Direct Mail Fundrasing

Sai Ananthula

Documentation is at end

Step 1: Partioning

I used cross validation to estimate out of sample error with seed set to 12345.

Attaching needed libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(splines)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:purrr':
##
##     cross
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

Importing Data

```
train <-  read_rds("fundraising.rds")

test <- read_rds("future_fundraising.rds")
```

Step 2: Model Building

Importance table

I used the importance table from a model developed using bagging to explore which predictor variables are or are not relevant

```r
bagged_model <- randomForest(y = train$target, x = train[,1:20], importance = T)

importance(bagged_model)
```

```
##                           Donor   No Donor MeanDecreaseAccuracy
## zipconvert2         -1.14200173 -2.4982180           -2.6765799
## zipconvert3          1.42750629 -2.6839818           -1.0204007
## zipconvert4          0.03910175 -2.2445155           -1.5782929
## zipconvert5          1.05772707 -2.9746201           -1.5481586
## homeowner           -0.22327690  0.7561727            0.4197253
## num_child            2.51534125  2.9943269            3.8117276
## income               2.91689324  0.3568888            2.3091128
## female              -1.84324032  0.7218717           -0.8122784
## wealth               6.15073099 -3.8740822            1.7552280
## home_value           2.99728937  1.0532968            3.4405185
## med_fam_inc         -0.32201873  2.9927451            2.8279774
## avg_fam_inc         -1.24437640  5.5614275            4.6526065
## pct_lt15k           -0.04075357  1.8541335            1.6691857
## num_prom             0.54283826 -0.9293108           -0.3784580
## lifetime_gifts       3.07427599 -1.4263310            1.5023055
## largest_gift         9.07151572 -1.8798224            7.2304037
## last_gift            5.48454950  0.6903747            6.0176163
## months_since_donate  7.83431866  6.0989739           10.2234337
## time_lag             0.48747505 -2.0950442           -1.2740827
## avg_gift             8.51513296 -1.7295286            7.3300759
##                     MeanDecreaseGini
## zipconvert2                 14.86235
## zipconvert3                 13.60972
## zipconvert4                 14.52238
## zipconvert5                 15.55076
## homeowner                   16.89400
## num_child                   12.35687
## income                      66.40431
## female                      19.37052
## wealth                      50.11828
## home_value                 148.70069
## med_fam_inc                133.48893
## avg_fam_inc                135.57735
## pct_lt15k                  111.17880
## num_prom                   121.80945
## lifetime_gifts             132.56253
## largest_gift                78.84027
## last_gift                   79.02510
## months_since_donate         91.34329
## time_lag                    99.27531
## avg_gift                   140.86634
```

Based off a bagged random tree model's importance ranking the most important predictors in decreasing order are: largest_gift, avg_gift, last_gift, num_child,pct_lt15k, months_since_donate.

Data Transformation & Subseting

However, further exploration is possible starting with stripping non-numeric columns from the dataset (see below). The homeowner and female columns will also be converted to numeric columns.

```
train <- train[,5:21]
test <- test[,5:20]

train <- train %>%
  mutate(homeowner = as.numeric(homeowner)) %>%
  mutate(female = as.numeric(female))


str(train)
```

```
## tibble [3,000 x 17] (S3: tbl_df/tbl/data.frame)
##  $ homeowner          : num [1:3000] 1 2 1 1 1 1 1 1 1 1 ...
##  $ num_child          : num [1:3000] 1 2 1 1 1 1 1 1 1 1 ...
##  $ income             : num [1:3000] 1 5 3 4 4 4 4 4 4 1 ...
##  $ female             : num [1:3000] 2 1 2 2 1 1 2 1 1 1 ...
##  $ wealth             : num [1:3000] 7 8 4 8 8 8 5 8 8 5 ...
##  $ home_value         : num [1:3000] 698 828 1471 547 482 ...
##  $ med_fam_inc        : num [1:3000] 422 358 484 386 242 450 333 458 541 203 ...
##  $ avg_fam_inc        : num [1:3000] 463 376 546 432 275 498 388 533 575 271 ...
##  $ pct_lt15k          : num [1:3000] 4 13 4 7 28 5 16 8 11 39 ...
##  $ num_prom           : num [1:3000] 46 32 94 20 38 47 51 21 66 73 ...
##  $ lifetime_gifts     : num [1:3000] 94 30 177 23 73 139 63 26 108 161 ...
##  $ largest_gift       : num [1:3000] 12 10 10 11 10 20 15 16 12 6 ...
##  $ last_gift          : num [1:3000] 12 5 8 11 10 20 10 16 7 3 ...
##  $ months_since_donate: num [1:3000] 34 29 30 30 31 37 37 30 31 32 ...
##  $ time_lag           : num [1:3000] 6 7 3 6 3 3 8 6 1 7 ...
##  $ avg_gift           : num [1:3000] 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 ...
```

