maxmind country database. This let me determine what country each user is from, based on their IP address. For most users, this should be the country where they originally registered their account. For those users who had the “ip fix”, it will likely be the country where they made their first post.

However, the website has existed for nearly 9 years, and people can move. Looking at my own account, I can see that I am assigned the country “United States” even though I now live in Poland. In any analysis, I should be aware of the limitations of Country as a variable. As well, the number of missing values has increased to 86. This includes the 18 users who had no ip address, as well as any users with ip addresses that do not match any country in the maxmind database. I assigned a value of “Missing” to these users.

At this point, there are 46 possible values in the country variable. Having a factor variable with 46 levels is not useful, so I have to decide which way to combine them. A simple command lets me visualize the state of the data:

```
table(user$country,user$active)
```

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

```
                    0   1
  Argentina         1   0
  Australia         9   0
  Belgium           1   0
  Bolivia           1   0
  Burkina Faso      1   0
  Canada           73   5
  China            45   0
  Czechia           1   0
  Denmark           0   1
  Ecuador           1   0
  Estonia           3   0
  Finland           5   0
  France            8   0
  Germany           6   0
  Greece            1   0
  Hong Kong         1   0
  India             2   1
  Ireland           4   0
  Jamaica           1   0
  Japan             1   0
  Kazakhstan        1   0
  Latvia            1   0
  Macao             1   0
  Malaysia          2   0
  Mexico            3   1
  Missing          86   0
  Netherlands       3   0
  New Zealand       5   0
  Panama            1   0
  Philippines       1   0
  Poland           14   2
  Portugal          2   1
  Republic of Korea 1   0
  Romania           2   0
  Russia           36   0
  Seychelles        3   0
  Singapore         1   0
  Slovenia          1   0
  Spain             2   0
  Sweden            6   0
  Taiwan            0   1
  Thailand          1   0
  Ukraine         246   0
  United Kingdom   19   3
  United States   440  70
  Venezuela         6   0
```

One approach could be to make five categories: USA, Canada, Europe, Other and Missing. This makes logical sense, but still leaves me with 5 factor levels. I can then do a logistic regression with just this variable, in order to determine the statistical significance of these factor levels, and decide if it is possible to further combine them.

It is worth noting here, that I have chosen to not include Russia or Ukraine in the Europe category. Although it may not be politically correct, I can say from experience that the vast majority of accounts from these two countries were spambots. I only know of one legitimate user from Russia (who is now inactive), and no legitimate users from Ukraine.

After some data manipulation, I'm left with this table:

```
                0   1
  Canada       73   5
  Europe       79   7
  Missing      86   0
  Other       371   3
  United States 440  70
```

By doing a logistic regression with just this variable, United States as the reference Category, I get:

```
Coefficients:
               Estimate Std. Error z value            Pr(>|z|)    
(Intercept)     -1.8383     0.1287 -14.286 < 0.0000000000000002 ***
countryCanada   -0.8427     0.4799  -1.756               0.079 .  
countryEurope   -0.5853     0.4148  -1.411               0.158    
countryMissing -16.7278   703.3542  -0.024               0.981    
countryOther    -2.9793     0.5938  -5.017         0.000000524 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here I can see that "Other" is statistically significant when compared to "United States", whereas Canada, Europe and Missing are not significantly different. Canada is almost at the threshold, but giving it a category while combining Europe with the US would not make logical sense. Therefore, I decide to combine US, Canada, Europe and Missing into a general "First World" category, and leave "Other" as everything else.

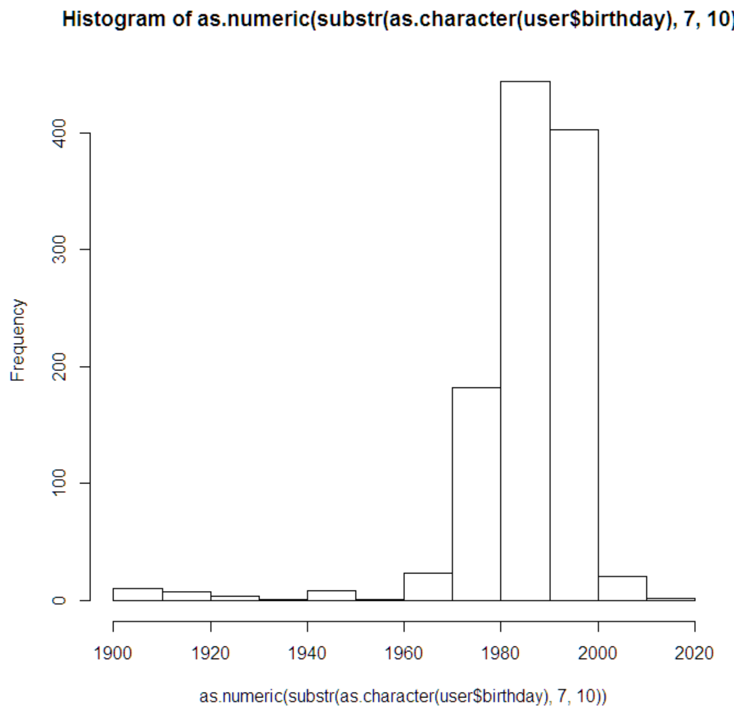
Performing a second logistic regression shows a much more reasonable result. Note that I also include the exp() function from the MASS library, in order to view odds ratio and confidence intervals.

```
Coefficients:
                   Estimate Std. Error z value             Pr(>|z|)    
(Intercept)         -4.8176     0.5797  -8.311 < 0.0000000000000002 ***
countryFirst World   2.7052     0.5913   4.575           0.00000477 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> exp(cbind(coef(GLM.Test2), confint(GLM.Test2)))
                                     2.5 %      97.5 %
(Intercept)         0.008086253 0.002004708  0.02109653
countryFirst World 14.956735436 5.551394661 61.25413524
```

I can interpret these results as showing that there is a statistically significant difference between users from the First World, and users from elsewhere. Users from the First World are nearly 15 times more likely to be active than users from anywhere else. With the Country variable set, I can remove ip address.

The final variable to consider is birthday. There are two problems with this variable. The first is obvious: there are missing values because some users choose to not declare their birthday. The second problem can be seen below:



Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

Some users have declared “fake” or “joke” birthdays. The oldest member that I am aware of is in his 30s, with a birth year in the 1980s. Anything before 1980 or after 2005 is likely fake. Birthdays between 1980 and 2005 could also be fake. I can’t do anything about fake values that look reasonable, other than recognize that this variable is not 100% accurate. Through some data manipulation, I end up with four categories: 1980s, 1990s, 2000s, and other.

Finally, my data looks to be ready. A variable.summary(user) shows me that there are no missing values, no variables (other than username) with more than 4 factor levels, and no variables using numeric values to represent something non numeric. I’m left with 43 variables, 38 of which could potentially be explanatory.

