Direct Mail Fundraising Project

Rudy Duran

8/5/2020

Objective

The purpose of this project is to build a classification model in order to improve the cost effectiveness of a national veterans organization's direct marketing campaign. This model will help by predicting which individuals will be more likely to donate to the organization as opposed to donors who will not donate.

Data Sources and Data Used

The original dataset being used will be the fundraising dataset which contains 3000 observations and a total of 21 variables. The target variable will be the dependent variable being used for this model.

The future_fundraising dataset contains 120 observations with 20 variables. The target variable is not included in the future_fundraising dataset because this dataset will be used to predict which individuals are more likely to donate to the campaign.

Below are the following libraries which were used in exploring, analyzing, and building this model:

The fundraising dataset is loaded into R below:

```
fundraising <- readRDS("fundraising.rds")
```

The future_fundraising dataset is loaded into R below:

```
future_fundraising <- readRDS("future_fundraising.rds")
```

I created a new variable named fundraisingnew from fundraising:

```
fundraisingnew <- fundraising
```

```
summary(fundraising)
```

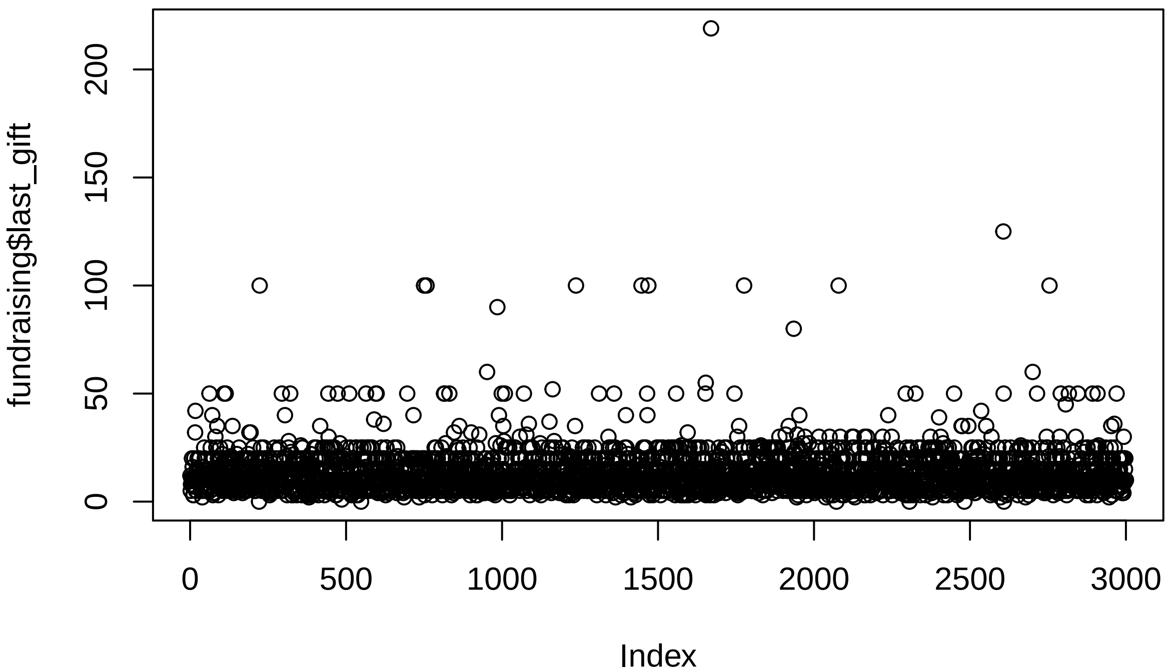
```
##  zipconvert2 zipconvert3 zipconvert4 zipconvert5 homeowner    num_child
##  No :2352    Yes: 551    No :2357    No :1846    Yes:2312   Min.   :1.000
##  Yes: 648    No :2449    Yes: 643    Yes:1154    No : 688   1st Qu.:1.000
##                                                             Median :1.000
##                                                             Mean   :1.069
##                                                             3rd Qu.:1.000
##                                                             Max.   :5.000
##      income      female         wealth        home_value      med_fam_inc
```

```
##  Min.   :1.000   Yes:1831   Min.   :0.000   Min.   :   0.0   Min.   :   0.0
##  1st Qu.:3.000   No :1169   1st Qu.:5.000   1st Qu.: 554.8   1st Qu.: 278.0
##  Median :4.000              Median :8.000   Median : 816.5   Median : 355.0
##  Mean   :3.899              Mean   :6.396   Mean   :1143.3   Mean   : 388.4
##  3rd Qu.:5.000              3rd Qu.:8.000   3rd Qu.:1341.2   3rd Qu.: 465.0
##  Max.   :7.000              Max.   :9.000   Max.   :5945.0   Max.   :1500.0
##   avg_fam_inc      pct_lt15k        num_prom      lifetime_gifts
##  Min.   :   0.0   Min.   : 0.00   Min.   : 11.00   Min.   :  15.0
##  1st Qu.: 318.0   1st Qu.: 5.00   1st Qu.: 29.00   1st Qu.:  45.0
##  Median : 396.0   Median :12.00   Median : 48.00   Median :  81.0
##  Mean   : 432.3   Mean   :14.71   Mean   : 49.14   Mean   : 110.7
##  3rd Qu.: 516.0   3rd Qu.:21.00   3rd Qu.: 65.00   3rd Qu.: 135.0
##  Max.   :1331.0   Max.   :90.00   Max.   :157.00   Max.   :5674.9
##   largest_gift       last_gift      months_since_donate    time_lag
##  Min.   :   5.00   Min.   :  0.00   Min.   :17.00       Min.   : 0.000
##  1st Qu.:  10.00   1st Qu.:  7.00   1st Qu.:29.00       1st Qu.: 3.000
##  Median :  15.00   Median : 10.00   Median :31.00       Median : 5.000
##  Mean   :  16.65   Mean   : 13.48   Mean   :31.13       Mean   : 6.876
##  3rd Qu.:  20.00   3rd Qu.: 16.00   3rd Qu.:34.00       3rd Qu.: 9.000
##  Max.   :1000.00   Max.   :219.00   Max.   :37.00       Max.   :77.000
##     avg_gift           target
##  Min.   :  2.139   Donor   :1499
##  1st Qu.:  6.333   No Donor:1501
##  Median :  9.000
##  Mean   : 10.669
##  3rd Qu.: 12.800
##  Max.   :122.167
```

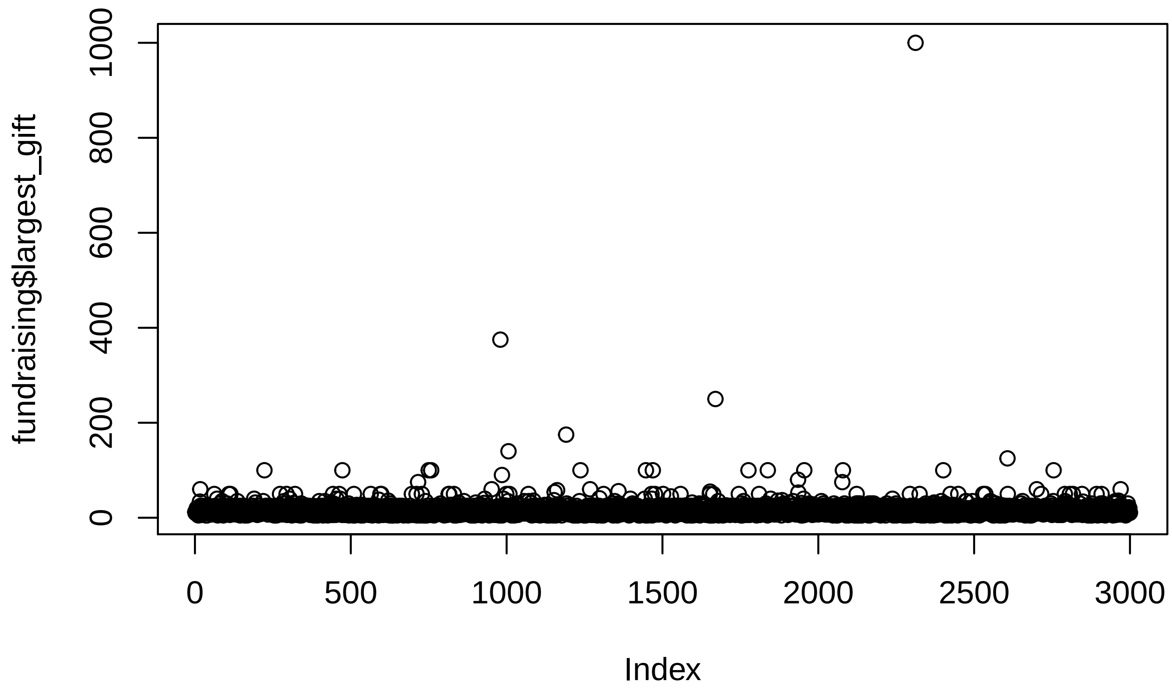
Above is a summary of the dataset. There are 7 factor variables including the target variable and about 14 numeric variables. In terms of the target variable, there is about an even split with 1499 observations categorized as "Donor" and 1501 categorized as "No Donor".

