Data Wrangling with the Tidyverse

Karen Mazidi

This notebook explores dplyr data wrangling with a phishing data set from the UCI repository

The data was converted from Weka format to an R data frame using this code:

```r
library(RWeka)
df <- read.arff("Training Dataset.arff")
```

Now using tbl_df(), convert the dataframe to a tibble. Examine the tibble with glimpse. Notice that all columns are factors. Many have two levels: 1 and -1, some have a 3rd level of 0. The documentation in the UCI ML link explains each column.

```r
library(tidyverse)
```

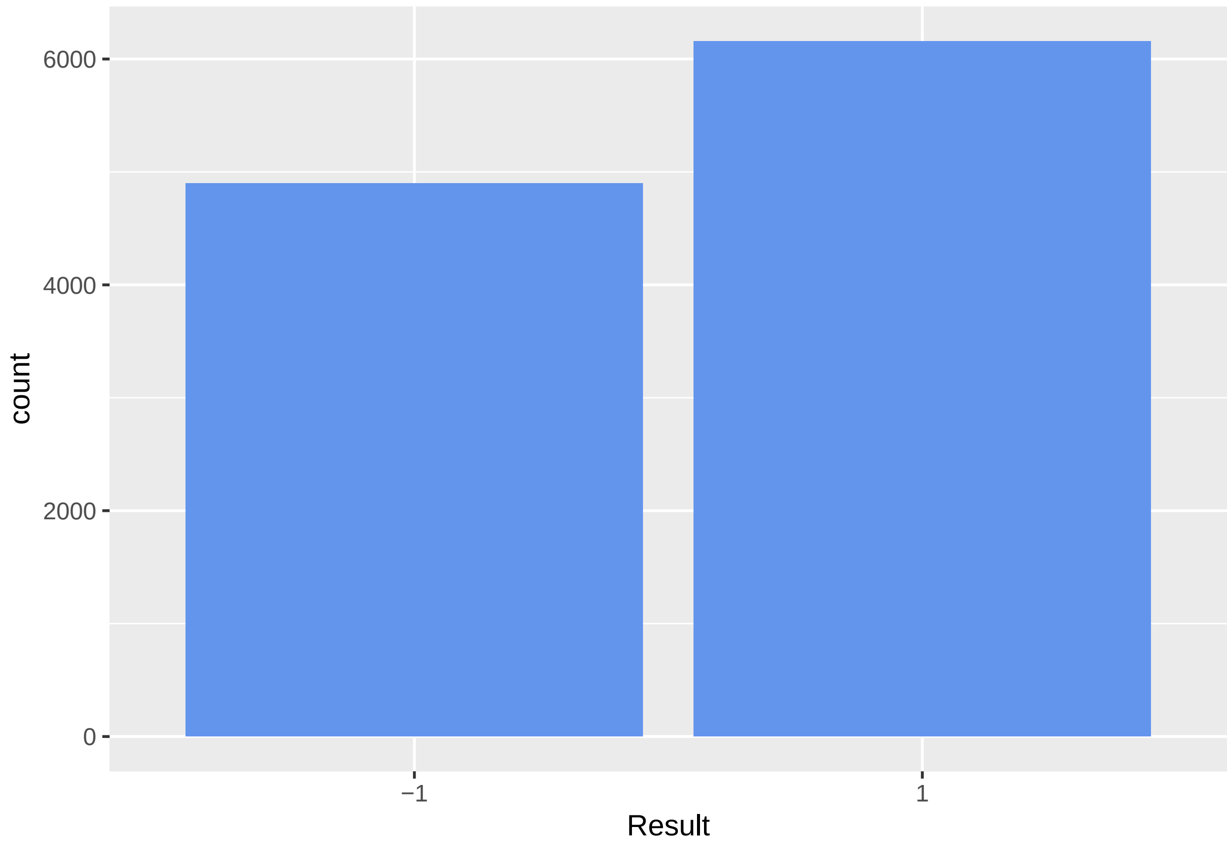
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
tb <- tbl_df(df)
glimpse(tb)
```