```
                              Class %.NA Levels Min.Level.Size            Mean           SD
userid                      integer    0     NA             NA   1050.77865961  419.7390433
usergroupid                  factor    0      3             58              NA           NA
username                     factor    0   1134              1              NA           NA
styleid                      factor    0      2            234              NA           NA
showvbcode                   factor    0      2             39              NA           NA
showbirthday                 factor    0      2            113              NA           NA
customtitle                  factor    0      2            104              NA           NA
posts                       integer    0     NA             NA    819.79805996 4122.6369719
pmpopup                      factor    0      2            138              NA           NA
birthday                     factor    0      4             29              NA           NA
maxposts                     factor    0      2             60              NA           NA
startofweek                  factor    0      2            242              NA           NA
referrerid                   factor    0      2            303              NA           NA
pmtotal                     integer    0     NA             NA     32.10141093  152.7951728
pmunread                    integer    0     NA             NA      0.59876543    2.1669992
profilevisits               integer    0     NA             NA     90.36684303  352.2620652
friendcount                 integer    0     NA             NA      1.98412698    8.0499168
friendreqcount              integer    0     NA             NA      0.16754850    0.6133019
vmunreadcount               integer    0     NA             NA      0.54938272    0.5735680
ucash                       numeric    0     NA             NA    978.85962553 5844.5855659
wikiedits                   integer    0     NA             NA      0.86948854    7.9997068
wikicreations               integer    0     NA             NA      0.22310406    2.6262566
gameroom_cash               integer    0     NA             NA    289.22045855 1254.0138757
timespentonline             integer    0     NA             NA 719535.99382716 2930959.4302438
market_donate_history        factor    0      2             55              NA           NA
market_steal_history         factor    0      2             42              NA           NA
market_gift_access           factor    0      2             57              NA           NA
market_purchases            integer    0     NA             NA      5.25132275   32.8412342
market_gambling              factor    0      2             42              NA           NA
market_bank1                numeric    0     NA             NA    788.73017665 5927.3485662
dbtech_usertag_mentioncount integer    0     NA             NA      0.88447972    5.3775273
dbtech_usertag_tagcount     integer    0     NA             NA      0.06525573    0.4555930
dbtech_usertag_mentions     integer    0     NA             NA     11.21516755   51.5245723
dbtech_usertag_tags         integer    0     NA             NA      0.66931217    3.3389028
dbtech_usertag_quotes       integer    0     NA             NA      2.26014109   16.2068562
post_thanks_user_amount     integer    0     NA             NA     10.06966490   89.2438644
post_thanks_thanked_posts   integer    0     NA             NA      7.02292769   38.0495961
post_thanks_thanked_times   integer    0     NA             NA      9.93298060   53.8031037
active                      numeric    0     NA             NA      0.07495591    0.2634364
inactive                    numeric    0     NA             NA   1642.07583774  803.7972894
days                        numeric    0     NA             NA   2106.48765432  646.5775670
life                        numeric    0     NA             NA    464.41181658  880.8258966
country                      factor    0      2            374              NA           NA
```

At this point, I save the dataset to am RDS file. The RDS File included with the project has had usernames and userids obfuscated, in order to protect user privacy.

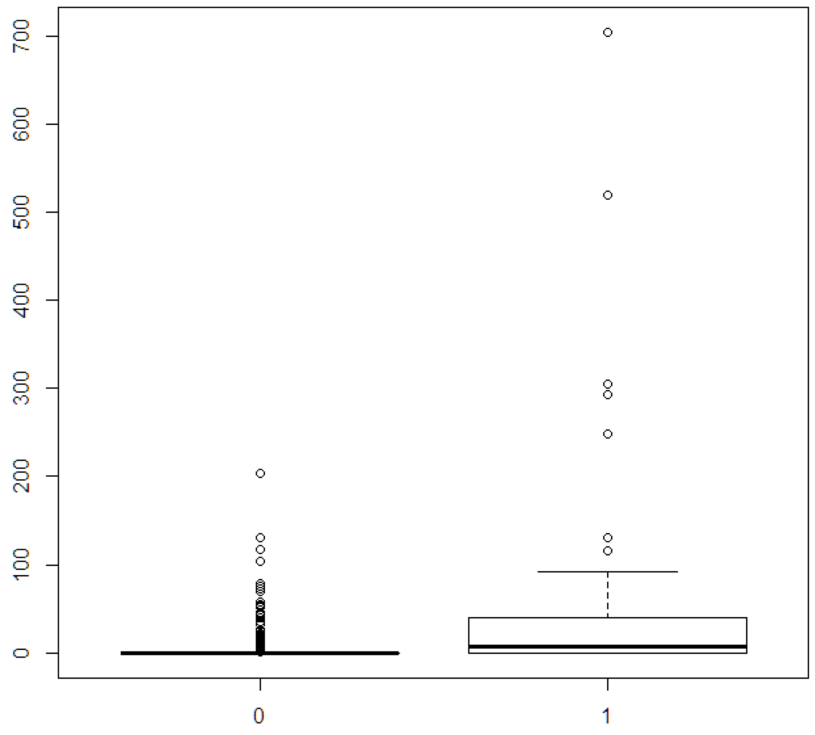
Data Analysis

At this point, the file 'Project_Analysis.R' can be used. I start with a simple logistic regression of all potential explanatory variables, to observe what is statistically significant and what is not. I tried to fit all of the variables into the model, but I encountered problems.

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

First, I realized that the dbtech variables were very highly related (they control aspects related to "tagging" another user, so a user who uses this feature would have high values in each dbtech variable). I kept the one variable that was the most interesting, (dbtech_usertag_mentioncount, which indicates the number of times the user has been tagged by others) and removed the others. I found the same problem with the post_thanks variables, and kept the variable which indictes the number of times a user's posts have been liked.

Finally, I had to remove market_purchases from the model. Doing a boxplot shows that this variable on its own is too powerful to include in a complicated model:



Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

Users who have more than 210 market purchases are guaranteed to be active. In fact, there are only four users out of 1134 who have more than 100 market purchases and are inactive. In this case, I have discovered one variable that is very useful for determining whether or not a user is going to stay active. Users who have a lot of interaction with the market can be classified as safe. But what about users who have under 100 purchases? In order to understand them, I will need to create models without this variable. However, when I get to decision trees, this variable will be an excellent addition to any model.

With these variables sorted out, I have my first general logistic model:

```
GLM <- glm(active ~  usergroupid + styleid + showvbcode + showbirthday
+ customtitle + posts + pmpopup + birthday + maxposts + startofweek +
referrerid + pmtotal + pmunread + profilevisits + friendcount +
friendreqcount + vmunreadcount + ucash + wikiedits + wikicreations +
gameroom_cash + timespentonline + market_donate_history +
market_steal_history + market_gift_access + market_gambling +
market_bank1 + dbtech_usertag_mentioncount + post_thanks_thanked_times
+ days + country, data=user, family=binomial(logit))
```

Running an Anova test shows that a lot of variables are not significant:

```
                            LR Chisq Df        Pr(>Chisq)
usergroupid                    6.400  2         0.0407585 *
styleid                        0.114  1         0.7358951
showvbcode                     1.600  1         0.2058486
showbirthday                   4.596  1         0.0320474 *
customtitle                    0.070  1         0.7915553
posts                          0.627  1         0.4286053
pmpopup                        0.808  1         0.3686008
birthday                       1.385  3         0.7089897
maxposts                       3.253  1         0.0712744 .
startofweek                    0.150  1         0.6984444
referrerid                     0.032  1         0.8576260
pmtotal                        3.549  1         0.0595844 .
pmunread                       0.357  1         0.5501333
profilevisits                  1.382  1         0.2397876
friendcount                    2.715  1         0.0993864 .
friendreqcount                 4.316  1         0.0377647 *
vmunreadcount                  0.592  1         0.4415916
ucash                          0.148  1         0.7003830
wikiedits                      3.385  1         0.0657975 .
wikicreations                  1.082  1         0.2982578
gameroom_cash                  0.017  1         0.8966566
timespentonline                2.833  1         0.0923711 .
market_donate_history          0.410  1         0.5219220
market_steal_history           2.576  1         0.1084888
market_gift_access             4.982  1         0.0256172 *
market_gambling                0.344  1         0.5577016
market_bank1                   2.660  1         0.1028810
dbtech_usertag_mentioncount    2.123  1         0.1450915
post_thanks_thanked_times     11.797  1         0.0005933 ***
days                          50.543  1 0.000000000001166 ***
country                        1.410  1         0.2351280
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In order to make a more productive regression model, I next performed a stepwise selection algorithm. The result is a much smaller model:

Step: AIC=288.68

```
active ~ usergroupid + showbirthday + maxposts + startofweek +
pmtotal + friendcount + friendreqcount + wikiedits + timespentonline +
market_steal_history + market_gift_access + market_bank1 +
post_thanks_thanked_times + days + country
```

Here, the Anova function shows each variable as significant, at least to the 0.05 level. Calculating the odds ratios gives me some insight to the way the variables influence user activity.

```
                                          2.5 %     97.5 %
(Intercept)                0.10698905 0.01615736  0.5034756
usergroupidFull            1.76453603 0.57472104  6.1692313
usergroupidOther          13.83545444 3.67261933 58.3534645
showbirthday1              2.95944586 1.19112795  7.3133303
maxposts1                  3.07845908 1.09061186  8.4728631
startofweek1               4.96811326 1.94401440 13.2839134
pmtotal                    1.00327657 1.00145863  1.0055641
friendcount                1.03196011 0.99962893  1.0696721
friendreqcount             1.73865650 1.07962214  2.7734830
wikiedits                  0.93180346 0.84714377  0.9952222
timespentonline            1.00000013 0.99999998  1.0000003
market_steal_history1      4.32966524 1.00879710 17.5596655
market_gift_access1        0.06974483 0.01112989  0.3781546
market_bank1               1.00004104 0.99999555  1.0000814
post_thanks_thanked_times  1.01461112 1.00727769  1.0229578
days                       0.99794250 0.99747153  0.9983717
countryFirst World         3.26239464 0.84398981 17.8851457
```

I can make some general observations about the odds ratios:

- When comparing the “Full” usergroupid to “Basic”, the confidence intervals cross 1.0, making the effect statistically insignificant. However, when looking at “Other” usergroups, there is a very clear effect. Users who have an “Other” usergroup (mainly moderators) are nearly 14 times more likely to be active than those who have a “Basic” usergroup.
- Users who have settings outside of the default (showbirthday, maxposts, startofweek) are 3-5 times more likely to be active than those who stick to default account settings.
- For each Private Message a user has received, they are very slightly more likely to be active.
- For each wiki edit a user has, they are about 7% less likely to be active. This matches the anecdotal evidence, as I recall a few users who were very active in contributing to the wiki and are now inactive.
- For each time a user’s posts were liked, they are roughly 1.5% more likely to be active.
- For each day of seniority that a user has, they are about 0.2% more likely to be inactive.

All of the other effects are too erratic to consider. Now that I have performed a statistical analysis of the data, I would like to build various predictive models using the various tools that we have learned throughout the course.

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

Predictive Modeling

Error Testing Function

In order to understand the effectiveness of my predictive models, I need a standard way to test them. Previously in class, we simply rounded the predictive probabilities and then compared them to actual, counting the number of bad predictions. However, that approach does not tell the full story.

