Poverty Data - Logistic Regression

Michael Bemus

2024-10-03

The last method we will use to assess the group structure of our data is logistic regression. Logistic regression is a classification method which relies on the logit function to assign probabilities of an observation being in a given class. In the case of multinomial regression, this method creates a model for each class other than the base class- which for our analysis will be the aid group. It then compares the probabilities of all outputs and concludes a given class.

Load Data

```
df <- read.csv("SubsidyClass/dataset.csv")
```

For this analysis, we finally get to leverage factor data in our model. While logistic regression still does transform variables with 3+ levels into dummy variables, the functions we use will do this automatically, simplifying our processes.

First, we need to drop the extra columns that we have not been using in our analysis.

```
df <- select(df, -X, -st, -off_pov, -spm_pov, -snap, -house_sub, -slunch,
             -energy, -wic)   # Remove unnecessary variables.
```

Next, we redefine our categorical variables as factors.

```
# Convert all character and binary columns to factors.
df$sex <- factor(df$sex)
df$mar <- factor(df$mar)
df$hispanic <- factor(df$hispanic)
df$race <-factor(df$race)
df$edu <- factor(df$edu)
df$ui_kids <- factor(df$ui_kids)
df$mortgage <- factor(df$mortgage)

# For group, we want to relevel to assure that it is the base class for our
# Logistic model.
df$group <- factor(df$group)
df$group <- relevel(df$group, "aid")
```

For our last preprocessing step, we need to break our data into training, testing, and validation samples. We will use the same exact sampling method as we did with the discriminant analysis model.

```r
set.seed(42)   # Assure that we get consistent samples when running the code.

# Create an index row to sample from.
df$sample <- 1:nrow(df)

s_vec <- c()   # Create a vector to hold sampled observation indexes in.

# For our analysis, we want to balance the percent of aid, poor, and no group
# observations there are relative to their proportions in the full data set.

for (i in c("aid", "poor", "no")) {
  # Here, for each class of Group, we sample 50% of the observations.

  s_vec <- c(s_vec,
             sample(df$sample[df$group == i],
                    round(length(df$sample[df$group == i])*0.5)))
}

# Label all sampled observations "train"
df$sample[as.integer(s_vec)] <- "train"


# Next, we repeat the above process to create a 30% Validation set.
# The main difference here is that we filter out values labeled train.
# From the remaining data, we will take 60% to make our validation set.

# Create an object that does not contain any training observations.
# For this object, we only need the group and sample columns to sample from.
no_train <- df |>
  filter(sample != "train") |>   # Remove all train values.
  select(group, sample)   # Select necessary columns.

s_vec <- c()   # Reset our sampling vector.

for (i in c("aid", "poor", "no")) {
  # Here, for each class of Group, we sample 60% of the observations.

  s_vec <- c(s_vec,
             sample(no_train$sample[no_train$group == i],
                    round(length(no_train$sample[no_train$group == i])*0.6)))
}

# Label all observations in the validation set.
df$sample[as.integer(s_vec)] <- "val"

# Everything else goes to Testing Set.
df$sample[(df$sample != "train")&(df$sample!="val")] <- "test"

# Clean up the variables we don't need anymore.
rm(s_vec)
rm(no_train)
```

```
# Print Train, Test, and Validation sizes.
print(paste("Train Size:", length(df$sample[df$sample == "train"])))
```

```
## [1] "Train Size: 1005488"
```

```
print(paste("Test Size:", length(df$sample[df$sample == "test"])))
```

```
## [1] "Test Size: 402195"
```

```
print(paste("Validation Size:", length(df$sample[df$sample == "val"])))
```

```
## [1] "Validation Size: 603292"
```

And now we'll divide our samples into separate data frames.

```
# Select training observations.
train_df <- df |>
  filter(sample == "train") |>
  select(-sample)

# Select validation observations.
val_df <- df |>
  filter(sample == "val") |>
  select(-sample)

# Select testing observations.
test_df <- df |>
  filter(sample == "test") |>
  select(-sample)

# We can now delete the full data set.
rm(df)
```