```
## Rows: 11,055
## Columns: 31
## $ having_IP_Address           <fct> -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -...
## $ URL_Length                  <fct> 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, ...
## $ Shortining_Service          <fct> 1, 1, 1, 1, -1, -1, -1, 1, -1, -1, 1, -...
## $ having_At_Symbol            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ double_slash_redirecting    <fct> -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -...
## $ Prefix_Suffix               <fct> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,...
## $ having_Sub_Domain           <fct> -1, 0, -1, -1, 1, 1, -1, -1, 1, -1, 0, ...
## $ SSLfinal_State              <fct> -1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, -...
## $ Domain_registeration_length <fct> -1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1,...
## $ Favicon                     <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ port                        <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ HTTPS_token                 <fct> -1, -1, -1, -1, 1, -1, 1, -1, -1, 1, 1,...
## $ Request_URL                 <fct> 1, 1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1,...
## $ URL_of_Anchor               <fct> -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, ...
## $ Links_in_tags               <fct> 1, -1, -1, 0, 0, 0, 0, -1, 1, 1, 0, -1,...
## $ SFH                         <fct> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,...
## $ Submitting_to_email         <fct> -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, -...
## $ Abnormal_URL                <fct> -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, -...
## $ Redirect                    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ on_mouseover                <fct> 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1...
## $ RightClick                  <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ popUpWidnow                 <fct> 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1...
## $ Iframe                      <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ age_of_domain               <fct> -1, -1, 1, -1, -1, 1, 1, -1, 1, 1, -1, ...
## $ DNSRecord                   <fct> -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, ...
## $ web_traffic                 <fct> -1, 0, 1, 1, 0, 1, -1, 0, 1, 0, 1, -1, ...
## $ Page_Rank                   <fct> -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, ...
## $ Google_Index                <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Links_pointing_to_page      <fct> 1, 1, 0, -1, 1, -1, 0, 0, 0, 0, -1, 0, ...
## $ Statistical_report          <fct> -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, -...
## $ Result                      <fct> -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1,...
```

Using a ggplot bar plot shows that the target column Result is failry evenly balanced.

```
ggplot(tb, aes(x=Result)) + geom_bar(fill="cornflowerblue")
```



### Train Test Split

A different way to split data. First create a row ID for each row, then sample 75% into train using sample_frac(), and put the rest in test using anti_join().

```
tb <- tb %>% mutate(id=row_number())
set.seed(1234)
train <- tb %>% sample_frac(.75)
test <- anti_join(tb, train, by='id')
```

Logistic regression on all predictors

The test accuracy is .93 using all 31 predictors. The mcc scores is .87. MCC returns a value between -1 and +1. The formula is:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MCC, or Matthews correlation coefficient, is used for binary classification. Some people like mcc because it will produce a high score only if all 4 categories TP, TN, FP, FN are good, and it's proportional to the number of true/false observations.

```r
library(mltools)
```

```
##
## Attaching package: 'mltools'

## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```r
glm1 <- glm(Result~.-id, data=train, family=binomial)
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, 2, 1)
acc1 <- mean(pred==as.integer(test$Result))
mcc1 <- mcc(pred, as.integer(test$Result))
```

Finding a simpler model

The model seems to be good but is hard to interpret with so many predictors. Let's search for a simpler model by first eliminating coefficients with low p-values. Extracting the p-values from summary() gives a named vector. This is subset to only those with p-values > .05

These columns will be removed.