Correlation Table

This table contains the correlation values from 1 (strong positive correlation) to -1 (strong negative correlation) for all pairs of numeric columns in the data set.

```
cor(train[1:16])
```

```
##                       homeowner    num_child       income       female
## homeowner           1.0000000000 -0.045246221 -0.320470476  0.001478444
## num_child          -0.0452462210  1.000000000  0.091893089  0.029596832
## income             -0.3204704764  0.091893089  1.000000000  0.043813261
## female              0.0014784437  0.029596832  0.043813261  1.000000000
## wealth             -0.0657703248  0.060175537  0.208993101  0.029384095
## home_value         -0.1187277614 -0.011964229  0.291973494  0.020975827
## med_fam_inc        -0.1382760204  0.046961647  0.367505334  0.023450114
## avg_fam_inc        -0.1334984472  0.047261395  0.378585352  0.025236799
```

```
## pct_lt15k            0.1375411145 -0.031717891 -0.283191234 -0.055176372
## num_prom            -0.0032595805 -0.086432604 -0.069008634 -0.038218180
## lifetime_gifts       0.0281638939 -0.050954766 -0.019565470 -0.037073694
## largest_gift         0.0333386920 -0.017554416  0.033180760 -0.001381799
## last_gift            0.0008854337 -0.012948678  0.109592754  0.046359831
## months_since_donate -0.0213538579 -0.005563603  0.077238810  0.045056823
## time_lag            -0.0262871410 -0.006069356 -0.001545727  0.008049159
## avg_gift             0.0083209968 -0.019688680  0.124055750  0.074529893
##                          wealth    home_value med_fam_inc avg_fam_inc
## homeowner           -0.06577032 -0.1187277614 -0.13827602 -0.13349845
## num_child            0.06017554 -0.0119642286  0.04696165  0.04726139
## income               0.20899310  0.2919734944  0.36750533  0.37858535
## female               0.02938410  0.0209758272  0.02345011  0.02523680
## wealth               1.00000000  0.2611611450  0.37776337  0.38589230
## home_value           0.26116115  1.0000000000  0.73815307  0.75256900
## med_fam_inc          0.37776337  0.7381530742  1.00000000  0.97227129
## avg_fam_inc          0.38589230  0.7525690021  0.97227129  1.00000000
## pct_lt15k           -0.37514558 -0.3990861577 -0.66536267 -0.68028480
## num_prom            -0.41211777 -0.0645138583 -0.05078270 -0.05731139
## lifetime_gifts      -0.22547332 -0.0240737013 -0.03524583 -0.04032716
## largest_gift        -0.02527652  0.0564942757  0.04703207  0.04310394
## last_gift            0.05259131  0.1588576542  0.13597600  0.13137862
## months_since_donate  0.03371398  0.0234285142  0.03233669  0.03126859
## time_lag            -0.06642133  0.0006789113  0.01520204  0.02434038
## avg_gift             0.09107875  0.1687736865  0.13716276  0.13175843
##                        pct_lt15k    num_prom lifetime_gifts largest_gift
## homeowner           0.137541114 -0.00325958     0.02816389  0.033338692
## num_child          -0.031717891 -0.08643260    -0.05095477 -0.017554416
## income             -0.283191234 -0.06900863    -0.01956547  0.033180760
## female             -0.055176372 -0.03821818    -0.03707369 -0.001381799
## wealth             -0.375145585 -0.41211777    -0.22547332 -0.025276518
## home_value         -0.399086158 -0.06451386    -0.02407370  0.056494276
## med_fam_inc        -0.665362675 -0.05078270    -0.03524583  0.047032066
## avg_fam_inc        -0.680284797 -0.05731139    -0.04032716  0.043103937
## pct_lt15k           1.000000000  0.03777518     0.05961881 -0.007882936
## num_prom            0.037775183  1.00000000     0.53861957  0.113810342
## lifetime_gifts      0.059618806  0.53861957     1.00000000  0.507262313
## largest_gift       -0.007882936  0.11381034     0.50726231  1.000000000
## last_gift          -0.061752121 -0.05586809     0.20205827  0.447236933
## months_since_donate -0.009014558 -0.28232212    -0.14462186  0.019789633
## time_lag           -0.019911490  0.11962322     0.03854575  0.039977035
## avg_gift           -0.062480892 -0.14725094     0.18232435  0.474830096
##                       last_gift months_since_donate      time_lag
## homeowner           0.0008854337        -0.021353858 -0.0262871410
## num_child          -0.0129486780        -0.005563603 -0.0060693555
## income              0.1095927542         0.077238810 -0.0015457272
## female              0.0463598310         0.045056823  0.0080491595
## wealth              0.0525913108         0.033713981 -0.0664213294
## home_value          0.1588576542         0.023428514  0.0006789113
## med_fam_inc         0.1359760028         0.032336691  0.0152020426
## avg_fam_inc         0.1313786241         0.031268594  0.0243403812
## pct_lt15k          -0.0617521213        -0.009014558 -0.0199114896
## num_prom           -0.0558680871        -0.282322122  0.1196232155
## lifetime_gifts      0.2020582715        -0.144621862  0.0385457538
```