Training Logistic Model

Now that we have split our data, we can begin training our model.

```
# This function comes from the nnet package. Like in most logistic regressions,
# it uses gradient descent to find the optimal values for our data. However,
# because the training set is so large, it can take a very long time to fully
# converge. For this reason, we have set our max iterations to 250.

log_mod <- multinom(group~., data=train_df, maxit=250)
```

```
## # weights:  111 (72 variable)
## initial  value 1104641.472929
## iter  10 value 667174.672748
## iter  20 value 618606.428450
## iter  30 value 590721.348981
## iter  40 value 560620.423969
## iter  50 value 557265.484748
## iter  60 value 475577.366058
## iter  70 value 428484.841355
## iter  80 value 165307.078623
## iter  90 value 133194.204328
## iter 100 value 126923.784451
## iter 110 value 122546.318078
## iter 120 value 115792.518246
## iter 130 value 107555.443115
## iter 140 value 104502.294037
## iter 150 value 103367.482281
## iter 160 value 103018.856587
## iter 170 value 102998.639314
## iter 180 value 101999.531385
## iter 190 value 101945.527179
## iter 200 value 101726.459601
## iter 210 value 101567.416009
## iter 220 value 101398.101086
## final  value 101116.553895
## converged
```

The above output shows the deviance we have remaining in the model as we train it on further iterations. We see that, a bit after completing 220 iterations, our model converged with a total deviance of about 101,117. We see that, by the time we reached 180 iterations, we really were not seeing huge gains in the removal of deviance. It is good to see that the model converged within 250 iterations.

Validation for this model will be very important because we have only found a relative minimum in the possible deviance of our model. By running this same method on a different set of data, we can make sure that the coefficients we find are accurate.

```
# We want to store this because it takes a good minute to run.
train_sum <- summary(log_mod)
train_sum
```

```
## Call:
## multinom(formula = group ~ ., data = train_df, maxit = 250)
##
## Coefficients:
##      (Intercept)           age      marM     marNM        marS        marW
## no      0.401731 -0.0082754356 0.2580787 0.1439641  0.10747575  0.10183425
## poor    5.698871  0.0005135324 0.6095145 0.1283603 -0.03603539 -0.01865248
##             sexM      edu<HS         eduC       eduHS     raceB      raceO
## no    0.03567916 -0.07984779 -0.006107977  0.03549733  0.315485  0.2632768
## poor -0.12739856 -0.18338654  0.376193159 -0.11730774 -1.107038 -0.9529699
##          raceW  hispanic1           agi       hi_prem          moop    num_kid
## no    0.461462 -0.1462330  2.673518e-05 -9.440551e-06 -1.570904e-05 -0.9941733
## poor -1.022778  0.5112254 -5.391695e-05  2.919142e-05 -5.525348e-06  0.7637450
##       num_adlt  mortgageN  mortgageR       spm_res       spm_inc      fed_tax
## no   -1.447029  0.7173293 -0.1778419 -0.4088087471  4.089756e-01 -4.09035e-01
## poor  1.772905 -0.9215125 -0.1365145 -0.0002450146 -5.184531e-05 -3.71972e-05
##         fed_tax_bc          eitc          fica        st_tax     cap_xpen
## no   -0.0003841689 -5.533732e-04 -0.4090020370 -0.4088365851 -0.408872568
## poor  0.0009020602 -4.216156e-05  0.0002007454 -0.0003206812 -0.000125082
##           wk_xpen       cc_xpen   spm_hi_prem      med_xpen         mc_pb
## no   4.617532e-05 -1.760434e-04  2.514649e-05 -4.089571e-01  3.502176e-04
## poor 1.469230e-04  3.991626e-05 -3.560259e-05  5.051138e-05 -5.365148e-05
##        cohabit   ui_kids1
## no   -1.282583 -0.8703095
## poor  2.661153  0.5025736
##
## Std. Errors:
##       (Intercept)          age         marM        marNM         marS
## no   4.719793e-06 0.0004066909 3.284443e-07 6.759416e-08 1.967692e-07
## poor 3.849428e-06 0.0003358013 1.716187e-07 1.255734e-08 2.358347e-07
##              marW         sexM       edu<HS         eduC        eduHS
## no   2.299671e-06 1.572260e-06 1.130457e-06 3.159250e-07 2.153577e-06
## poor 1.604743e-06 1.233261e-06 1.321779e-06 3.672611e-07 1.361578e-06
##             raceB        raceO        raceW    hispanic1          agi
## no   6.860282e-07 2.452512e-07 3.801122e-06 7.606329e-08 6.252253e-07
## poor 9.388252e-07 3.580486e-07 2.498594e-06 2.273376e-07 5.748903e-07
##           hi_prem         moop      num_kid     num_adlt    mortgageN
## no   7.253945e-06 8.160850e-06 6.242063e-07 1.900966e-06 3.727081e-06
## poor 6.382471e-06 7.279084e-06 2.916261e-06 8.993249e-07 1.787410e-06
##         mortgageR      spm_res      spm_inc      fed_tax   fed_tax_bc
## no   7.603618e-07 2.956848e-05 2.953534e-05 3.090481e-05 1.269422e-05
## poor 2.266768e-06 2.122565e-06 2.385274e-06 6.054866e-06 8.380679e-06
##              eitc         fica       st_tax     cap_xpen      wk_xpen
## no   2.697655e-05 3.101728e-05 3.219879e-05 0.0001765178 0.0002064398
## poor 1.602069e-05 9.772246e-06 1.367678e-05 0.0001276295 0.0001285087
##           cc_xpen  spm_hi_prem     med_xpen        mc_pb      cohabit
## no   0.0001949081 8.478432e-06 2.979688e-05 1.535341e-05 6.581809e-07
## poor 0.0001206237 7.328235e-06 4.871764e-06 1.403319e-05 5.644966e-07
##          ui_kids1
## no   2.046145e-08
## poor 4.028907e-08
##
```

```
## Residual Deviance: 202233.1
## AIC: 202377.1
```