Exploratory Data Analysis

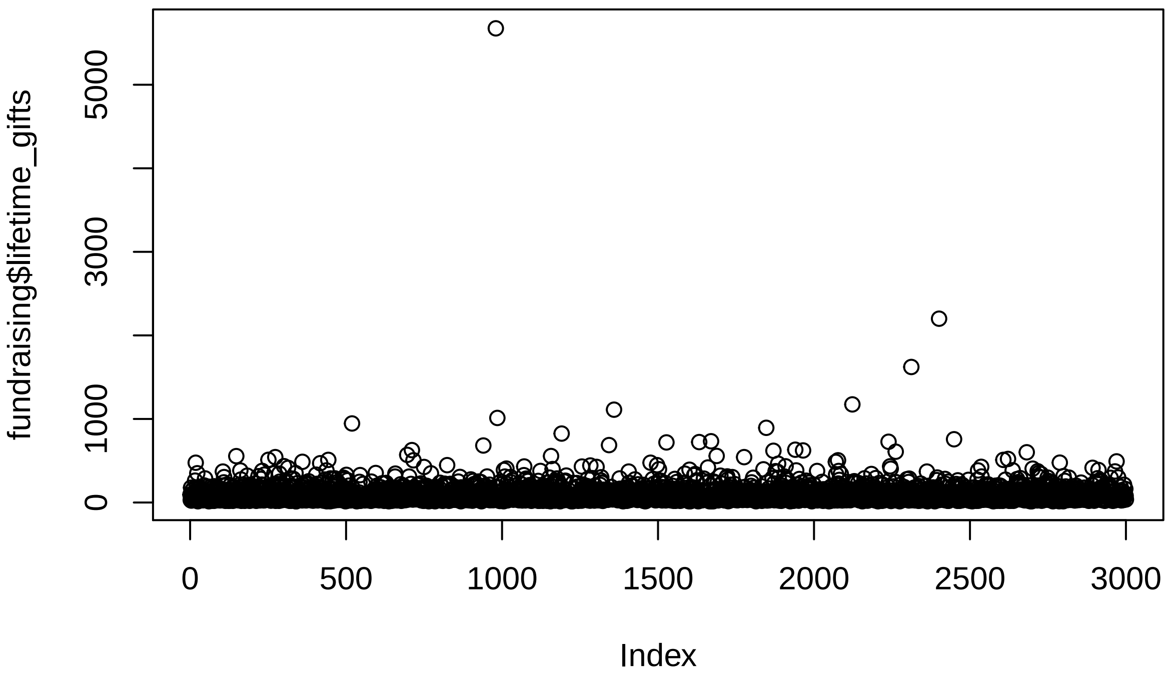
```
plot(fundraising$last_gift)
```




```
plot(fundraising$largest_gift)
```

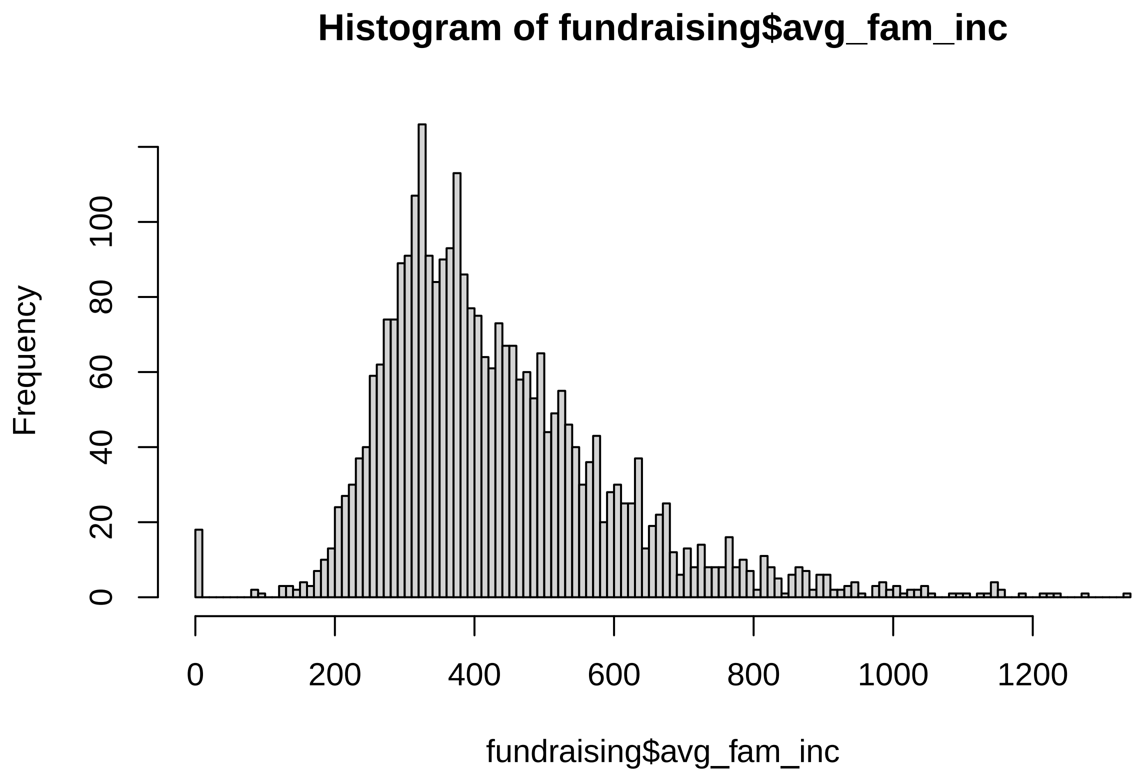



```
plot(fundraising$lifetime_gifts)
```

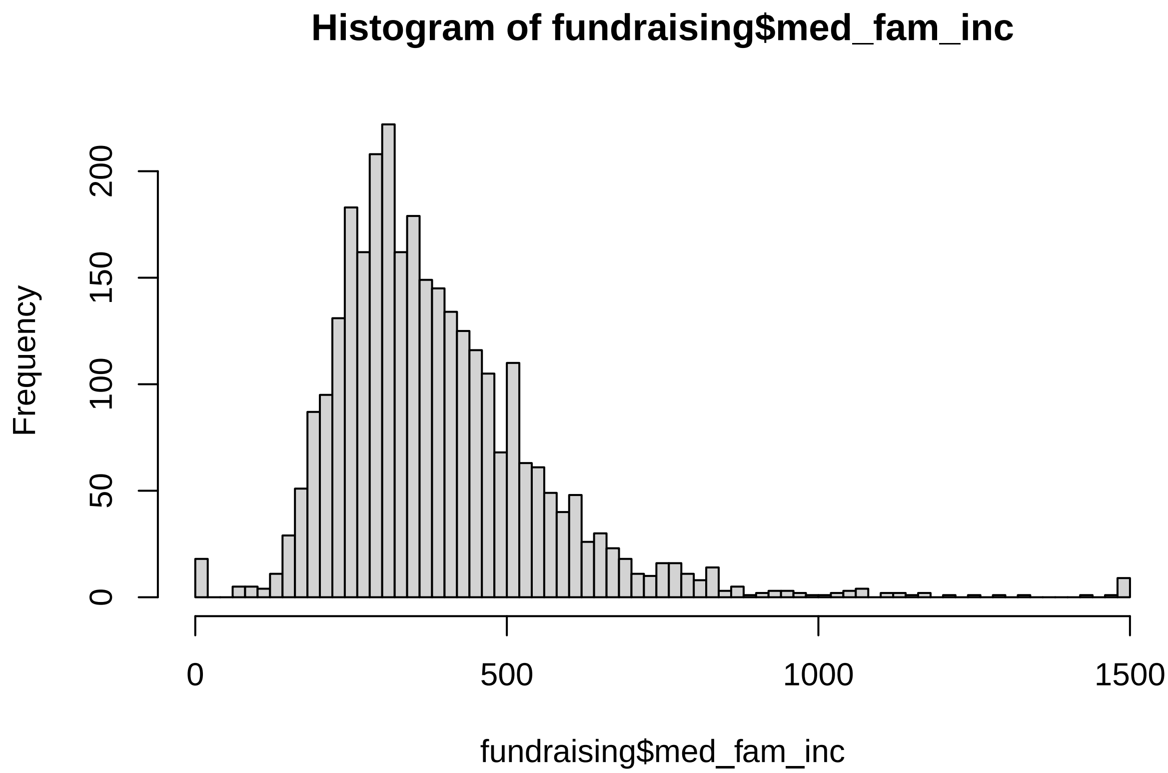


I plotted a number of histograms above in order to understand the data. You'll notice that last_gift has an outlier above 200, largest_gift has an outlier around 1000, and lifetime_gifts has an outlier above 5000.