```r
temp <- summary(glm1)$coefficients[,4]  # extract all 39 p-values
temp <- temp[temp>0.05]
temp
```

```
##               URL_Length0              URL_Length-1 double_slash_redirecting1
##                0.64962773                0.75624818                0.72837216
##            Prefix_Suffix1        having_Sub_Domain0                 Favicon-1
##                0.94155729                0.65165379                0.05694403
##                    port-1            Links_in_tags0      Submitting_to_email1
##                0.15335785                0.17869278                0.47025591
##             Abnormal_URL1            on_mouseover-1              RightClick-1
##                0.10461685                0.45107491                0.83603344
##             popUpWidnow-1                  Iframe-1            age_of_domain1
##                0.81135405                0.32954789                0.32811310
##                Page_Rank1
##                0.05985442
```

Next we use stringr's str_extract() and str_remove() functions to get rid of the dummy endings.

```r
delete_cols <- names(temp) %>% str_extract("\\w+") %>%
  str_remove_all("\\d$")
```

```
delete_cols
```

```
##  [1] "URL_Length"                "URL_Length"
##  [3] "double_slash_redirecting" "Prefix_Suffix"
##  [5] "having_Sub_Domain"        "Favicon"
##  [7] "port"                     "Links_in_tags"
##  [9] "Submitting_to_email"      "Abnormal_URL"
## [11] "on_mouseover"             "RightClick"
## [13] "popUpWidnow"              "Iframe"
## [15] "age_of_domain"            "Page_Rank"
```

Remove columns with high p-values

```
tb_reduced <- tb %>% select(-one_of(delete_cols))
glimpse(tb_reduced)
```

```
## Rows: 11,055
## Columns: 17
## $ having_IP_Address           <fct> -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, -...
## $ Shortining_Service          <fct> 1, 1, 1, 1, -1, -1, -1, 1, -1, -1, 1, -...
## $ having_At_Symbol            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ SSLfinal_State              <fct> -1, 1, -1, -1, 1, 1, -1, -1, 1, 1, 1, -...
## $ Domain_registeration_length <fct> -1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1,...
## $ HTTPS_token                 <fct> -1, -1, -1, -1, 1, -1, 1, -1, -1, 1, 1,...
## $ Request_URL                 <fct> 1, 1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1,...
## $ URL_of_Anchor               <fct> -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, ...
## $ SFH                         <fct> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,...
## $ Redirect                    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ DNSRecord                   <fct> -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, ...
## $ web_traffic                 <fct> -1, 0, 1, 1, 0, 1, -1, 0, 1, 0, 1, -1, ...
## $ Google_Index                <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Links_pointing_to_page      <fct> 1, 1, 0, -1, 1, -1, 0, 0, 0, 0, -1, 0, ...
## $ Statistical_report          <fct> -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, -...
## $ Result                      <fct> -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1,...
## $ id                          <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ...
```

```
train2 <- tb_reduced %>% sample_frac(.75)
test2 <- anti_join(tb_reduced, train2, by='id')
```

Run logistic regression on the reduced data set

The logistic regression model on the reduced data set is slightly less accurate but more interpretable with only 16 predictors.

```
glm2 <- glm(Result~.-id, data=train2, family=binomial)
probs <- predict(glm2, newdata=test2, type="response")
pred <- ifelse(probs>0.5, 2, 1)
acc2 <- mean(pred==as.integer(test2$Result))
mcc2 <- mcc(pred, as.integer(test2$Result))
summary(glm2)
```

```
##
## Call:
## glm(formula = Result ~ . - id, family = binomial, data = train2)
##
## Deviance Residuals:
```