Similarly to our discriminant models, we see two sets of coefficients, one for the no group and one for the poor group. These coefficients are found in a linear regression function within the logit function itself. Positive coefficients correspond with a stronger positive correlation to the model's corresponding group. Negative coefficients correspond with a stronger positive correlation to the aid group.

To find the relative effect of these coefficients, we must take the exponential of each.

```
# Take the exponential of all of our coefficients.
t(exp(train_sum$coefficients))
```

```
##                    no        poor
## (Intercept) 1.4944093 298.5301468
## age         0.9917587   1.0005137
## marM        1.2944407   1.8395380
## marNM       1.1548426   1.1369625
## marS        1.1134639   0.9646062
## marW        1.1071999   0.9815204
## sexM        1.0363233   0.8803827
## edu<HS      0.9232569   0.8324463
## eduC        0.9939106   1.4567285
## eduHS       1.0361349   0.8893115
## raceB       1.3709241   0.3305367
## raceO       1.3011868   0.3855941
## raceW       1.5863917   0.3595945
## hispanic1   0.8639564   1.6673331
## agi         1.0000267   0.9999461
## hi_prem     0.9999906   1.0000292
## moop        0.9999843   0.9999945
## num_kid     0.3700292   2.1462992
## num_adlt    0.2352682   5.8879348
## mortgageN   2.0489538   0.3979167
## mortgageR   0.8370747   0.8723937
## spm_res     0.6644413   0.9997550
## spm_inc     1.5052749   0.9999482
## fed_tax     0.6642910   0.9999628
## fed_tax_bc  0.9996159   1.0009025
## eitc        0.9994468   0.9999578
## fica        0.6643129   1.0002008
## st_tax      0.6644228   0.9996794
## cap_xpen    0.6643989   0.9998749
## wk_xpen     1.0000462   1.0001469
## cc_xpen     0.9998240   1.0000399
## spm_hi_prem 1.0000251   0.9999644
## med_xpen    0.6643428   1.0000505
## mc_pb       1.0003503   0.9999463
## cohabit     0.2773200  14.3127839
## ui_kids1    0.4188219   1.6529699
```

Generally, values greater than 1 suggest that the variable is more positively correlated with the corresponding category than it is to the aid group. If it is less than 1, the variable is more positively correlated to the aid group.