First of all, with my use case the occurrence of a “False Positive” is much worse than a “False Negative”. Positive (Active=1) indicates that a user is active. A false positive is the case where a user is predicted to be active, when they are actually inactive. Alternatively, a False negative is the case where a user is predicted to be inactive, when they are actually active.

I would rather be warned of a user being at risk of inactivity when the user is actually active, than be told a user is active when they are at risk of inactivity. Further, most cases in the dataset are inactive, so if a model predicts that everyone as inactive it would still have a relatively low error rate. In the example below, although the model only has 30 errors, it is predicting a large number of False Positives, which is bad. I would want to raise the threshold, to reduce the number of positive predictions.

```
               Active                                    Active
predvals   0   1                       predvals   0   1
PR     0 285   3                       PR     0  TN  FN
  ED   1  30  23                         ED   1  FP  TP
```

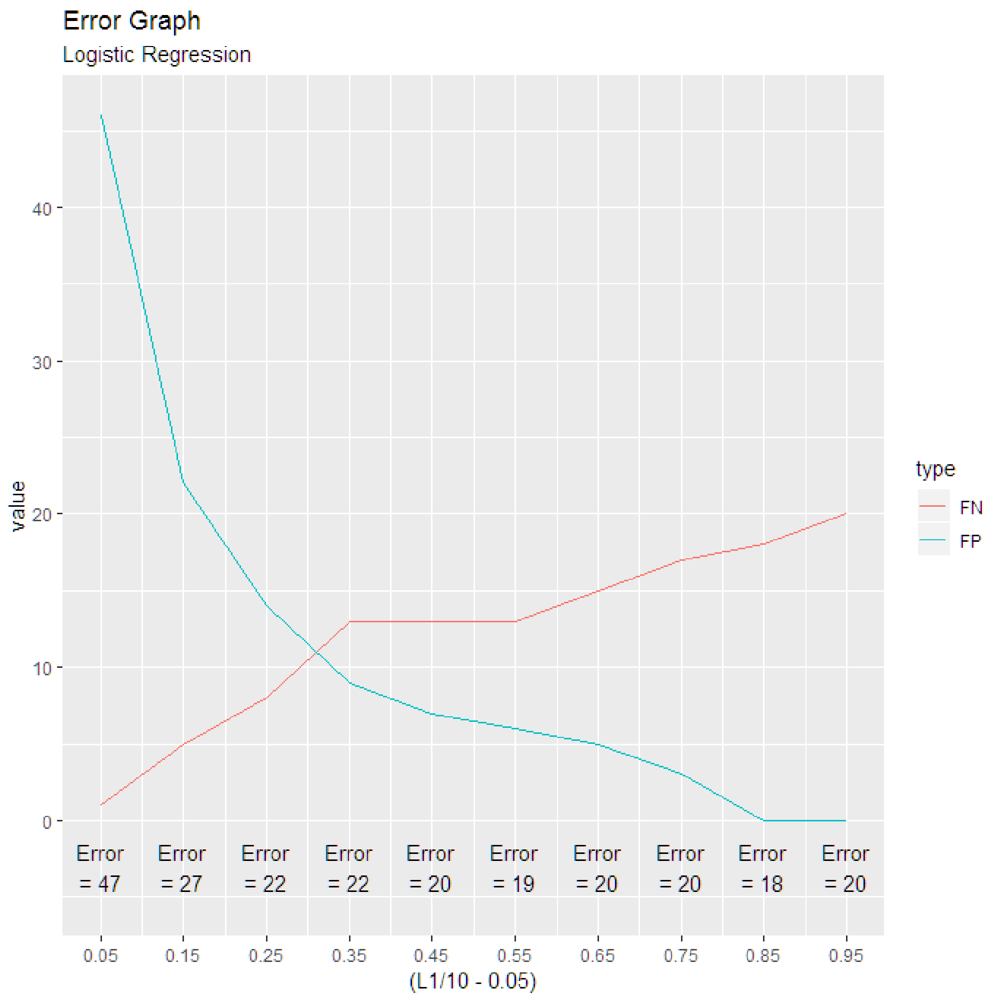
The next issue to consider is the threshold of probability for declaring a prediction one way or the other. Simply rounding the prediction puts this threshold at 50%. However, it may be beneficial to raise or lower the threshold, depending on the “level of risk” that I want to consider. To solve these two issues, I decided to create a formula that will graph the FP, FN, TP and TN rates for various thresholds. There is a bug in ggplot that makes calling a ggplot inside of a function a difficult task. I will have to include the ggplot code each time I wish to make a plot.

```
test_graph <- function(pred,valid) {
  obs <- length(pred)
  out <- list()
  for(i in 1:10){
    thresh <- i/10 - 0.05
    predvals <- as.numeric(pred>thresh)
    out[[i]] <- matrix(table(predvals,valid$active),2,2)
  }
  return(out)
}
melt_fun <- function(pred,valid) {
  melt_all <- list()
  melt <- melt(test_graph(pred,valid))
  melt$type <- ifelse(melt$Var1==1 & melt$Var2==1, "TN",
                      ifelse(melt$Var1==2 & melt$Var2==2, "TP",
                             ifelse(melt$Var1==1 & melt$Var2==2, "FN",
                                    ifelse(melt$Var1==2 & melt$Var2==1, "FP", "NA"))))
  melt2 <- melt[melt$type %in% c("TP","TN"),]
  melt2$sum <- ave(melt2$value, melt2$L1, FUN=sum)
  melt2 <- unique(melt2[c("L1","sum")])
  melt_all[['melt']] <- melt
  melt_all[['melt2']] <- melt2
  melt_all[['obs']] <- length(pred)
  return(melt_all)
}
```

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project

The melt_fun() formula takes the predicted probabilities and the validation set, and creates the data for an error chart.
A test of the function on predictions with my previous logistic model produces a list of 10 confusion matrices. Plotting the errors gives me a much more useful representation of my model:

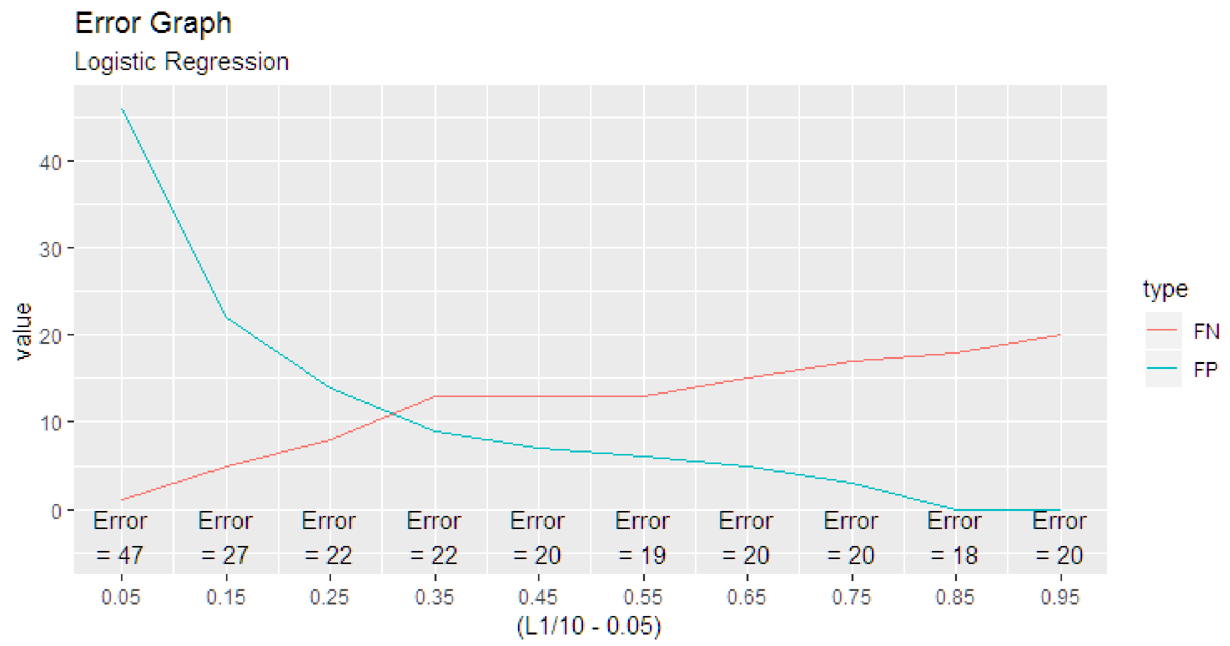


Now I can see that a threshold of 0.85 gives me the lowest amount of total errors, while providing 0 false positives. With this type of graph, I can compare my different predictive models on their performance in my intended use case, and understand what threshold I should set for the model predictions.