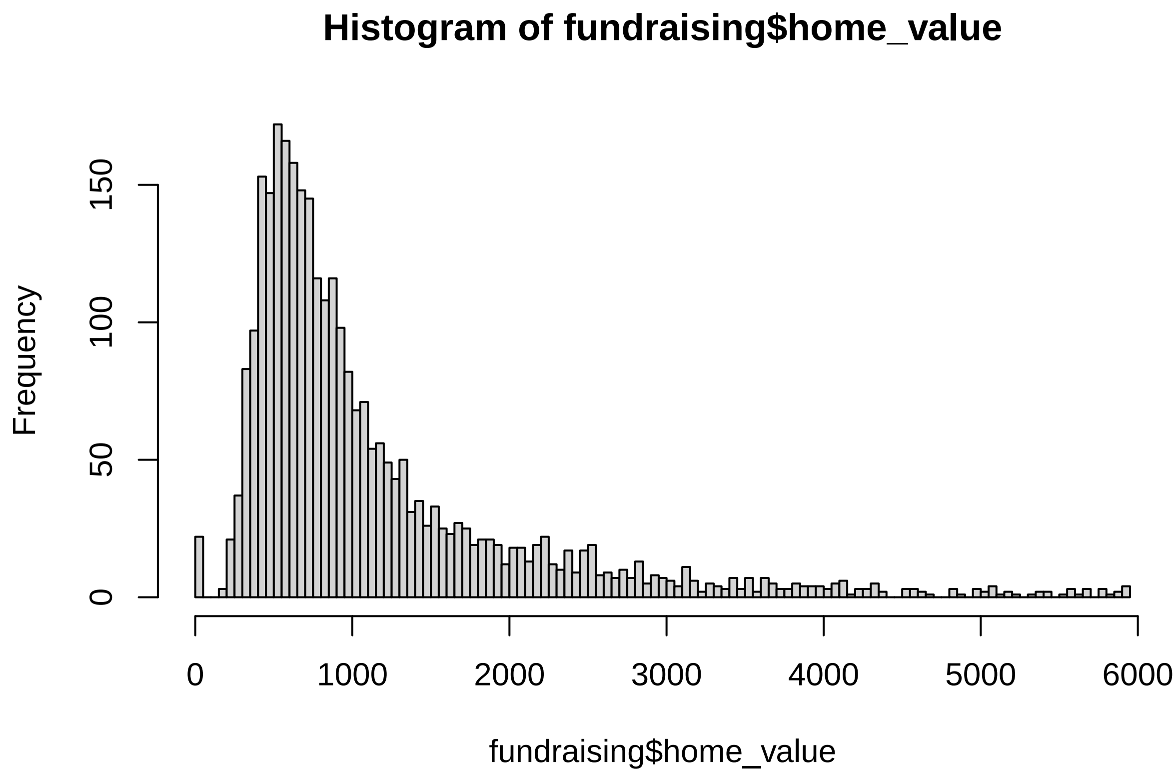
```
hist(fundraising$avg_fam_inc, breaks = 100)
```



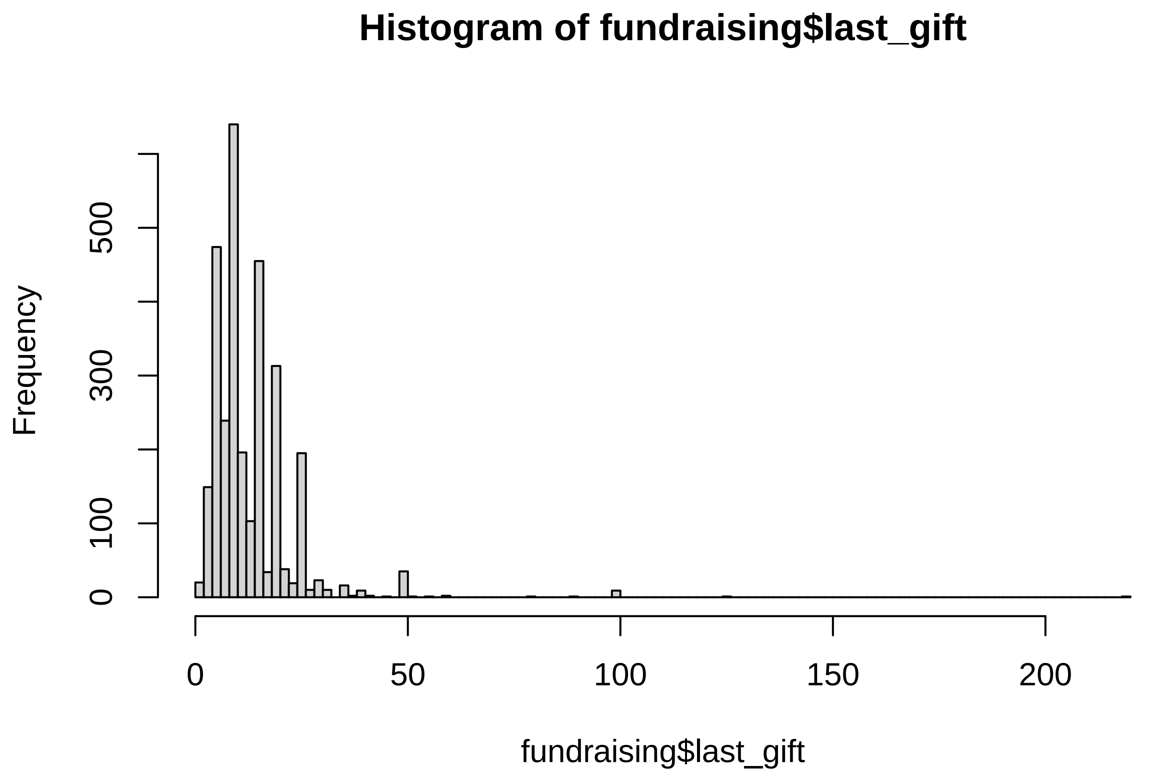

```r
hist(fundraising$med_fam_inc, breaks = 100)
```



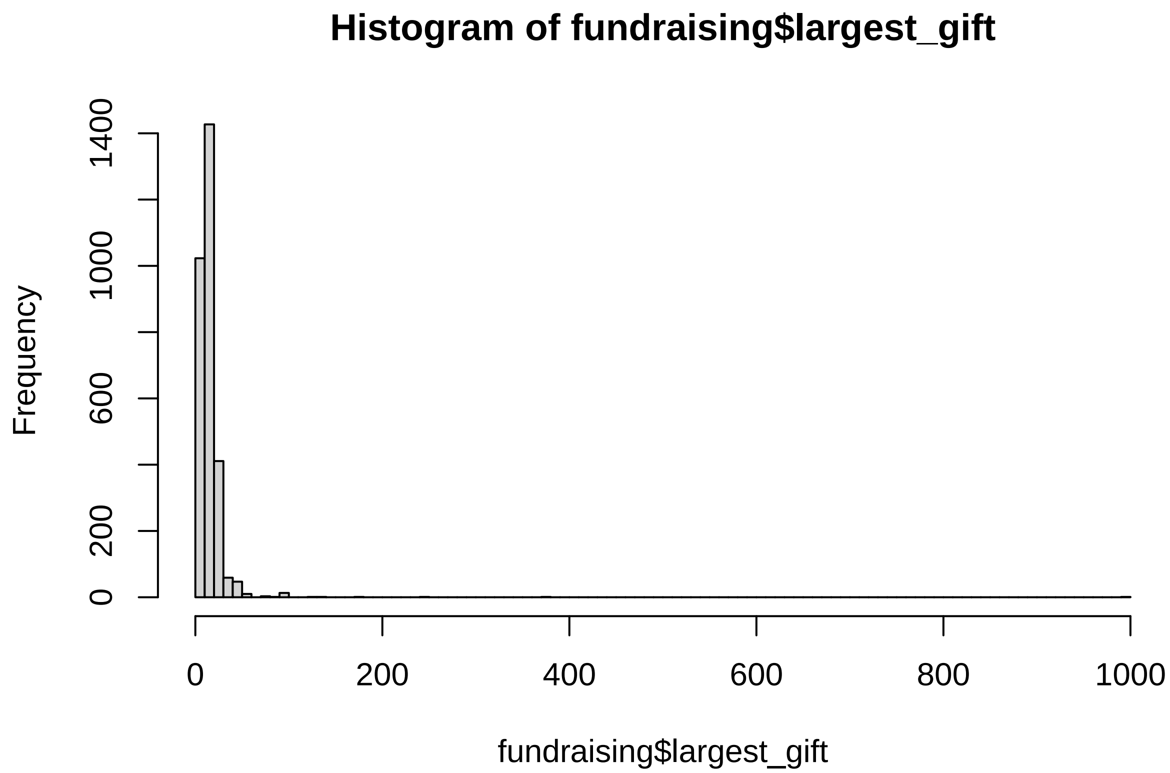

```
hist(fundraising$home_value, breaks = 100)
```



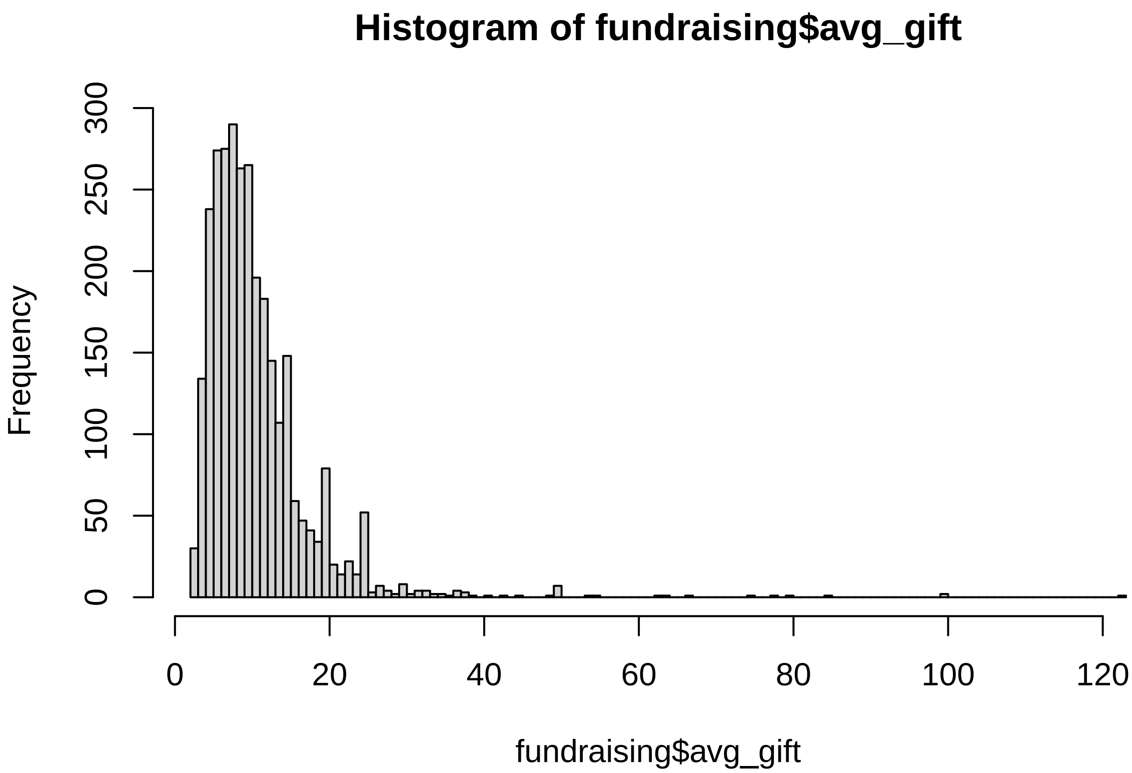

```
hist(fundraising$last_gift, breaks = 100)
```




```
hist(fundraising$largest_gift, breaks = 100)
```