The relative effect of these values are multipliers to the value of a coefficient. So, for example, for each 1 point of age, the model is 0.9917587 times less likely to be of the no group and 1.0005137 times more likely to be of the poor group, both compared to the aid group.

With these coefficients, we can see a few interesting trends. First, this model implies that the aid group is the least likely to be married, which seems strange due to the fact that we believe the aid group to generally represent families with children in school. This model also implies that the poor group is the most likely to have children, which is the opposite of what we were seeing in our visualizations.

Many of our numeric variables have very small effects. This is because their scale is so much greater than the scale of the other variables. If we were to normalize their values, we might see somewhat different results.

We see that this model implies that the aid group has the highest state tax, total expenses, out of pocket medical costs, and total resources.

For many of our financial variables, it appears as though the poor group is most similar to the aid group, but there could very well be multicolinearity playing into those estimates.

What might be useful would be seeing which variables are significant to this model. To determine this, we must compute p-values as follows.

```
mlog_pvals <- function(log_sum) {
  # We want to input the log-model summary into this function.
  # First, we normalize our coefficients by dividing them by their standard
  # errors.
  # Then, we compute a normal probability of whether the value is non-zero.

  z <- log_sum$coefficients/log_sum$standard.errors
  p <- (1 - pnorm(abs(z), 0, 1)) * 2
  return(p)
}

# Run the above function on our model.
mlog_pvals(train_sum)
```

```
##      (Intercept)       age marM marNM marS marW sexM edu<HS eduC eduHS raceB
## no             0 0.0000000    0     0    0    0    0      0    0     0     0
## poor           0 0.1261964    0     0    0    0    0      0    0     0     0
##      raceO raceW hispanic1 agi      hi_prem       moop num_kid num_adlt
## no       0     0         0   0 1.931090e-01 0.05423842       0        0
## poor     0     0         0   0 4.792189e-06 0.44780961       0        0
##      mortgageN mortgageR spm_res spm_inc      fed_tax fed_tax_bc        eitc
## no           0         0       0       0 0.000000e+00          0 0.000000000
## poor         0         0       0       0 8.079561e-10          0 0.008496035
##      fica st_tax  cap_xpen   wk_xpen   cc_xpen  spm_hi_prem med_xpen
## no      0      0 0.0000000 0.8230106 0.3664133 3.017633e-03        0
## poor    0      0 0.3270663 0.2529175 0.7407082 1.184125e-06        0
##             mc_pb cohabit ui_kids1
## no   0.0000000000       0        0
## poor 0.0001317393       0        0
```

In the above, anything with value greater than 0.05 is not significant. We see that many of our values 0 out because their p-values are so small. In the end, we see that only wk_xpen and cc_xpen are insignificant to both models. For our sub=models, we see, hi_prem is insignificant to the no group model. Age, MOOP, and Cap_Xpen are also insignificant for the poor group model.

Because the poor group has more insignificant variables, it may suggest that the poor group is more similar to the aid group. However, we will rely on our cross validation to make any final conclusions.

Validating Logistic Model

Before we run our final tests, we want to see whether we get similar coefficients using the validation set to train the data. This model should hopefully be faster to train because it has less data.

```
val_mod <- multinom(group~., data=val_df, maxit=250)
```

```
## # weights:  111 (72 variable)
## initial  value 662784.004859
## iter  10 value 334158.819736
## iter  20 value 311531.727143
## iter  30 value 291026.755833
## iter  40 value 262429.026560
## iter  50 value 260527.875274
## iter  60 value 229099.829515
## iter  70 value 214486.601431
## iter  80 value 103302.169427
## iter  90 value 81885.579196
## iter 100 value 78141.734011
## iter 110 value 69628.073473
## iter 120 value 64090.789884
## iter 130 value 61929.992024
## iter 140 value 61140.964364
## iter 150 value 60904.593480
## iter 160 value 60821.034120
## iter 170 value 60810.521212
## iter 180 value 60582.304554
## iter 190 value 60566.430659
## iter 200 value 60559.751166
## iter 210 value 60532.344924
## iter 220 value 60519.775571
## final  value 60517.923387
## converged
```