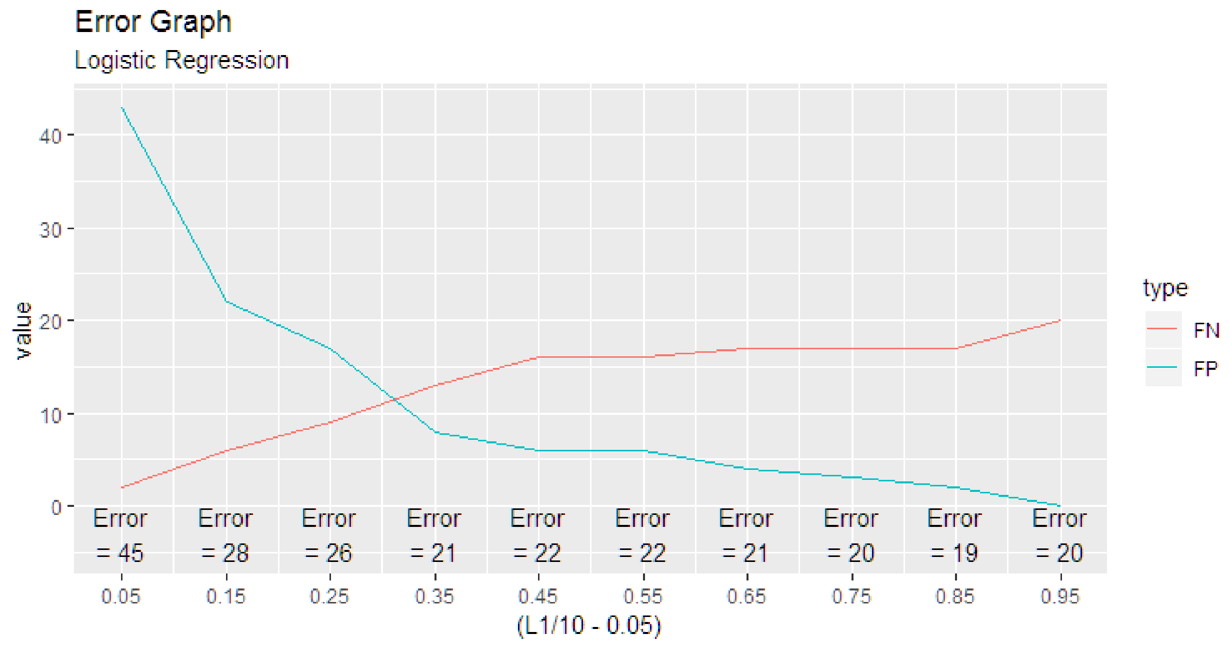
The Models

Logistic Regression

For predictive modeling, more variables can be better. Even if the variables are not very significant, they can still increase the power of predictions. However, when I test the original logistic regression using all variables, the resulting prediction is actually weaker than the reduced model. The lowest number of errors while avoiding false positives is 20. 2 more errors than the reduced model! It is also more complicated, and thus harder to use and interpret.

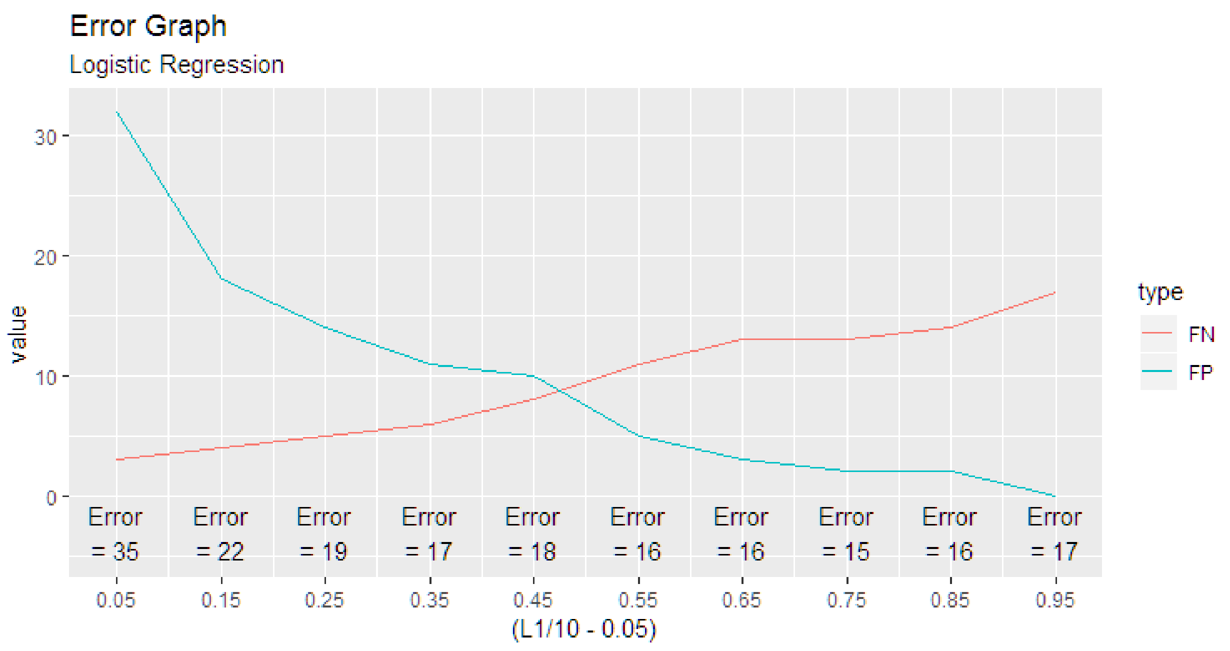


My next model is a mixed model, where I take the log values of any numerical variables with large Standard Deviations. These variables are: posts, pmtotal, profilevisits, ucash, gameroom_cash, timespentonline, market_bank1, and days.



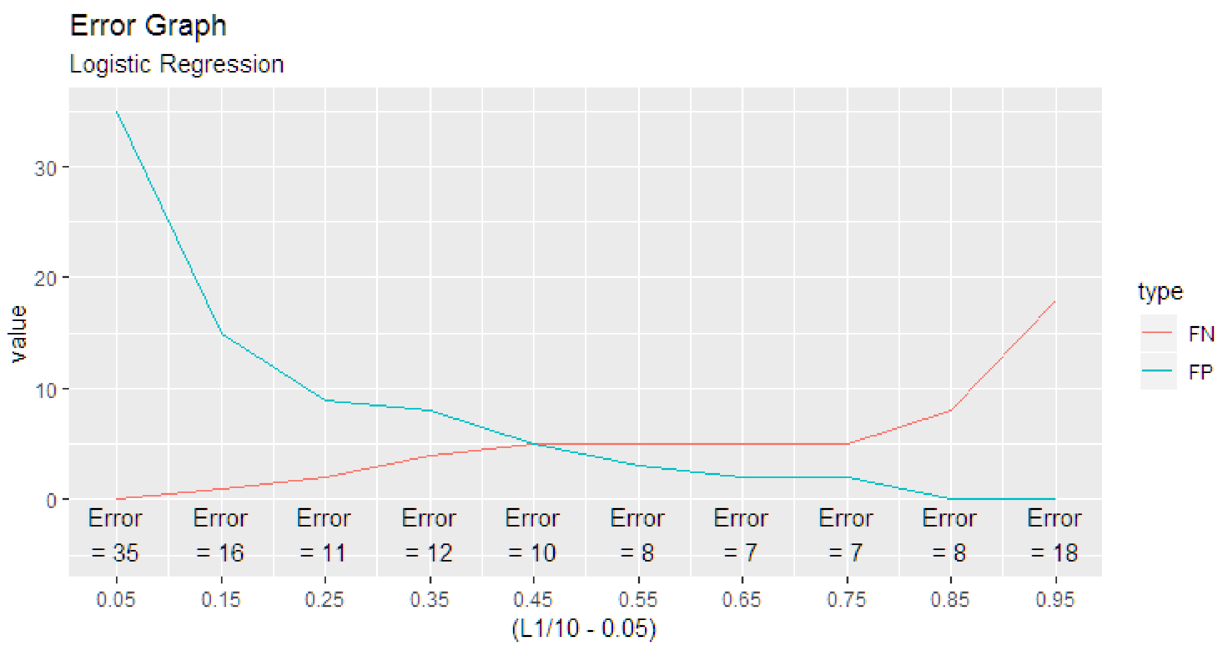
The mixed model will all variables seems to perform better overall, though it can never fully remove the false positives. So, I once again performed the stepwise selection, this time on the mixed model. The result is shown below:

```
active ~ styleid + showbirthday + customtitle + logposts + pmpopup + maxposts +
friendreqcount + loggameroom_cash + dbtech_usertag_mentioncount +
post_thanks_thanked_times + logdays
```



Using just these mixed variables, I end up with a second mixed model, with error graph as shown above. This mixed model is the best one so far, with minimum errors of 15 or 17 errors with 0 False Positives. For the overall tool comparison, this will be the model representing logistic regression.