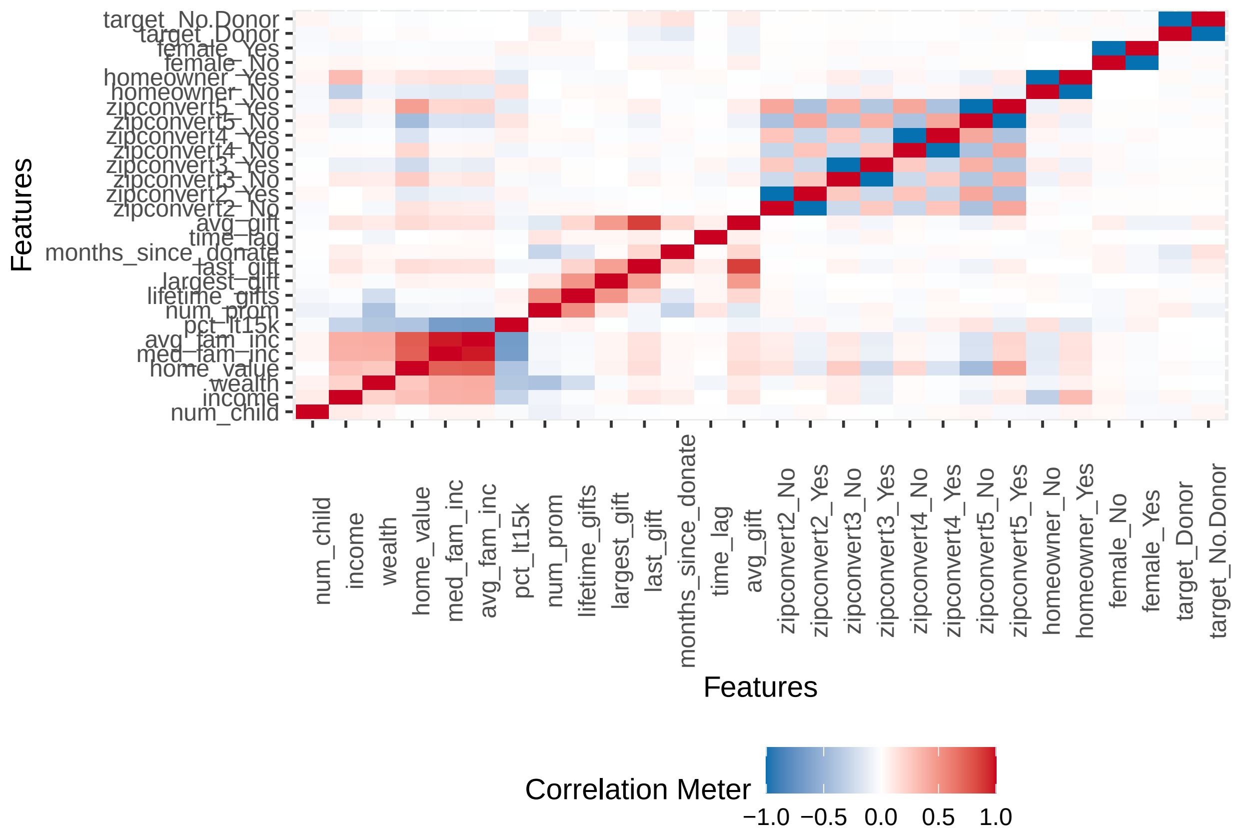

```r
hist(fundraising$avg_gift, breaks = 100)
```



Here, the following variables all show left skewedness:

Avg_fam_inc Med_fam_inc Home_value Last_gift largest_gift avg_gift

```
plot_correlation(na.omit(fundraising), maxcat = 5L)
```



Above, I also experimented with plotting a heat map correlation matrix in order to see which variables could possibly have an effect on the target variable. Already, I can see the following variables have a strong correlation with the target donor variable:

- avg_gift
- months_since_donate
- last_gift
- lifetime_gifts
- num_prom
- num_child
- income
- wealth
- pct_lt15k

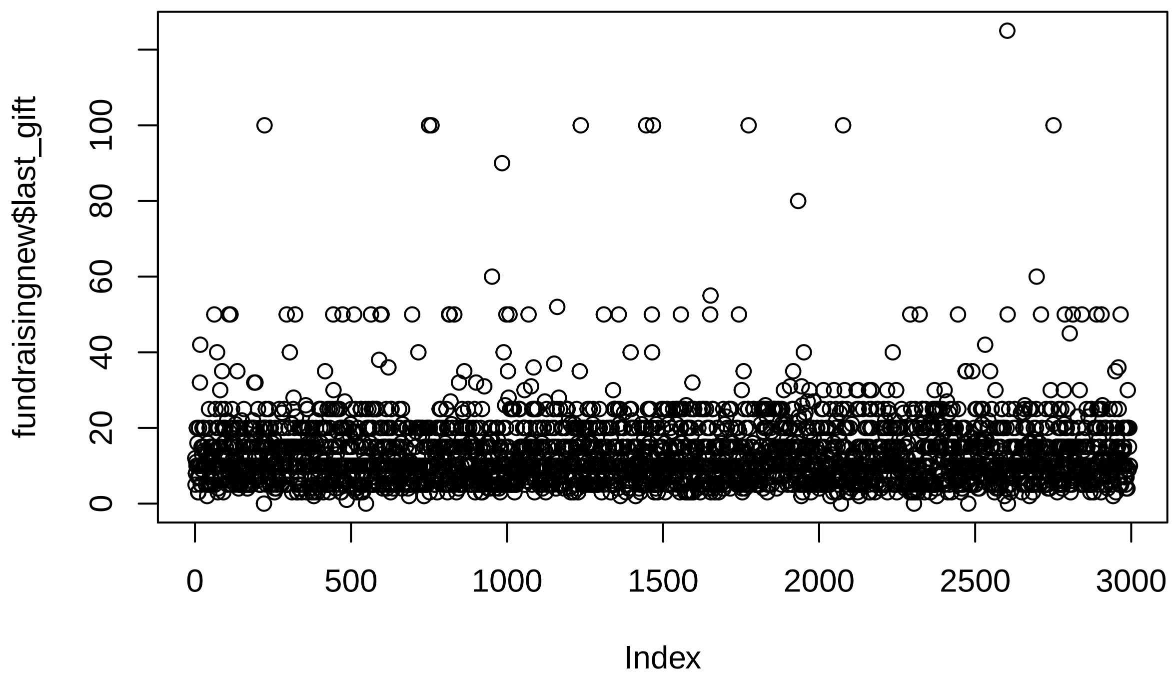
The rest of the variables either have a weak correlation or no correlation at all.

```
fundraisingnew <- subset(fundraising, largest_gift < 1000
                          & lifetime_gifts < 2000
                          & last_gift <200)
                          #& avg_fam_inc > 0
                          #& med_fam_inc > 0
                         # & last_gift > 0)
                         # & num_child < 5)
                          #& largest_gift > 0)
                          #& avg_gift > 0
```

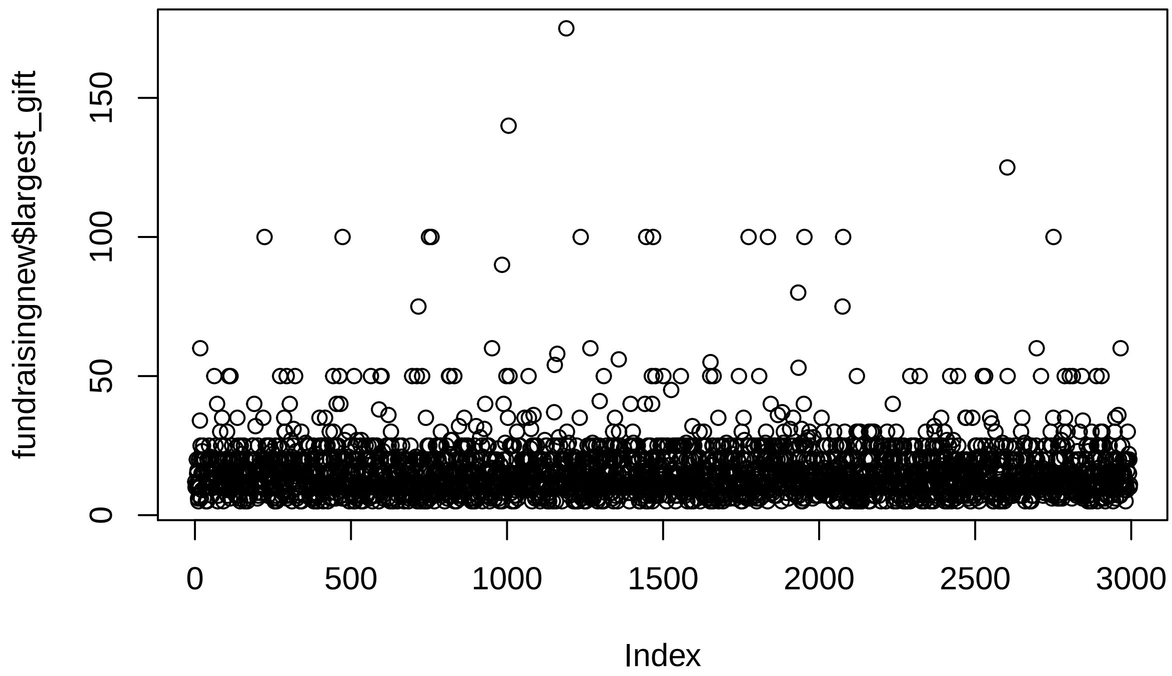
```
                     #   & home_value > 0
                   #   & wealth > 0)
```

I subsetted the data above on fundraisingnew to exclude outliers from the dataset in order to improve the prediction accuracy of the model.