We see that our data starts with almost half the deviance of the training set, which is reasonable due to the fact that it has less data. However, due to the number of variables in our model, it still took about the same number of iterations to find a convergence point in the deviance.

```
# Get the coefficients of the validation model.
val_sum <- summary(val_mod)
val_sum
```

```
## Call:
## multinom(formula = group ~ ., data = val_df, maxit = 250)
##
## Coefficients:
##      (Intercept)           age      marM      marNM       marS        marW
## no     0.5292405 -0.0092061004 0.2555461 0.07295392 0.10270053  0.04587191
## poor   5.8697502  0.0005723987 0.6387521 0.18154086 0.09519392 -0.03124772
##             sexM      edu<HS      eduC       eduHS      raceB      raceO
## no    0.02829075 -0.03319879 0.0206171  0.08893832  0.2902887  0.3662651
## poor -0.14981913 -0.18085228 0.3797387 -0.12331715 -1.1211378 -1.0109442
##          raceW  hispanic1           agi       hi_prem          moop    num_kid
## no    0.490010 -0.2344679  2.633115e-05 -5.854402e-06 -1.484015e-05 -0.9198091
## poor -1.057238  0.5310431 -5.468758e-05  2.330667e-05 -9.833747e-06  0.7566142
##       num_adlt  mortgageN  mortgageR       spm_res       spm_inc       fed_tax
## no   -1.441506  0.6900199 -0.1910441 -0.4168200052  4.169835e-01 -4.170188e-01
## poor  1.791310 -0.9591887 -0.2058942 -0.0002494984 -5.208144e-05 -5.015852e-05
##         fed_tax_bc          eitc          fica        st_tax      cap_xpen
## no   -0.0004018061 -5.044841e-04 -0.4170311137 -0.4168213329 -0.4166971273
## poor  0.0009369259 -4.929171e-05  0.0002049047 -0.0003468607 -0.0001014424
##            wk_xpen       cc_xpen   spm_hi_prem      med_xpen         mc_pb
## no   -1.086121e-04 -3.105389e-04  1.263964e-05 -0.4169662349  3.490852e-04
## poor  9.205678e-05  3.072087e-05 -3.871826e-05  0.0000551332 -6.124504e-05
##        cohabit   ui_kids1
## no   -1.208209 -1.0602867
## poor  2.638100  0.3804082
##
## Std. Errors:
##       (Intercept)           age         marM        marNM         marS
## no   6.066966e-06 0.0005249614 4.308794e-07 1.616467e-07 2.660438e-07
## poor 4.988275e-06 0.0004385979 1.845957e-07 1.103987e-07 2.937782e-07
##              marW         sexM       edu<HS         eduC        eduHS
## no   3.043542e-06 1.881599e-06 1.554876e-06 3.283640e-07 2.769399e-06
## poor 2.181024e-06 1.646847e-06 1.703495e-06 4.133177e-07 1.767398e-06
##             raceB        raceO        raceW    hispanic1          agi
## no   8.402666e-07 3.543604e-07 4.879811e-06 1.041279e-07 7.954157e-07
## poor 1.206681e-06 3.931288e-07 3.334248e-06 2.940711e-07 7.416604e-07
##           hi_prem         moop      num_kid     num_adlt    mortgageN
## no   9.252853e-06 1.060294e-05 7.526729e-07 2.424601e-06 4.776802e-06
## poor 8.224532e-06 9.536707e-06 3.779991e-06 1.115060e-06 2.320475e-06
##         mortgageR      spm_res      spm_inc      fed_tax   fed_tax_bc
## no   1.005867e-06 2.691541e-05 2.686145e-05 2.914792e-05 1.657573e-05
## poor 3.003816e-06 2.787039e-06 3.136915e-06 7.985857e-06 1.086391e-05
##              eitc         fica       st_tax     cap_xpen      wk_xpen
## no   3.514693e-05 2.953073e-05 3.142805e-05 0.0001595838 0.0001871805
## poor 2.105735e-05 1.292832e-05 1.726919e-05 0.0001514539 0.0001525975
##           cc_xpen  spm_hi_prem     med_xpen        mc_pb      cohabit
## no   0.0001702505 1.087696e-05 2.735544e-05 1.984753e-05 8.123017e-07
## poor 0.0001425715 9.504655e-06 6.447697e-06 1.824345e-05 7.842107e-07
##          ui_kids1
## no   2.356880e-08
## poor 3.983048e-08
##
```

```
## Residual Deviance: 121035.8
## AIC: 121179.8
```