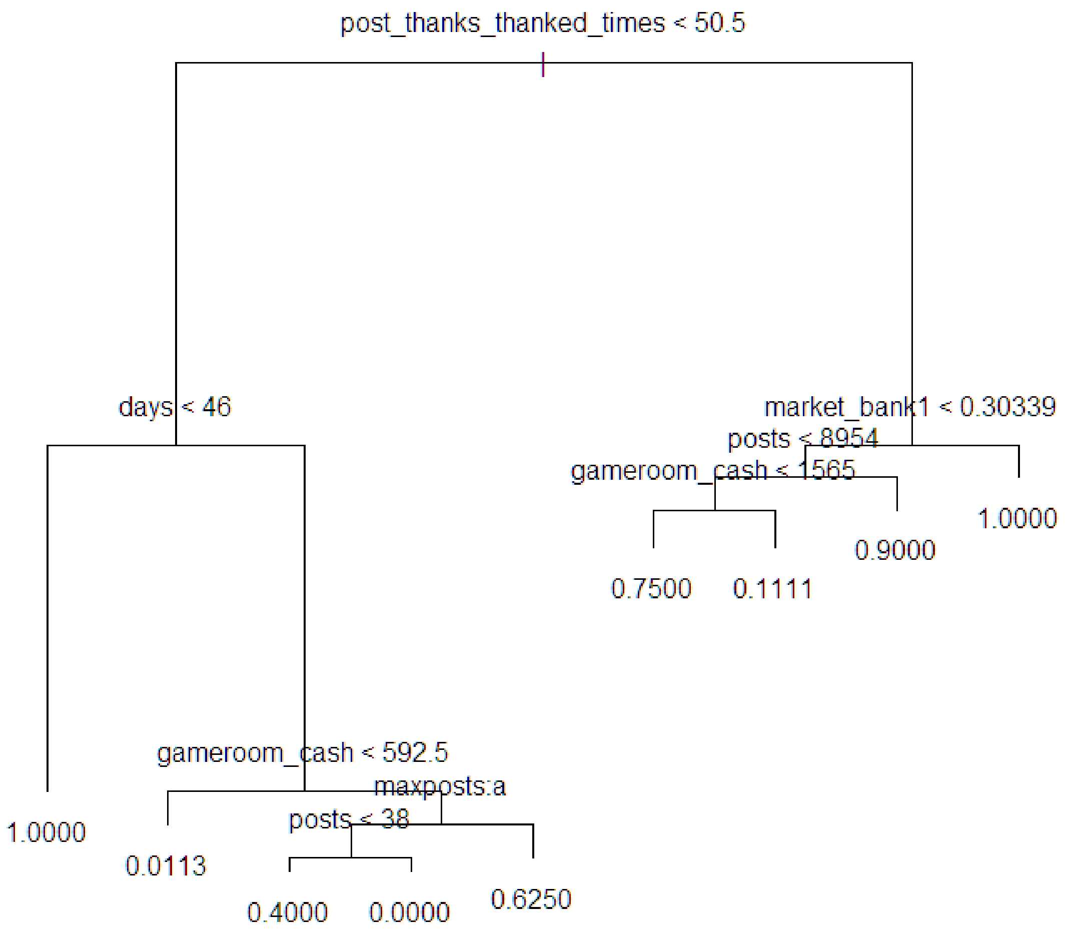
Later in the analysis, when preparing decision trees, I noticed that the training set was classifying "Days < 76" as 100% chance to be active. Actually, days of 30 or less is by definition going to be active, as an account can only become inactive after 30 days with no activity. This led me to think the data could be sampled differently, so I changed the random seed.



The resulting training set was much more effective, and the error graph for this new validation set shows that at a threshold of 0.85, the model can have only 8 errors out of 341 (2.35%), with zero cases of False Positive.

Decision Trees

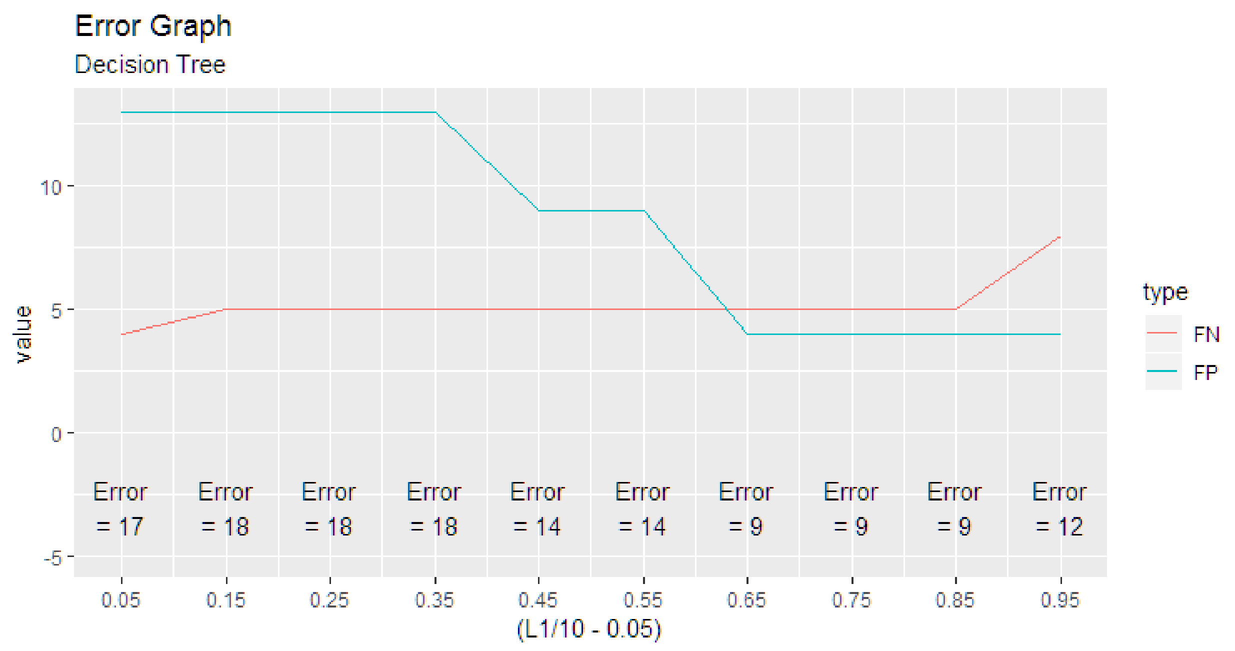
The next tool for predictive modeling is decision trees. I start by doing a simple decision tree with all of my variables.



This decision tree is extremely easy to follow and interpret, and it follows a logic that makes sense, given my understanding of the active users. A user who has more than 50 posts "liked" will probably be very engaged with the community and well known. Most users have a balance close to 0 in market_bank1 (because the bank serves no purpose), so there must be only one or two examples that teach the decision tree to classify a bank balance as 100% chance to be active. Given no bank balance, a user with a lot of likes and a lot of posts is very likely to be an active member of the community. For the last branch on this side, a user with a high gameroom_cash balance is likely to be inactive, as active users would likely have traded in their gameroom cash for the more relevant currency ucash.

Looking to cases where a user has less than 50 likes, the decision tree has clearly picked up on the fact that an account that is near or less than 30 days old is going to be active, because they have not existed long enough to be inactive. The remaining cases for older accounts do not make logical sense to me.

Looking at the error chart, I can see that the decision tree is fairly good at predictions, but it can not eliminate the false positive at any threshold.

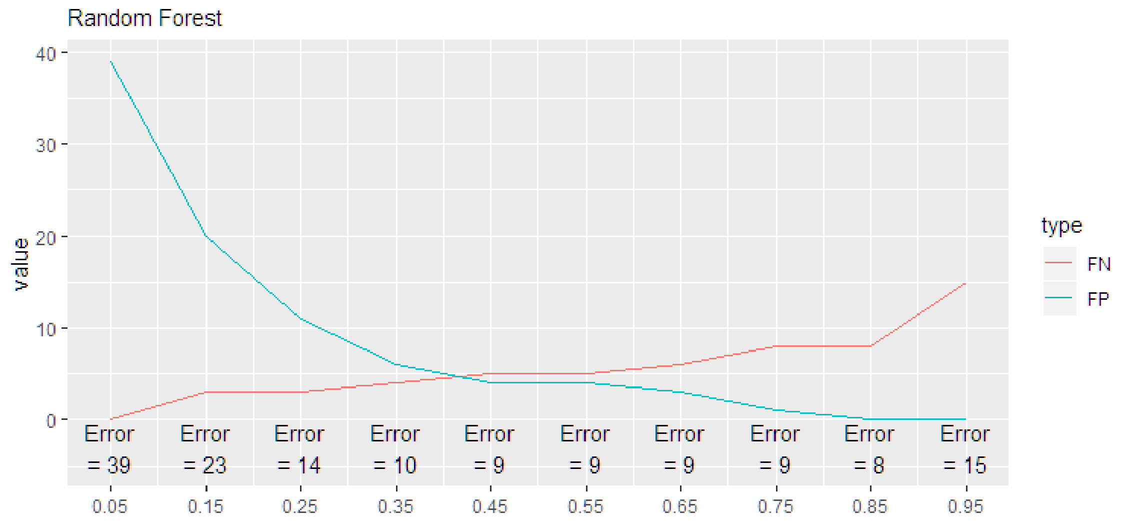


In order to better understand the fringe cases, I look to rattle. The decision trees produced by rattle will show the number of cases that have reached a given node on the decision tree. The rattle decision tree confirms my suspicions. The number of cases with likes > 51 and bank balance greater than 0.3 are only 13. These 13 users happened to all be active, but this does not define any sort of hard rule, such as days under 30. The decision tree treats it as a hard rule, and as such can't avoid the false positives. The error plot for rattle is almost identical to the error plot for tree package.