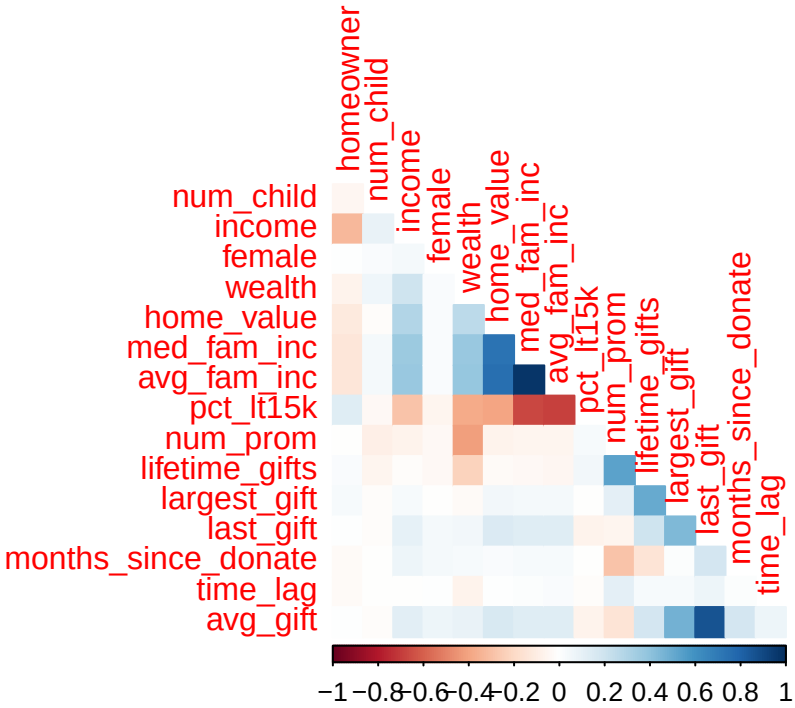
```
## largest_gift         0.4472369329         0.019789633  0.0399770354
## last_gift            1.0000000000         0.186715010  0.0751112090
## months_since_donate  0.1867150099         1.000000000  0.0155284995
## time_lag             0.0751112090         0.015528499  1.0000000000
## avg_gift             0.8663999778         0.189110799  0.0700816428
##                         avg_gift
## homeowner            0.008320997
## num_child           -0.019688680
## income               0.124055750
## female               0.074529893
## wealth               0.091078754
## home_value           0.168773687
## med_fam_inc          0.137162758
## avg_fam_inc          0.131758434
## pct_lt15k           -0.062480892
## num_prom            -0.147250943
## lifetime_gifts       0.182324349
## largest_gift         0.474830096
## last_gift            0.866399978
## months_since_donate  0.189110799
## time_lag             0.070081643
## avg_gift             1.000000000
```