We have a lot of coefficients, so in this format, it is hard to compare our models. Placing them side-by-side…

```
comp_out <- t(rbind(train_sum$coefficients[1,], val_sum$coefficients[1,],
                    train_sum$coefficients[2,], train_sum$coefficients[2,]))
colnames(comp_out) <- c("no-Train", "no-Val", "poor-Train", "poor-Val")
comp_out
```

```
##                 no-Train        no-Val    poor-Train      poor-Val
## (Intercept)  4.017310e-01  5.292405e-01  5.698871e+00  5.698871e+00
## age         -8.275436e-03 -9.206100e-03  5.135324e-04  5.135324e-04
## marM         2.580787e-01  2.555461e-01  6.095145e-01  6.095145e-01
## marNM        1.439641e-01  7.295392e-02  1.283603e-01  1.283603e-01
## marS         1.074757e-01  1.027005e-01 -3.603539e-02 -3.603539e-02
## marW         1.018343e-01  4.587191e-02 -1.865248e-02 -1.865248e-02
## sexM         3.567916e-02  2.829075e-02 -1.273986e-01 -1.273986e-01
## edu<HS      -7.984779e-02 -3.319879e-02 -1.833865e-01 -1.833865e-01
## eduC        -6.107977e-03  2.061710e-02  3.761932e-01  3.761932e-01
## eduHS        3.549733e-02  8.893832e-02 -1.173077e-01 -1.173077e-01
## raceB        3.154850e-01  2.902887e-01 -1.107038e+00 -1.107038e+00
## raceO        2.632768e-01  3.662651e-01 -9.529699e-01 -9.529699e-01
## raceW        4.614620e-01  4.900100e-01 -1.022778e+00 -1.022778e+00
## hispanic1   -1.462330e-01 -2.344679e-01  5.112254e-01  5.112254e-01
## agi          2.673518e-05  2.633115e-05 -5.391695e-05 -5.391695e-05
## hi_prem     -9.440551e-06 -5.854402e-06  2.919142e-05  2.919142e-05
## moop        -1.570904e-05 -1.484015e-05 -5.525348e-06 -5.525348e-06
## num_kid     -9.941733e-01 -9.198091e-01  7.637450e-01  7.637450e-01
## num_adlt    -1.447029e+00 -1.441506e+00  1.772905e+00  1.772905e+00
## mortgageN    7.173293e-01  6.900199e-01 -9.215125e-01 -9.215125e-01
## mortgageR   -1.778419e-01 -1.910441e-01 -1.365145e-01 -1.365145e-01
## spm_res     -4.088087e-01 -4.168200e-01 -2.450146e-04 -2.450146e-04
## spm_inc      4.089756e-01  4.169835e-01 -5.184531e-05 -5.184531e-05
## fed_tax     -4.090350e-01 -4.170188e-01 -3.719720e-05 -3.719720e-05
## fed_tax_bc  -3.841689e-04 -4.018061e-04  9.020602e-04  9.020602e-04
## eitc        -5.533732e-04 -5.044841e-04 -4.216156e-05 -4.216156e-05
## fica        -4.090020e-01 -4.170311e-01  2.007454e-04  2.007454e-04
## st_tax      -4.088366e-01 -4.168213e-01 -3.206812e-04 -3.206812e-04
## cap_xpen    -4.088726e-01 -4.166971e-01 -1.250820e-04 -1.250820e-04
## wk_xpen      4.617532e-05 -1.086121e-04  1.469230e-04  1.469230e-04
## cc_xpen     -1.760434e-04 -3.105389e-04  3.991626e-05  3.991626e-05
## spm_hi_prem  2.514649e-05  1.263964e-05 -3.560259e-05 -3.560259e-05
## med_xpen    -4.089571e-01 -4.169662e-01  5.051138e-05  5.051138e-05
## mc_pb        3.502176e-04  3.490852e-04 -5.365148e-05 -5.365148e-05
## cohabit     -1.282583e+00 -1.208209e+00  2.661153e+00  2.661153e+00
## ui_kids1    -8.703095e-01 -1.060287e+00  5.025736e-01  5.025736e-01
```