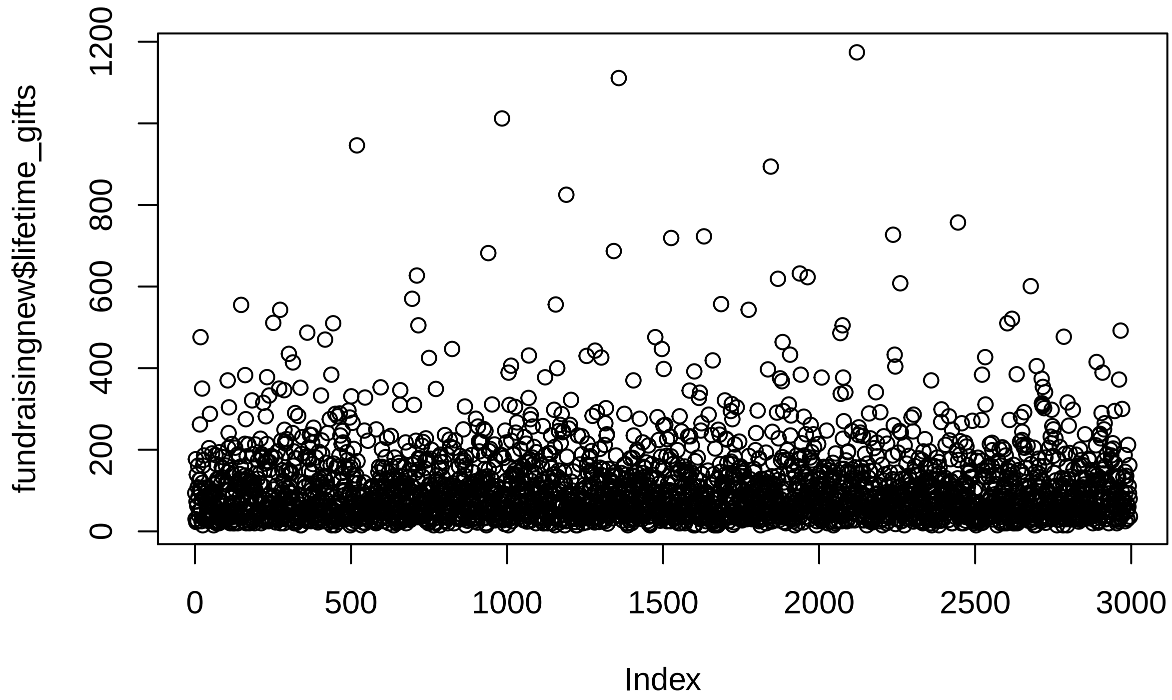
```
plot(fundraisingnew$last_gift)
```




```
plot(fundraisingnew$largest_gift)
```




```
plot(fundraisingnew$lifetime_gifts)
```



Based on the plots above on the new dataset, last_gift, largest_gift, and lifetime_gifts now show minimal to no outliers.

```
#fundraising2[1:5] <- sapply(fundraising2[1:5], as.character)
fundraisingnew[1:5] <- sapply(fundraisingnew[1:5], as.numeric)
#fundraising3[1:5]
fundraisingnew[8] <- sapply(fundraisingnew[8], as.numeric)
```

In order to improve the dataset and the accuracy of the model, I transformed every factor variable except the target variable into a numeric variable in order to have a cleaner dataset.

The code below is used in order to view the new dataset:

```
view(fundraisingnew)
```

```
str(fundraisingnew)
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
## tibble [2,996 x 21] (S3: tbl_df/tbl/data.frame)
##  $ zipconvert2        : num [1:2996] 2 1 1 1 1 1 1 2 1 2 ...
##  $ zipconvert3        : num [1:2996] 2 2 2 1 1 2 2 2 2 2 ...
##  $ zipconvert4        : num [1:2996] 1 1 1 1 1 1 2 1 1 1 ...
```

```
##  $ zipconvert5       : num [1:2996] 1 2 2 1 1 2 1 1 2 1 ...
##  $ homeowner         : num [1:2996] 1 2 1 1 1 1 1 1 1 1 ...
##  $ num_child         : num [1:2996] 1 2 1 1 1 1 1 1 1 1 ...
##  $ income            : num [1:2996] 1 5 3 4 4 4 4 4 4 1 ...
##  $ female            : num [1:2996] 2 1 2 2 1 1 2 1 1 1 ...
##  $ wealth            : num [1:2996] 7 8 4 8 8 8 5 8 8 5 ...
##  $ home_value        : num [1:2996] 698 828 1471 547 482 ...
##  $ med_fam_inc       : num [1:2996] 422 358 484 386 242 450 333 458 541 203 ...
##  $ avg_fam_inc       : num [1:2996] 463 376 546 432 275 498 388 533 575 271 ...
##  $ pct_lt15k         : num [1:2996] 4 13 4 7 28 5 16 8 11 39 ...
##  $ num_prom          : num [1:2996] 46 32 94 20 38 47 51 21 66 73 ...
##  $ lifetime_gifts    : num [1:2996] 94 30 177 23 73 139 63 26 108 161 ...
##  $ largest_gift      : num [1:2996] 12 10 10 11 10 20 15 16 12 6 ...
##  $ last_gift         : num [1:2996] 12 5 8 11 10 20 10 16 7 3 ...
##  $ months_since_donate: num [1:2996] 34 29 30 30 31 37 37 30 31 32 ...
##  $ time_lag          : num [1:2996] 6 7 3 6 3 3 8 6 1 7 ...
##  $ avg_gift          : num [1:2996] 9.4 4.29 7.08 7.67 7.3 ...
##  $ target            : Factor w/ 2 levels "Donor","No Donor": 1 1 2 2 1 1 1 2 1 1 ...
```

The above shows all factor variables have now been converted to numeric minus the target variable.

Training the model

The following code below is used to partition the model. I experimented with several different splits:

80/20 70/30 75/25 90/10

Eventually, I decided to stick with 80% in my training dataset and 20% in my testing dataset for my first run because this split gave me the best prediction accuracy for my final model.

```
set.seed(1)
inTrain <- createDataPartition(y = fundraisingnew$target, p = 0.80, list = FALSE)
training<- fundraisingnew[inTrain,]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
testing <- fundraisingnew[-inTrain,]
```

```
dim(training)
```

```
## [1] 2397   21
```

The above shows 2397 variables are in my training dataset.

```
dim(testing)
```

```
## [1] 599  21
```

The above shows 599 variables for my testing dataset.

Random Forest

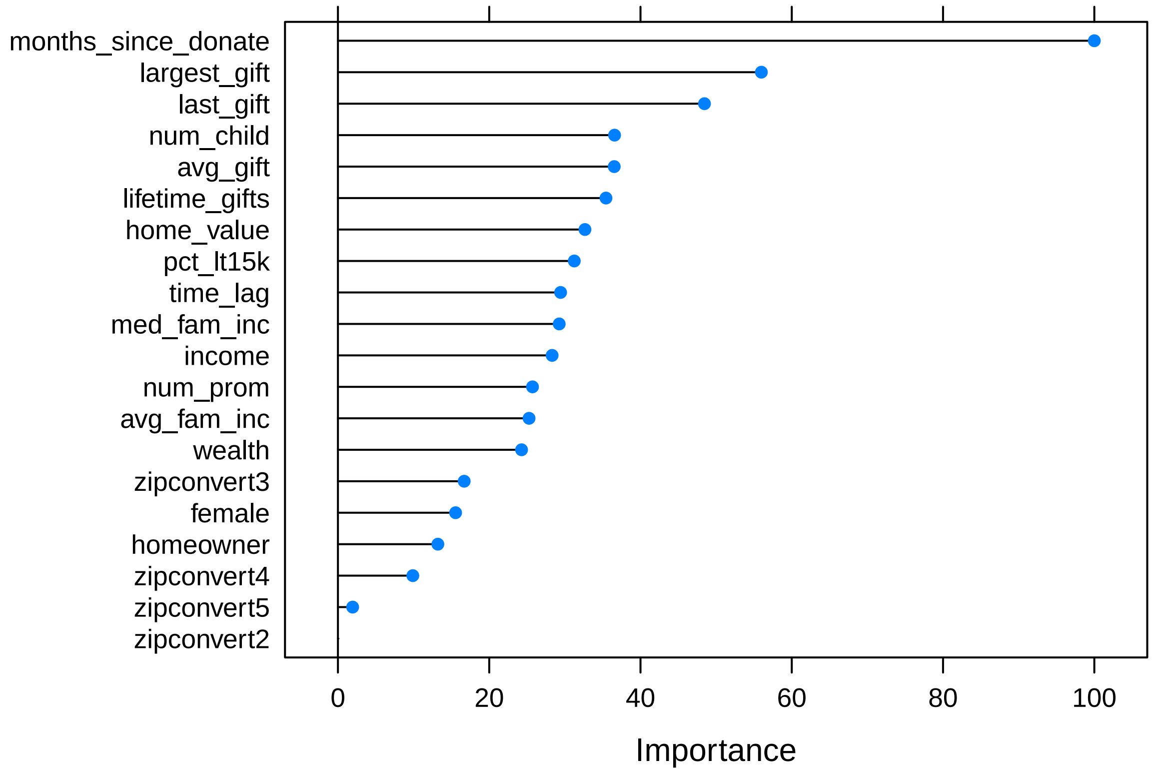
First, I experimented with a random forest model in order to figure out which variables would be deemed important for the model. I used every predictor as the independent variables against the target variable. I used repeated cross validation against the data set 100 times in order to divide my training dataset 100 times in order to get the best possible accuracy from the random forest model.

```
fundraising.rfnew <- train(target~., data = training, method = 'rf',
                           trContol = trainControl("repeatedcv", number = 100), importance = TRUE)
```

```
varImp(fundraising.rfnew)
```

```
## rf variable importance
##
##                      Importance
## months_since_donate    100.000
## largest_gift            55.984
## last_gift               48.457
## num_child               36.573
## avg_gift                36.531
## lifetime_gifts          35.439
## home_value              32.657
## pct_lt15k               31.248
## time_lag                29.440
## med_fam_inc             29.256
## income                  28.317
## num_prom                25.726
## avg_fam_inc             25.268
## wealth                  24.277
## zipconvert3             16.693
## female                  15.556
## homeowner               13.212
## zipconvert4              9.901
## zipconvert5              1.944
## zipconvert2              0.000
```

```
plot(varImp(fundraising.rfnew))
```