```
##      Min        1Q    Median        3Q       Max
## -3.03209  -0.07374   0.08644   0.26472   2.67691
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -5.9162     0.3714 -15.929  < 2e-16 ***
## having_IP_Address1            1.4824     0.1376  10.773  < 2e-16 ***
## Shortining_Service-1          2.0712     0.2776   7.462 8.54e-14 ***
## having_At_Symbol-1           -0.3233     0.1614  -2.003 0.045149 *
## SSLfinal_State1               3.3892     0.1103  30.730  < 2e-16 ***
## SSLfinal_State0              -1.8264     0.2999  -6.089 1.13e-09 ***
## Domain_registeration_length1 -0.3965     0.1288  -3.077 0.002091 **
## HTTPS_token1                 -0.7482     0.2155  -3.471 0.000518 ***
## Request_URL-1                -0.6159     0.1211  -5.087 3.64e-07 ***
## URL_of_Anchor0                4.4289     0.2217  19.973  < 2e-16 ***
## URL_of_Anchor1                6.4458     0.2581  24.976  < 2e-16 ***
## SFH1                          1.3246     0.1632   8.114 4.88e-16 ***
## SFH0                          1.8128     0.2303   7.871 3.51e-15 ***
## Redirect1                    -1.3329     0.1871  -7.124 1.05e-12 ***
## DNSRecord1                    1.4164     0.1419   9.980  < 2e-16 ***
## web_traffic0                 -1.4147     0.1568  -9.021  < 2e-16 ***
## web_traffic1                  0.5764     0.1424   4.049 5.14e-05 ***
## Google_Index-1               -1.4275     0.1346 -10.602  < 2e-16 ***
## Links_pointing_to_page0      -1.6752     0.1355 -12.362  < 2e-16 ***
## Links_pointing_to_page-1     -0.9459     0.2419  -3.910 9.22e-05 ***
## Statistical_report1           0.3543     0.1675   2.115 0.034435 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11385.8  on 8290  degrees of freedom
## Residual deviance:  2943.7  on 8270  degrees of freedom
## AIC: 2985.7
##
## Number of Fisher Scoring iterations: 7
```

Group Features by Type

The documentation at the link above grouped the features into 4 general categories:

- address bar features
- abnormal based features
- html and JavaScript features
- domain based features

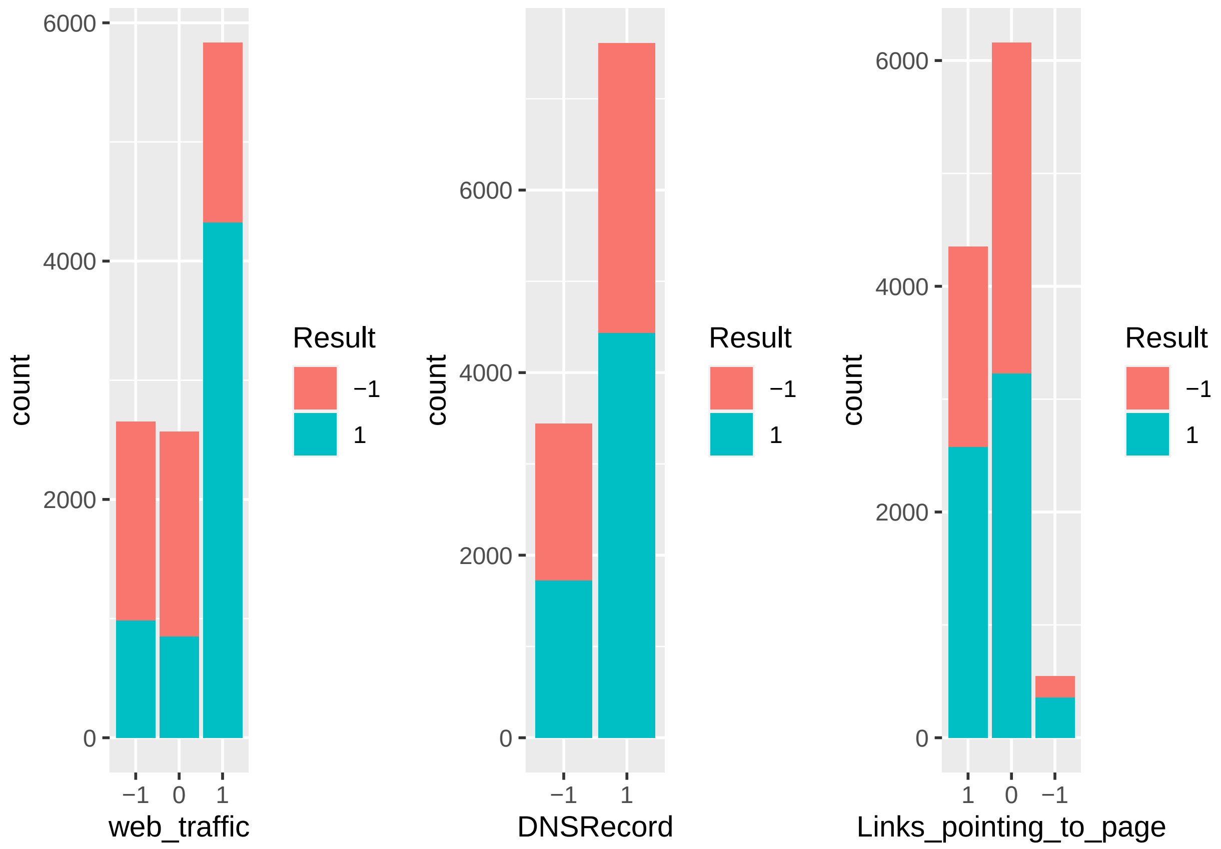
Looking at a few of the features in the domain based category does not show any strong relationship between the 2 or 3 levels of the feature and the target. Most of these got low p-values in the summary() done at the console, but their coefficients are small, indicating a minimal contribution to the log odds.

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
```

```
##
##     combine
```

```
p1 <- ggplot(tb_reduced, aes(x=web_traffic, fill=Result)) + geom_bar()
p2 <- ggplot(tb_reduced, aes(x=DNSRecord, fill=Result)) + geom_bar()
p3 <- ggplot(tb_reduced, aes(x=Links_pointing_to_page, fill=Result)) + geom_bar()
grid.arrange(p1, p2, p3, nrow=1)
```