Comparing the models in this way, we do not see much difference between the coefficients of the poor group.

However, the coefficients for the no group are somewhat different. We see sign changes for wk_xpen, which we know was insignificant and therefore not too troublesome, and education-College, which was significant to the model. We also see noticeable changes of magnitude in ui_kids1, spm_hi_prem, cc_xpen, hi_prem, hispanic, raceO, eduHS, edu<HS, marW, marNM, and age.

Because we have seen more changes in the no group, this might suggest that the discernible differences between it and the aid group are weaker, which would imply they are more similar. However, we will still rely on our cross validation to come to a final conclusion.

Testing Model

Now, we will see how accurately we can predict the test data, and how misclassification typically occurs.

```
# Create predictions of the Groups in our test data.
log_pred <- predict(log_mod, test_df)

# Compute a confusion matrix to show our results.
log_conf <- table(test_df$group, log_pred)
log_conf
```

```
##       log_pred
##           aid     no   poor
##   aid   67629    103   4915
##   no        1 270957   2265
##   poor   1413   5329  49583
```

These results are very interesting. Generally, the accuracy of this model seems pretty good. We have only misclassified 14,026 out of our 402,195 observations, which gives us an accuracy above 95%. It is a fairly impressive result.

More interesting, though, is that the aid group and the no group did not misclassify into each other very frequently. Both misclassified into the poor group much more frequently, with the no group only misclassifying into the aid group once. According to the logic we have been using, this would suggest there is a very fine line between the aid and the no groups.

For the poor group, we see that it misclassifies more frequently into the no group. This could be a result of the sample size of the no group biasing which group should be the most frequent. We see that the poor group is classified the worst out of our classes, but the result is still better than what we saw with our discriminant analysis.

```
# Calculate the accuracy of our class predictions.
print(paste("Accuracy:", mean(test_df$group == log_pred)))
```

```
## [1] "Accuracy: 0.965126369049839"
```

```
# Calculate the recall and precision related to the Aid Group.
recl_log <- log_conf[1, 1]/(log_conf[1,1] + log_conf[1,2] + log_conf[1,3])
prci_log <- log_conf[1, 1]/(log_conf[1,1] + log_conf[2,1] + log_conf[3,1])

# Print out all metrics.
print(paste("Recall:", recl_log))
```

```
## [1] "Recall: 0.93092625985932"
```

```
print(paste("Precision:", prci_log))
```

```
## [1] "Precision: 0.979520009269586"
```

```
print(paste("F1:", 2*recl_log*prci_log/(recl_log + prci_log)))
```

```
## [1] "F1: 0.954605123861952"
```

Our test scores show that we could be doing somewhat better predicting the aid group itself, but we are not misclassifying into it very frequently at all. The fact that our F1 score is greater than 95% is very promising for us in terms of the applicability of this predictive model.

Generally, the results of this model have thrown somewhat of a curveball into our overall analysis. The PCA method seemed to suggest the aid group was more similar to the no group, and the Discriminant Method gave us somewhat muddled results as far as the mutual relationships between the groups were concerned. Now, this model has swung all the way to the other side, seeming to show that the aid group is most similar to the poor group.

We see that these predictive models are perhaps not the best for determining whether a certain group should exist through this direct classification method. However, there are certainly other methods we could try, such as removing classes to see how the models change or rebalancing the sample sizes of each of our groups.