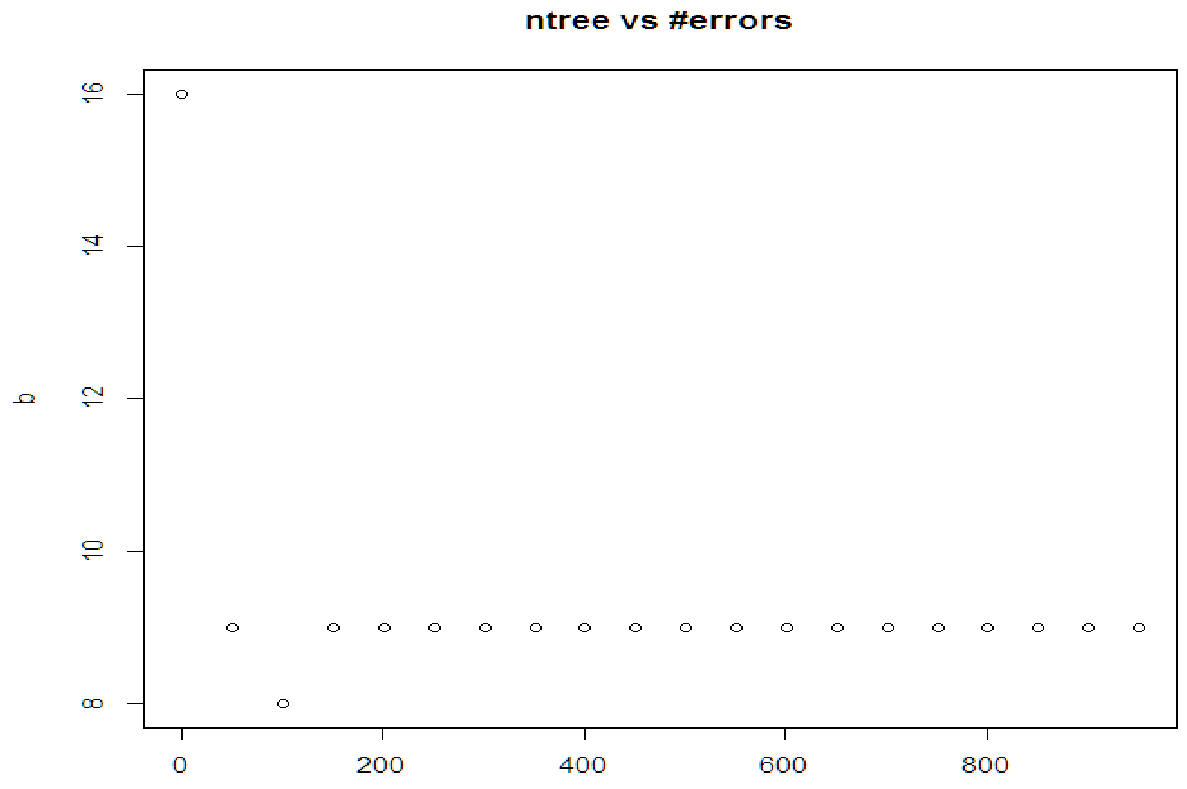
Although the decision tree does not satisfy my use case needs, it was very helpful in showing me how my training data set behaved, which led to the resampling that more than halved the number of errors in the mixed regression model.

Random Forest

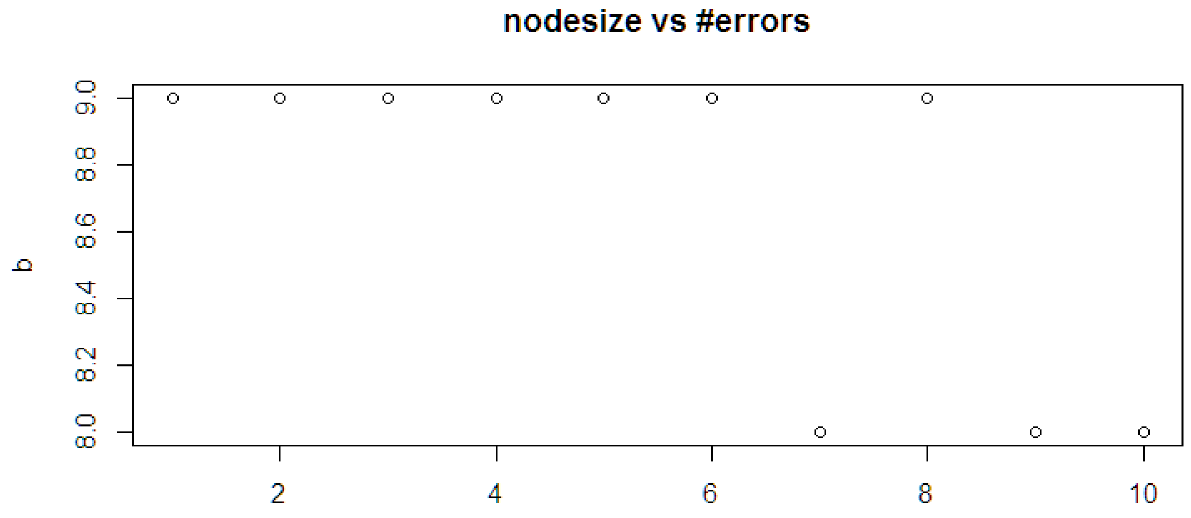
Beyond the logistic regression or decision trees are the more complicated predictive tools. The first one that I will be using is Random Forest. My first step is to simply run a standard model with ntrees=100, in order to understand the error chart.



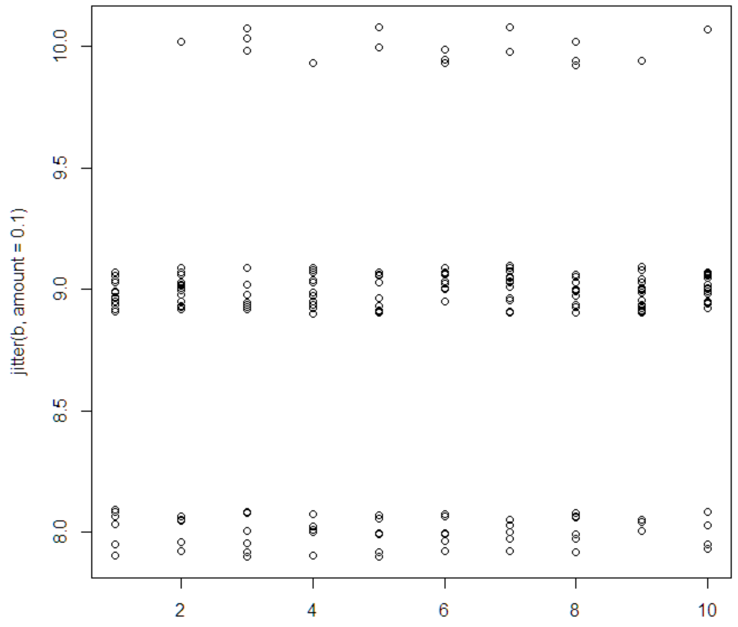
Without any optimization, the random forest is almost as good as the best logistic regression model. As usual, I see that the best threshold to eliminate False Positives is 0.85, so I will stick to that while optimizing the Random Forest model. Next, I want to determine the ideal number of ntrees which will provide the best training, without overfitting the model to the training data. To do that, I set up a simple loop that can iterate over several values of ntrees.



This graph shows me that the RandomForest model performs best at an ntrees value of 100. This is the same value that was already used, suggesting that I can't improve the model by changing the ntrees parameter. Next, I do the same observation for difference in the minimum node size. This parameter determines how many observations must land in a specific node.



However, I noticed that this graph was producing a different result each time I ran it. This is likely due to the random aspect of the random forest calculation. In order to understand which value of nodesize would be best, I ran this analysis 20 times.



```
> table(b,test)
    test
b     1  2  3  4  5  6  7  8  9 10
  8   7  5  7  7  9  6  5  6  3  4
  9  13 14 10 12  9 11 13 11 16 15
  10  0  1  3  1  2  3  2  3  1  1
```

By adding some random noise through the "jitter" function, I am able to visualize how each level of nodesize behaves. The visible trend seems to be that the Random Forest will usually result in 9 errors, regardless of nodesize. However, I can summarize the data with table() and get a more precise understanding. With this table I can see that a minimum nodesize of 5 produced the largest number of models with only 8 errors. Although the effect seems minimal, I will use a nodesize of 5 in the final model.