Based on the plot above, I can see that months_since_donate, largest_gift, avg_gift, last_gift, and num_child were the highest variables in that order in the dataset.

```
rf.pred = predict(fundraising.rfnew, testing)
table(rf.pred, testing$target)
```

```
##
## rf.pred    Donor No Donor
##   Donor      169      141
##   No Donor   130      159
```

I ran the random forest model against my test set to see how well it would predict donors:

```
1- (168 + 44) / (168 + 130 + 155 + 144)
```

```
## [1] 0.6448911
```

The test error rate obtained is 0.6448%.

Next, I transformed some of the predictors in the future_fundraising dataset to numeric in order to match what was obtained in the testing set and not pull back any error when running the model against the future_fundraising dataset.

```r
future_fundraising_transformed <- future_fundraising
future_fundraising_transformed[1:5] <- sapply(future_fundraising_transformed[1:5], as.numeric)
future_fundraising_transformed[8] <- sapply(future_fundraising_transformed[8], as.numeric)
```

The bottom code shows the random forest model against the transformed future_fundraising dataset:

```r
rf.pred = predict(fundraising.rfnew, future_fundraising_transformed)
```

I next saved the file to my drive and inputted the file into the shiny app in Blackboard:

```r
outdata<-data.frame(rf.pred)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"randomforestfile.csv")
```

When I ran the file, I received a prediction accuracy of 0.525%. When I saw that, I felt I needed to improve the accuracy.

Logistic Regression

With logistic regression, I decided to take some of the important predictors from the random forest model to use in my logistic regression model.

I decided to try out an 80/20 split on the fundraisingnew dataset.

```r
set.seed(1)
inTrain <- createDataPartition(y = fundraisingnew$target, p = 0.80, list = FALSE)
training<- fundraisingnew[inTrain,]
testing <- fundraisingnew[-inTrain,]
```

Next, I ran the entire model against the training dataset to see which variables were important:

```r
glm.fit = glm(target~., data = training
              , family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = training)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.86619  -1.14515   0.00165   1.16391   1.76074
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          2.512e+01  6.148e+02   0.041   0.9674
## zipconvert2         -1.375e+01  3.074e+02  -0.045   0.9643
## zipconvert3          1.362e+01  3.074e+02   0.044   0.9646
## zipconvert4         -1.374e+01  3.074e+02  -0.045   0.9643
```

```
## zipconvert5         -1.372e+01  3.074e+02  -0.045   0.9644
## homeowner            3.538e-02  1.064e-01   0.333   0.7395
## num_child            3.319e-01  1.310e-01   2.535   0.0113 *
## income              -6.034e-02  2.949e-02  -2.046   0.0408 *
## female              -1.582e-02  8.597e-02  -0.184   0.8540
## wealth               1.072e-03  2.040e-02   0.053   0.9581
## home_value          -7.558e-05  8.050e-05  -0.939   0.3478
## med_fam_inc         -1.265e-03  1.038e-03  -1.219   0.2228
## avg_fam_inc          1.708e-03  1.121e-03   1.525   0.1274
## pct_lt15k            6.259e-04  5.023e-03   0.125   0.9008
## num_prom             1.052e-03  3.341e-03   0.315   0.7529
## lifetime_gifts      -9.154e-04  7.978e-04  -1.147   0.2512
## largest_gift         4.987e-03  8.726e-03   0.571   0.5677
## last_gift            9.795e-03  9.737e-03   1.006   0.3144
## months_since_donate  6.551e-02  1.149e-02   5.703 1.18e-08 ***
## time_lag            -1.067e-02  7.823e-03  -1.364   0.1726
## avg_gift             8.968e-03  1.403e-02   0.639   0.5227
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3322.9  on 2396  degrees of freedom
## Residual deviance: 3236.1  on 2376  degrees of freedom
## AIC: 3278.1
##
## Number of Fisher Scoring iterations: 12
```

Based on the output above, it seems that months_since_donate, income, and num_child were the most significant predictors. However, I suspected that there was colinnearity and decided to check for variance inflation factors to see where the VIF was high.

```r
vif(glm.fit)
```

```
##         zipconvert2         zipconvert3         zipconvert4         zipconvert5
##        9.393415e+06        8.320413e+06        9.133064e+06        1.288077e+07
##           homeowner           num_child              income              female
##        1.146315e+00        1.030574e+00        1.340437e+00        1.020493e+00
##              wealth          home_value         med_fam_inc         avg_fam_inc
##        1.572419e+00        3.392019e+00        1.917540e+01        2.115840e+01
##           pct_lt15k            num_prom      lifetime_gifts        largest_gift
##        2.080150e+00        3.196176e+00        3.211313e+00        4.499509e+00
##           last_gift months_since_donate            time_lag            avg_gift
##        4.404852e+00        1.155774e+00        1.059565e+00        4.954420e+00
```

Based on the above, it seems zipconvert5 shows a high VIF of 9.75. Thus, I ran the model again without zipconvert5:

```r
glm.fit = glm(target~. - zipconvert5, data = training
              , family = binomial)
summary(glm.fit)
```

```
##
```

```
## Call:
## glm(formula = target ~ . - zipconvert5, family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8684  -1.1472   0.4228   1.1635   1.7598
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.280e+00  5.759e-01  -3.959 7.53e-05 ***
## zipconvert2         -3.471e-02  1.210e-01  -0.287   0.7741
## zipconvert3         -8.572e-02  1.327e-01  -0.646   0.5182
## zipconvert4         -2.987e-02  1.263e-01  -0.237   0.8130
## homeowner            4.165e-02  1.063e-01   0.392   0.6950
## num_child            3.315e-01  1.309e-01   2.531   0.0114 *
## income              -6.178e-02  2.948e-02  -2.096   0.0361 *
## female              -1.212e-02  8.590e-02  -0.141   0.8878
## wealth              -5.905e-05  2.036e-02  -0.003   0.9977
## home_value          -6.999e-05  8.013e-05  -0.873   0.3824
## med_fam_inc         -1.284e-03  1.037e-03  -1.238   0.2156
## avg_fam_inc          1.705e-03  1.120e-03   1.521   0.1281
## pct_lt15k            2.848e-04  5.019e-03   0.057   0.9547
## num_prom             8.733e-04  3.338e-03   0.262   0.7936
## lifetime_gifts      -8.999e-04  7.974e-04  -1.129   0.2591
## largest_gift         4.838e-03  8.727e-03   0.554   0.5793
## last_gift            9.903e-03  9.735e-03   1.017   0.3090
## months_since_donate  6.515e-02  1.148e-02   5.675 1.38e-08 ***
## time_lag            -1.033e-02  7.815e-03  -1.322   0.1861
## avg_gift             8.694e-03  1.402e-02   0.620   0.5353
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3322.9  on 2396  degrees of freedom
## Residual deviance: 3240.7  on 2377  degrees of freedom
## AIC: 3280.7
##
## Number of Fisher Scoring iterations: 4
```

The above output now shows months_since_donate and income are still the most significant predictors.

```r
vif(glm.fit)
```

```
##         zipconvert2         zipconvert3         zipconvert4           homeowner
##            1.454972            1.550524            1.542155            1.147124
##           num_child              income              female              wealth
##            1.030614            1.341422            1.020486            1.570020
##          home_value         med_fam_inc         avg_fam_inc           pct_lt15k
##            3.383648           19.181243           21.174292            2.078077
##            num_prom      lifetime_gifts        largest_gift           last_gift
##            3.191739            3.210413            4.506768            4.411673
## months_since_donate            time_lag            avg_gift
##            1.155385            1.059243            4.958773
```