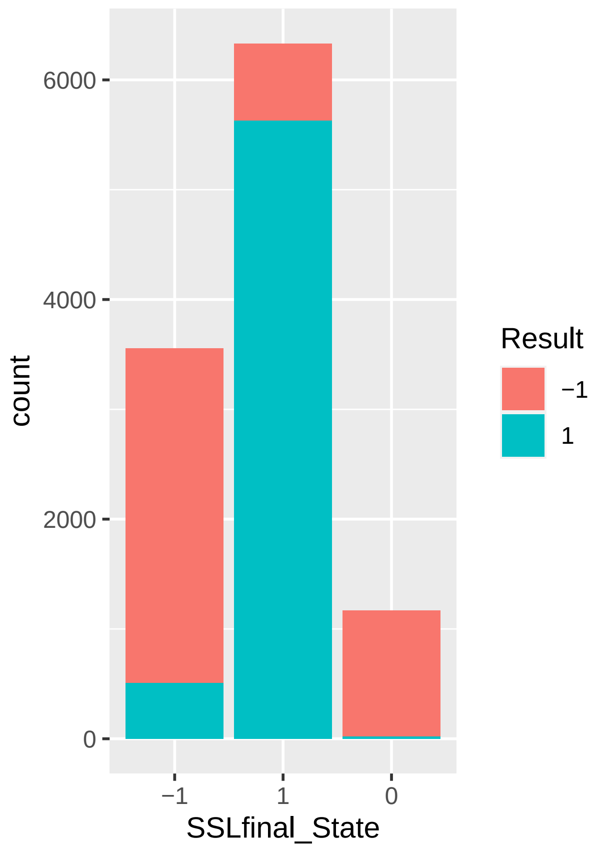


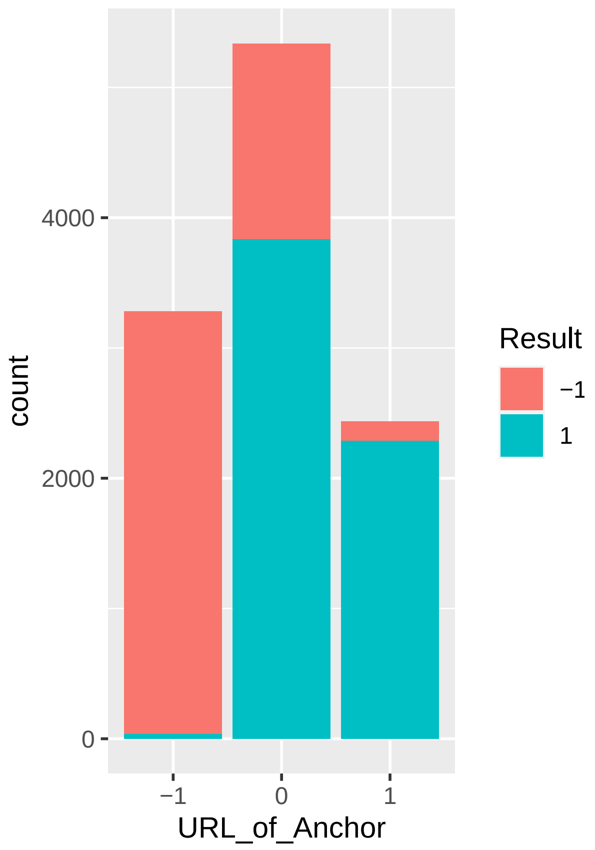
Two predictors with larger coefficients for the dummy factors include SSLfinal_State (looks for suspicious htyps protocols) and URL_of_Anchor (looks for funny things in tags). We can see the same thing in the graph:

- SSLfinal_State=1 has most of the observations as Result=1, most for levels 0 and -1 most of the observations have Result=-1
- URL_of_Anchor levels 0 and 1 have most observations as Result=1 while level -1 has almost all observations a Result=-1

This is an indication that there is predictive power in some of these levels.

```
p1 <- ggplot(tb_reduced, aes(x=SSLfinal_State, fill=Result)) + geom_bar()
p2 <- ggplot(tb_reduced, aes(x=URL_of_Anchor, fill=Result)) + geom_bar()
grid.arrange(p1, p2, nrow=1)
```





mutate and replace

The next code uses mutate and replace to:

- Make SSL final state binary by making 0 and -1 zero
- Make URL__of__Anchor binary by making 0 and 1 one and -1 zero (0 becomes 1 and -1 becomes 0)

```
tb3 <- mutate(tb_reduced, SSLfinal_State = replace(SSLfinal_State, which(SSLfinal_State == -1), 0))

tb3 <- mutate(tb3, URL_of_Anchor = replace(URL_of_Anchor, which(URL_of_Anchor==0), 1))
tb3 <- mutate(tb3, URL_of_Anchor = replace(URL_of_Anchor, which(URL_of_Anchor==-1), 0))
```

Another model

A third logistic regression model is used, this time only use the two variable that were just mutated. The accuracy is lower than either model, but is much more interpretable as you can see in the summary.

```
train3 <- tb3 %>% sample_frac(.75)
test3 <- anti_join(tb3, train3, by='id')

glm3 <- glm(Result~SSLfinal_State+URL_of_Anchor, data=train3, family=binomial)
probs <- predict(glm3, newdata=test3, type="response")
pred <- ifelse(probs>0.5, 2, 1)
acc3 <- mean(pred==as.integer(test3$Result))
mcc3 <- mcc(pred, as.integer(test3$Result))
summary(glm3)
```

```
##
## Call:
```

```
## glm(formula = Result ~ SSLfinal_State + URL_of_Anchor, family = binomial,
##     data = train3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.27825  -0.08464   0.39386   0.39386   2.20090
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -2.32905    0.20428  -11.40   <2e-16 ***
## SSLfinal_State0 -3.30112    0.08082  -40.84   <2e-16 ***
## URL_of_Anchor1   4.84669    0.20693   23.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11393.9  on 8290  degrees of freedom
## Residual deviance:  4243.5  on 8288  degrees of freedom
## AIC: 4249.5
##
## Number of Fisher Scoring iterations: 7
```