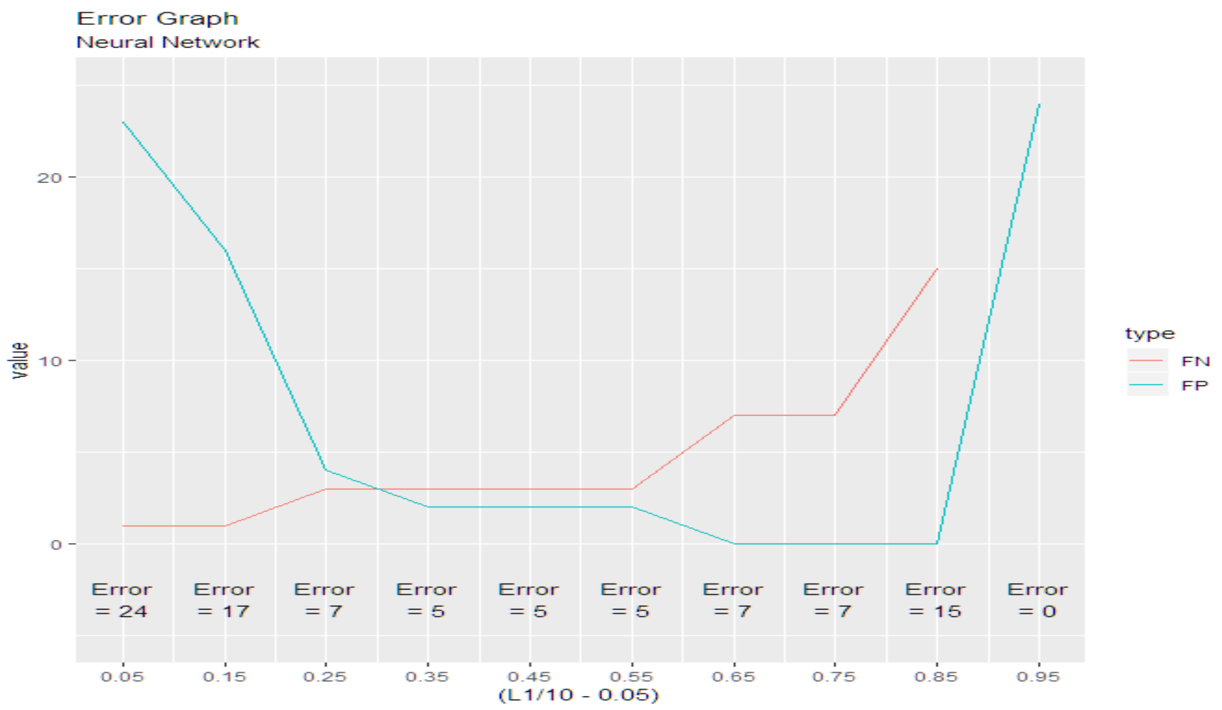
Finally, I tried out the experimental corr.bias parameter, which should adjust the model based on the correlation between variables. However, 20 models with the parameter performed about the same as without the parameter. There were in fact fewer models with 8 errors than the previous test. That means that my final model will use ntrees=100, nodesize=5, and corr.bias=FALSE.

Neural Network

The final tool at my disposal is the Neural Network. Using the nnet package with the RcmdrPlugin.BCA plugin, I generated a test model with 4 neurons, in order to understand the error chart. I have to limit the y axis to 25, because at the extremes the neural net completely fell apart and generated hundreds of errors.

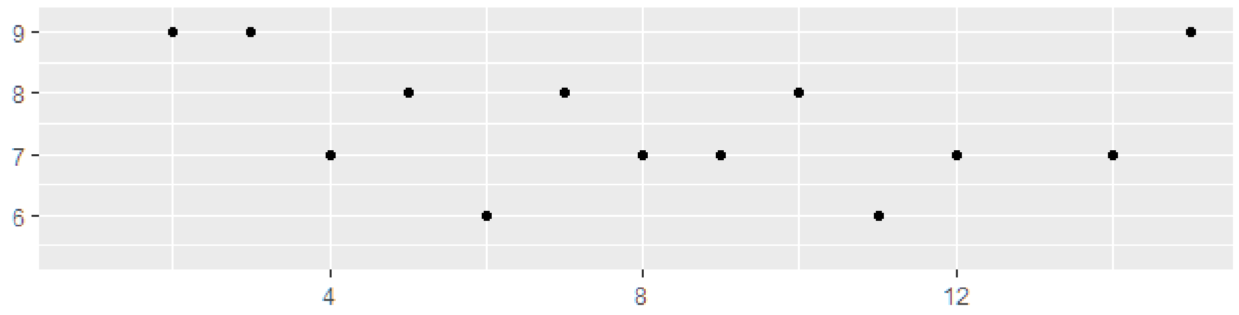
Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project



Unlike the previous models, the Neuralnet seems to perform best at the threshold of .65 or .75. The total errors (with no false positives) is also the lowest so far, at 7 errors out of 341 observations. But I can still improve the model.

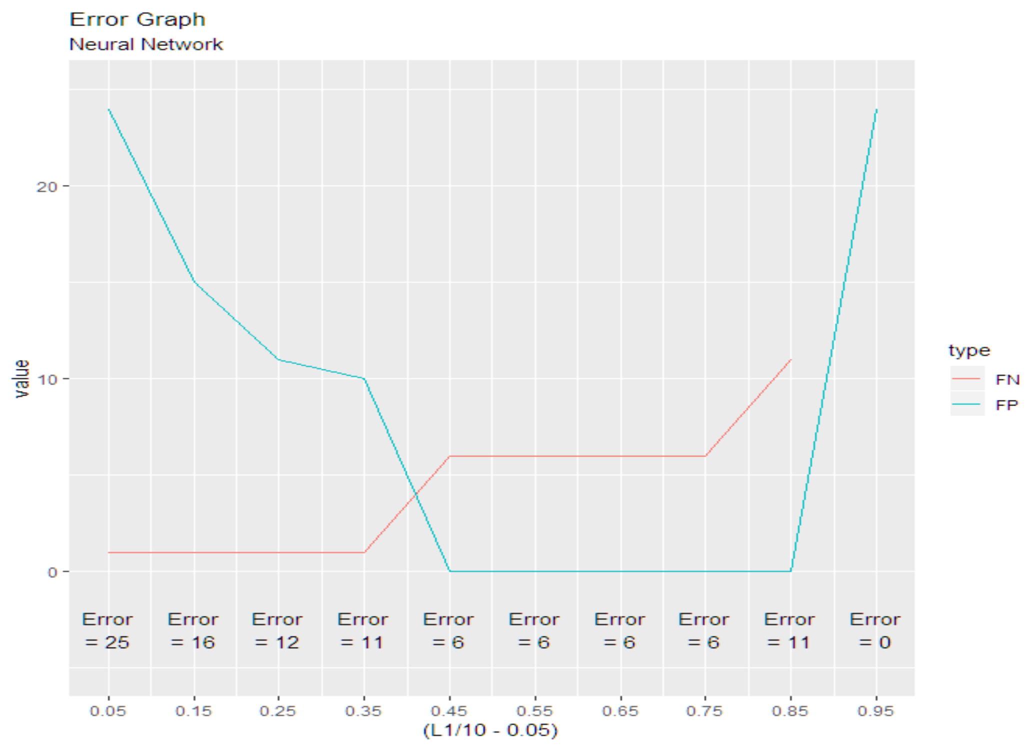
My first step is to determine the best number of neurons. I already know from my lab homework that the neural net can easily be overfit. However, by training on the training set and testing on the validation set, I should be able to avoid overtraining problems.



Here I can see that the lowest number of errors (6) occurred at 6 neurons and 11 neurons. Less neurons means a faster calculation speed, so I could say that 6 neurons are ideal. With 6 neurons, the error chart produces amazing results.