Now, avg_fam_inc is showing a very high VIF of 21.030. Thus, I removed avg_fam_inc as well and ran the model again:

```r
glm.fit = glm(target~. - zipconvert5 - avg_fam_inc, data = training
              , family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = target ~ . - zipconvert5 - avg_fam_inc, family = binomial,
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8628  -1.1461   0.4219   1.1630   1.7518
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.162e+00  5.702e-01  -3.792  0.00015 ***
## zipconvert2         -2.208e-02  1.206e-01  -0.183  0.85471
## zipconvert3         -9.100e-02  1.326e-01  -0.686  0.49242
## zipconvert4         -1.640e-02  1.259e-01  -0.130  0.89637
## homeowner            4.997e-02  1.060e-01   0.471  0.63743
## num_child            3.359e-01  1.308e-01   2.568  0.01021 *
## income              -5.760e-02  2.932e-02  -1.965  0.04947 *
## female              -1.550e-02  8.583e-02  -0.181  0.85667
## wealth               1.312e-03  2.033e-02   0.065  0.94853
## home_value          -3.900e-05  7.736e-05  -0.504  0.61416
## med_fam_inc          1.216e-04  4.724e-04   0.257  0.79687
## pct_lt15k           -1.290e-03  4.907e-03  -0.263  0.79264
## num_prom             7.854e-04  3.332e-03   0.236  0.81368
## lifetime_gifts      -8.684e-04  7.966e-04  -1.090  0.27564
## largest_gift         4.856e-03  8.722e-03   0.557  0.57768
## last_gift            9.947e-03  9.730e-03   1.022  0.30664
## months_since_donate  6.534e-02  1.148e-02   5.693 1.25e-08 ***
## time_lag            -9.501e-03  7.776e-03  -1.222  0.22175
## avg_gift             7.949e-03  1.398e-02   0.568  0.56978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3322.9  on 2396  degrees of freedom
## Residual deviance: 3243.0  on 2378  degrees of freedom
## AIC: 3281
##
## Number of Fisher Scoring iterations: 4
```

Once again, months_since_donate and income were the highest predictors.

```r
vif(glm.fit)
```

```
##         zipconvert2         zipconvert3         zipconvert4           homeowner
```

```
##           1.447613           1.548859           1.534489           1.144044
##          num_child             income             female             wealth
##           1.030195           1.328873           1.019890           1.567200
##         home_value        med_fam_inc           pct_lt15k           num_prom
##           3.159017           3.982023           1.988914           3.185290
##     lifetime_gifts       largest_gift          last_gift months_since_donate
##           3.207027           4.512062           4.418140           1.155738
##           time_lag           avg_gift
##           1.053476           4.954853
```

Based on the above output, there still seems to be a little colinearity with the VIF for last_gift being 4.98. I decided to remove last_gift.=:

```r
glm.fit = glm(target~. - zipconvert5 - avg_fam_inc - last_gift, data = training
              , family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = target ~ . - zipconvert5 - avg_fam_inc - last_gift,
##     family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8802  -1.1471   0.4429   1.1604   1.7575
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.228e+00  5.665e-01  -3.933 8.40e-05 ***
## zipconvert2          -2.174e-02  1.205e-01  -0.180  0.85689
## zipconvert3          -9.215e-02  1.325e-01  -0.695  0.48679
## zipconvert4          -1.697e-02  1.259e-01  -0.135  0.89281
## homeowner             5.185e-02  1.060e-01   0.489  0.62471
## num_child             3.396e-01  1.307e-01   2.598  0.00937 **
## income               -5.756e-02  2.931e-02  -1.964  0.04957 *
## female               -1.702e-02  8.580e-02  -0.198  0.84271
## wealth                1.149e-03  2.032e-02   0.057  0.95490
## home_value           -3.727e-05  7.732e-05  -0.482  0.62975
## med_fam_inc           1.305e-04  4.721e-04   0.276  0.78221
## pct_lt15k            -1.282e-03  4.904e-03  -0.261  0.79377
## num_prom              1.017e-03  3.323e-03   0.306  0.75969
## lifetime_gifts       -9.196e-04  7.942e-04  -1.158  0.24691
## largest_gift          8.368e-03  8.017e-03   1.044  0.29659
## months_since_donate   6.706e-02  1.135e-02   5.907 3.49e-09 ***
## time_lag             -9.339e-03  7.772e-03  -1.202  0.22952
## avg_gift              1.505e-02  1.213e-02   1.241  0.21467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3322.9  on 2396  degrees of freedom
## Residual deviance: 3244.0  on 2379  degrees of freedom
```

```
## AIC: 3280
##
## Number of Fisher Scoring iterations: 4
```

Once again, the above shows that income and month_since_donate are still the highest predictors.

```r
vif(glm.fit)
```

```
##         zipconvert2        zipconvert3        zipconvert4          homeowner
##            1.447003           1.548479           1.534196           1.143534
##           num_child             income             female             wealth
##            1.029486           1.328369           1.019366           1.567133
##          home_value        med_fam_inc           pct_lt15k           num_prom
##            3.160776           3.985409           1.988796           3.168884
##      lifetime_gifts       largest_gift months_since_donate           time_lag
##            3.202698           3.864179           1.130749           1.052985
##            avg_gift
##            3.746426
```

Based on the above, there is no colinearity present anymore. I decided to fit the entire model onto the testing dataset.

```r
glm.pred= predict(glm.fit, testing, type="response")
glm.class=rep("No Donor", nrow(testing))
glm.class[glm.pred> 0.49] = "Donor"
table(glm.class, testing$target)
```

```
##
## glm.class  Donor No Donor
##   Donor      138      162
##   No Donor   161      138
```

```r
1 - (146 + 119) / (146 + 152 + 180 + 119)
```

```
## [1] 0.5561139
```

The test error rate is about 0.55%. I then decided to run the logistic model against the future_fundraising dataset. I chose a probability of 0.4905 because this probability cutoff was generating the best accuracy.

```r
glm.pred= predict(glm.fit, future_fundraising_transformed, type="response")
glm.class=rep("No Donor", nrow(future_fundraising_transformed))
glm.class[glm.pred> 0.4905] = "Donor"
table(glm.class)
```

```
## glm.class
##    Donor No Donor
##       69       51
```

```r
outdata<-data.frame(glm.class)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"NewLogisticRregression.csv")
```

Running the test model gave me a 54.166677% accuracy rate. This improved from the Random Forest model but did not get me the accuracy I wanted. Therefore, I decided to experiment with several variables.

I kept experimenting with removing and adding different variables throughout the logistic regression process. Through countless trial and error, I ended up going with an 80/20 split with the training dataset from fundraisingnew with no transformations or removing outliers and with the following predictors:

- months_since_donate
- num_child
- num_prom
- pct_lt15k
- avg_fam_inc

Best Model

```r
set.seed(1)
inTrain <- createDataPartition(y = fundraisingnew$target, p = 0.80, list = FALSE)
training<- fundraisingnew[inTrain,]
testing <- fundraisingnew[-inTrain,]
```

```r
glm.fit = glm(target~ months_since_donate + num_prom+ pct_lt15k
              + num_child + avg_fam_inc  , data = training
              , family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = target ~ months_since_donate + num_prom + pct_lt15k +
##     num_child + avg_fam_inc, family = binomial, data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7892  -1.1547   0.6968   1.1706   1.7111
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.449e+00  4.523e-01  -5.414 6.18e-08 ***
## months_since_donate  7.168e-02  1.103e-02   6.501 8.00e-11 ***
## num_prom            -2.114e-03  1.928e-03  -1.096   0.2729
## pct_lt15k            2.551e-05  4.673e-03   0.005   0.9956
## num_child            3.033e-01  1.280e-01   2.370   0.0178 *
## avg_fam_inc         -7.974e-06  3.263e-04  -0.024   0.9805
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3322.9  on 2396  degrees of freedom
## Residual deviance: 3264.1  on 2391  degrees of freedom
## AIC: 3276.1
##
## Number of Fisher Scoring iterations: 4
```

The above output shows that months_since_donate and num_child are the most significant predictors. However, I know the combination of these 5 variables has a significant effect in improving the accuracy of my model.

I next ran the model with a probability cutoff of 0.4935 on the testing dataset.

```r
glm.pred= predict(glm.fit, testing, type="response")
glm.class=rep("No Donor", nrow(testing))
glm.class[glm.pred> 0.4935] = "Donor"
table(glm.class, testing$target)
```

```
##
## glm.class  Donor No Donor
##   Donor      136      167
##   No Donor   163      133
```

```r
1 - (141 + 126) / (141 + 158 + 174 + 121)
```

```
## [1] 0.5505051
```

The test error rate shows as 0.55%. I then run the glm.fit model against the future_fundraising dataset.

```r
glm.pred= predict(glm.fit, future_fundraising, type="response")
glm.class=rep("No Donor", nrow(future_fundraising))
glm.class[glm.pred> 0.4925] = "Donor"
table(glm.class)
```

```
## glm.class
##    Donor No Donor
##       72       48
```

```r
outdata<-data.frame(glm.class)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"Best model.csv")
```

Running the model above against the future_fundraising dataset has ranged from 0.60% to 0.641677777% accuracy. This was the best model that I was able to generate. I did run LDA, QDA, SVM, and GBM to see if I could obtain a better prediction accuracy.

LDA

I ran the lda model using months_since_donate, num_prom, pct_lt15k, avg_fam_inc, and num_child as my independent variables with target as my dependent variable. I decided to

```
lda.fit=train(target~ months_since_donate + num_prom + pct_lt15k +
                avg_fam_inc + num_child,data=training,method='lda',
              trControl = trainControl(method = "repeatedcv", number = 100))
lda.fit
```

```
## Linear Discriminant Analysis
##
## 2397 samples
##    5 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (100 fold, repeated 1 times)
## Summary of sample sizes: 2373, 2373, 2373, 2373, 2373, 2373, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.562337  0.1249578
```

As can be seen from the output above, I get an accuracy against the training set with 55%. Next, I ran the model against the testing dataset.

```
pred.lda<-predict(lda.fit,testing)
table(pred.lda, testing$target)
```

```
##
## pred.lda   Donor No Donor
##   Donor      173      147
##   No Donor   126      153
```

```
1 - (89 + 86) / (89 + 60 + 64 + 86)
```

```
## [1] 0.4147157
```

I get a test error rate of 41%. Next, I ran the LDA model against the future_fundraising dataset in order to see how accurate my predictions would be:

```
pred.lda<-predict(lda.fit,future_fundraising)
```

```
outdata<-data.frame(pred.lda)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"LDA Model.csv")
```

I get an accuracy rate of 0.425% which is by far the worst model I've generated. I then tried it with QDA to see how my accuracy would turn out.

QDA

```
qda.fit=train(target~ months_since_donate + num_prom + pct_lt15k + avg_fam_inc + num_child,
              data=training,method='qda',trControl = trainControl(method = "repeatedcv", number = 100))
qda.fit
```

```
## Quadratic Discriminant Analysis
##
## 2397 samples
##    5 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (100 fold, repeated 1 times)
## Summary of sample sizes: 2373, 2373, 2373, 2373, 2373, 2373, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5068297  0.01474055
```

The above output shows the accuracy turned out to be 0.50% for the training set. Next, I ran the model against the testing set.

```
pred.qda<-predict(qda.fit,testing)
table(pred.qda, testing$target)
```

```
##
## pred.qda   Donor No Donor
##   Donor      259      260
##   No Donor    40       40
```

```
1 - (128 + 17)/ (128 + 21 + 17 + 133)
```

```
## [1] 0.5150502
```

The test error rate above is shown as 0.51%. I then ran the qda model against the future_fundraising dataset.

```
pred.Qda<-predict(qda.fit,future_fundraising)
```

```
outdata<-data.frame(pred.qda)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"QDA Model.csv")
```

The accuracy turned out to be 0.46%. It was better than the LDA model but still far from being the best model.

Finally, I ran a support vector machine model against the following predictors:

- months_since_donate
- num_child
- num_prom
- pct_lt15k
- avg_fam_inc

Support Vector Machine

```
svm.linear <- svm(target ~  months_since_donate + num_child + pct_lt15k + avg_fam_inc + num_prom,
                  data = training, kernel = "linear", cost = 0.01)
summary(svm.linear)
```

```
##
## Call:
## svm(formula = target ~ months_since_donate + num_child + pct_lt15k +
##     avg_fam_inc + num_prom, data = training, kernel = "linear", cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  2274
##
##  ( 1134 1140 )
##
##
## Number of Classes:  2
##
## Levels:
##  Donor No Donor
```

As can be seen above, the number of support vectors obtained were 2575. Next, I ran the model against the test set and produced a confusion matrix.

```
pred.svm1<-predict(svm.linear,testing)
table(pred.svm1, testing$target)
```

```
##
## pred.svm1  Donor No Donor
##   Donor       55       46
##   No Donor   244      254
```

```
1 - (30 + 133) / (30 + 119 + 17 + 133)
```

```
## [1] 0.4548495
```

The test error rate for the SVM Linear model shows as 0.45%. I ran the model against the future_fundraising dataset.

```r
svm.pred<-predict(svm.linear,future_fundraising)
```

```r
outdata<-data.frame(svm.pred)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"SVM Linear Model.csv")
```

The accuracy for the model turned out to be 0.475%. While this was an improvement among the LDA and QDA Models, logistic regression was still by far the best one.

GBM Boosting

The last model I wanted to try was gbm. I ran the GBM model with the following predictors:

- months_since_donate
- num_prom
- num_child
- pct_lt15k
- avg_fam_inc

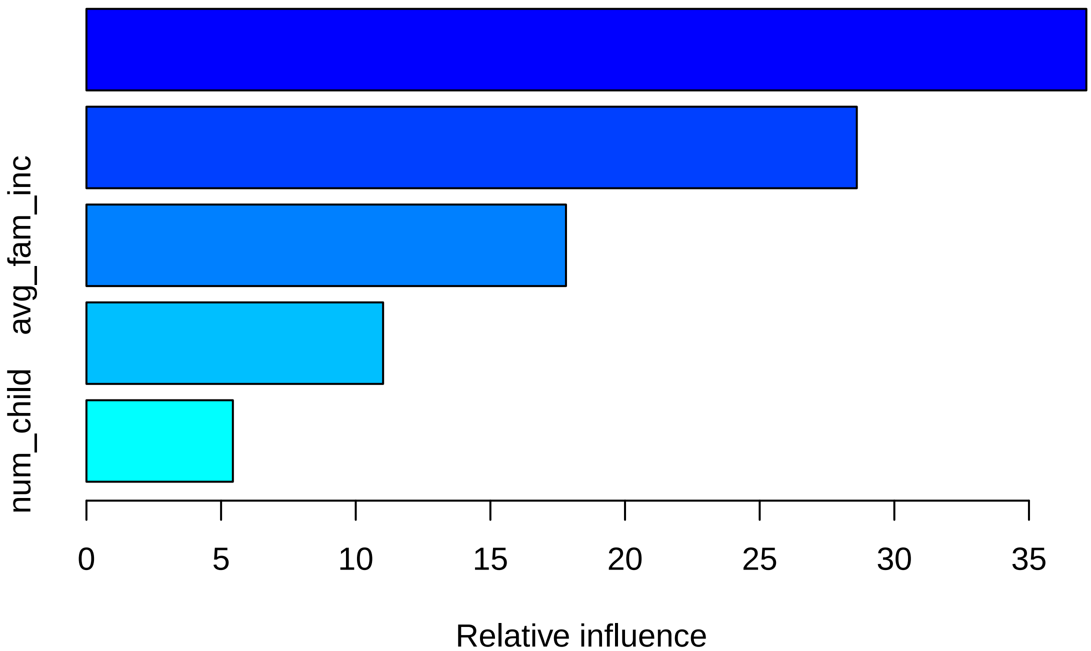
First, I partitioned the model with an 80/20 split as shown below:

```r
set.seed(1)
inTrain <- createDataPartition(y = fundraising$target, p = 0.80, list = FALSE)
training<- fundraising[inTrain,]
testing <- fundraising[-inTrain,]
```

Next, I ran the gbm model with 10 fold cross validation with the predictors mentioned previously. The distribution being used is bernoulli since the dependent variable is a binary classification model. The fraction that I chose to use was 0.80 with a metric of "Accuracy" since this is the main measure which I am testing for this model.

```r
train_control = trainControl(method = "cv", number = 10)
gbm.fit = train(target~ months_since_donate + num_prom + num_child + avg_fam_inc + pct_lt15k,
                data = training, distribution = "bernoulli", method = 'gbm',
                trControl = train_control, verbose = FALSE, metric = "Accuracy", bag.fraction = 0.80)
```

```r
summary(gbm.fit)
```




```
##                                     var  rel.inf
## months_since_donate months_since_donate 37.13065
## num_prom                       num_prom 28.60626
## avg_fam_inc                 avg_fam_inc 17.80769
## pct_lt15k                     pct_lt15k 11.01668
## num_child                     num_child  5.43872
```

Based on the output above, it seems every predictor has a relative influence on the target variable with months_since_donate having the highest influence.

```
gbm.pred=  predict(gbm.fit, testing)
gbm.class=rep("No Donor", nrow(testing))
gbm.class[gbm.pred>0.48] = "Donor"
```

```
## Warning in Ops.factor(gbm.pred, 0.48): '>' not meaningful for factors
```

```
table(gbm.pred, testing$target)
```

```
##
## gbm.pred   Donor No Donor
##   Donor      158      150
##   No Donor   141      150
```

```r
1 - (179 + 157) / (179 + 120 + 143 + 157)
```

```
## [1] 0.4390651
```

Based on the confusion matrix,the test error rate for the GBM model is 0.43%. Next, I decided to run the model against the future_fundraising dataset.

```r
pred.gbm<-predict(gbm.fit,future_fundraising)
```

```r
outdata<-data.frame(pred.gbm)
names(outdata)<-"value"
library(readr)
write_csv(outdata,"GBM Model.csv")
```

The accuracy rate based on the file I submitted turned out to be 0.4583%. The GBM Model also did not perform very well compared to the logistic regression model.

Conclusions/Recommendations

Based on the various models which were run during this project, the best variables which worked best were months_since_donate, num_child, num_prom, pct_lt15k, and avg_fam_inc with an 80/20 split with no transformations or variable exclusions being done on the model.

In summary, the most important predictors for deciding whether an individual will donate to this marketing campaign based on my models are a combination of the 5 variables:

1. Holding all other variables constant, the number of months since a person has donated will have a significant effect as to whether an individual will donate to the campaign.
2. Holding all other variables constant, the average family income will have a significant effect as to whether an individual will donate to the campaign.
3. Holding all other variables constant, the lifetime number of promotions received to date will have a significant effect as to whether an individual will donate to the campaign.
4. Holding all other variables constant, the percent earning less than $15K in a potential donor's neighborhood will have a significant effect as to whether an individual will donate to the campaign.
5. Holding all other variables constant, the number of children will have a significant effect as to whether an individual will donate to the campaign.

I believe looking at these 5 predictors from the fundraising dataset will help improve the cost effectiveness of the direct marketing campaign's efforts as well as predict who is more likely to donate to the campaign.