Analysis of User Activity as of 08-05-2019

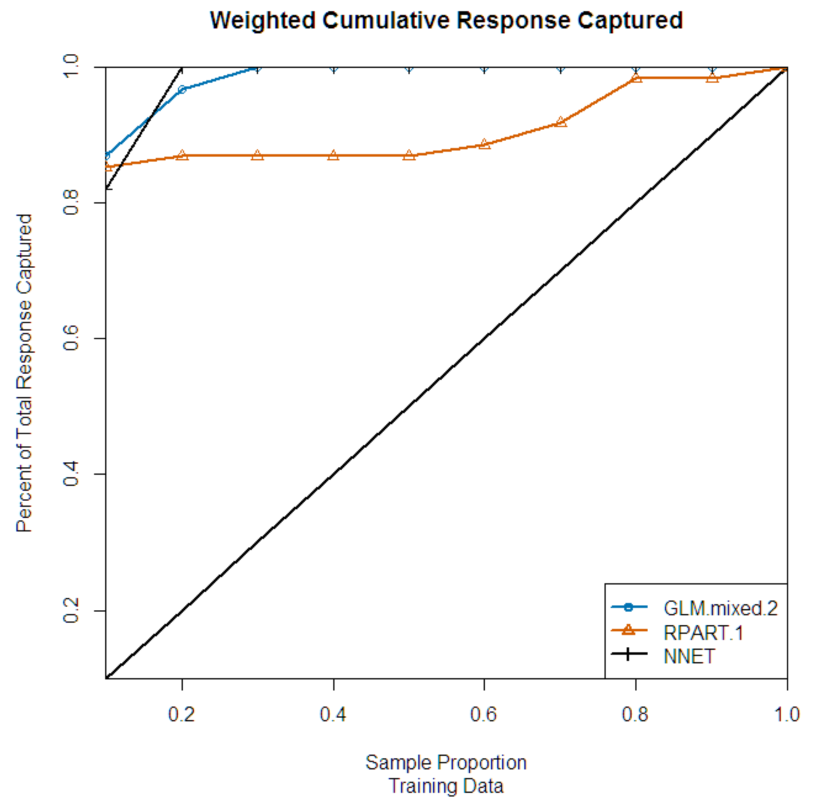
Big Data – Jared Crivello 83472 Final Project

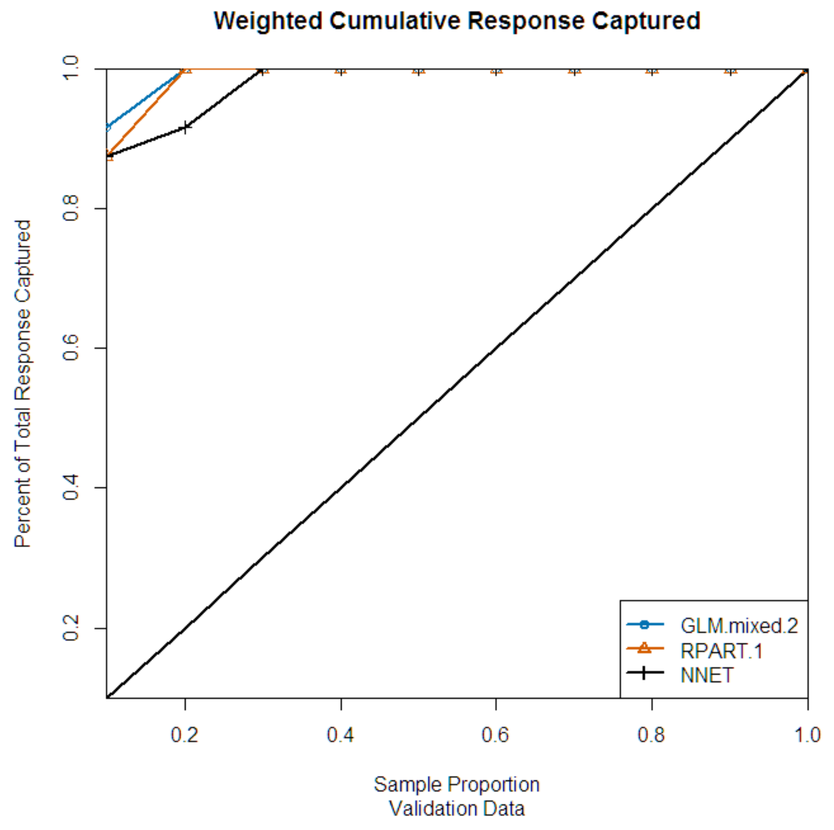


From 0.45 to 0.75, the neural net produces only 6 errors out of 341 observations, with no False Positives. By my metric, it is the best model so far.

Next, I'd like to test the best models on a lift chart, to see how they compare by traditional performance criteria. I originally chosen the mixed stepwise logistic regression model (GLM.mixed.2), the original decision tree (tree), the random forest with ntrees=100 and nodesize=5 (forest.final), and the neural network with 6 neurons (NNET.6).

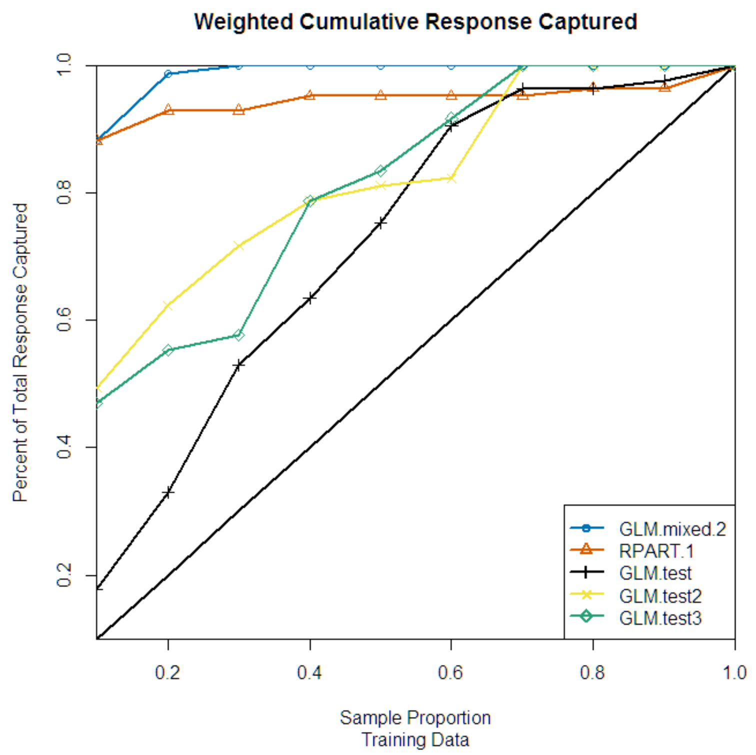
However, I discovered that I can't include tree or random forest packages on the lift charts. Instead, I used RPART.1 and left out the random forest model. First, I made a lift chart of the Training Dataset. The resulting chart shows that my mixed model and my neural net are "wizard" models. This was shocking at first, and I assumed that there must be some sort of error. Perhaps the models are all over-trained. But the performance on the validation lift chart tells me that the models really are this good.


Weighted Cumulative Response Captured
Percent of Total Response Captured
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0
GLM.mixed.2
RPART.1
NNET
Sample Proportion
Training Data


Weighted Cumulative Response Captured
Percent of Total Response Captured
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0
GLM.mixed.2
RPART.1
NNET
Sample Proportion
Validation Data

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project



With weaker models, the lift chart looks more normal. So, my final models are in fact very good. This is actually in line with my own error charts, which showed some models correctly predicting up to 98.5% of the observations in the validation set. Intuition would suggest that one of the variables is nearly identical to the active flag, but direct knowledge of the variables tells me that this is not the case. All of these variables are independent of activity.

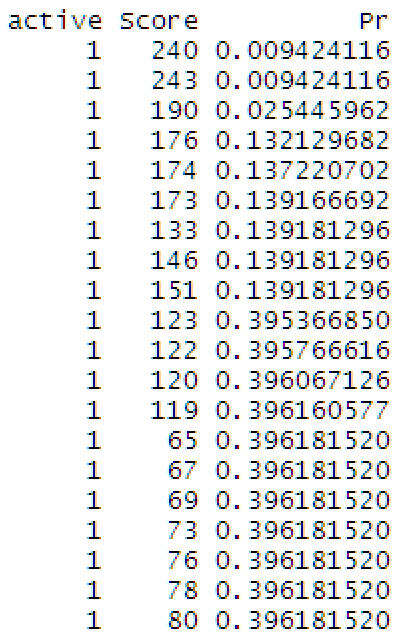
So, in the end my user base is extremely predictable, and my models are very useful. This is good news for my own use case. The next step is to observe the active users, and determine which ones are most likely to be inactive. These users should be given special attention, to hopefully prevent them from going inactive in the future.

To do this, I assign the “Probability of being active” to each observation, then sort by probability ascending and filter out the users who are already inactive. The result gives me a list of users, ranked in terms of their likelihood to be inactive given their data.

To do this, I assign the “Probability of being active” to each observation, then sort by probability ascending and filter out the users who are already inactive. The result gives me a list of users, ranked in terms of their likelihood to be inactive given their data.

Analysis of User Activity as of 08-05-2019

Big Data – Jared Crivello 83472 Final Project



```
active Score          Pr
     1   240 0.009424116
     1   243 0.009424116
     1   190 0.025445962
     1   176 0.132129682
     1   174 0.137220702
     1   173 0.139166692
     1   133 0.139181296
     1   146 0.139181296
     1   151 0.139181296
     1   123 0.395366850
     1   122 0.395766616
     1   120 0.396067126
     1   119 0.396160577
     1    65 0.396181520
     1    67 0.396181520
     1    69 0.396181520
     1    73 0.396181520
     1    76 0.396181520
     1    78 0.396181520
     1    80 0.396181520
```

The users' names and ids were not included in the report, to maintain their privacy. However, looking at the names I can definitely confirm that the list is accurate. Many of the names are users who are missing for long periods of time. Also on the list are "alt" accounts, which are second or third accounts that users only use in special circumstances.

However, the top 2 names on the list is are users that I took for granted. They are very active right now, but according to the model they are prime subjects for inactivity. I will have to watch them closely, and try to offer some sort of special attention in order to retain them.