The raw numbers representing correlation are a little hard to understand so the correlation plots below were built to see which predictors might be collinear.

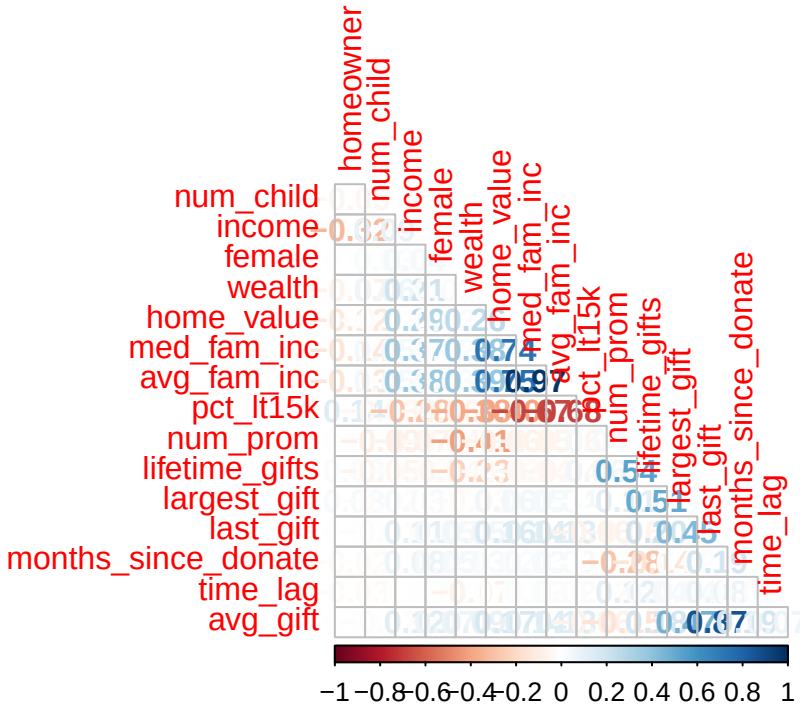
Correlation Plots

The 2 plots below are graphical representations of the correlation table above. The first table has just colored squares assigned a color on a scale from blue to red mirroring the 1 to -1 correlation scale. The second has the same color scheme but also features a truncated version of each correlation number.

```
corrplot(cor(train[1:16]),diag= FALSE, method = "color", type = "lower")
```




```
corrplot(cor(train[1:16]),diag= FALSE, method = "number", type = "lower")
```



Strong Correlations: absolute value(corr) > .7 home_value avg_fam_inc med_fam_inc pct_lt15k last_gift avg_gift

There is also a high likelihood avg_fam_inc and median_fam_inc are collinear due their correlation value is .98.

Based on the variables from the importance: largest_gift, avg_gift, last_gift, num_child, and pct_lt15k, months_since_donate.

I combined the two sets of variables since there is a lot of overlap but decided to exclude num_child since according to the correlation plot it has a low absolute correlation value. Home value which was not initially identified in the correlation plot but in the importance data will be included since there is some evidence of correlation and it did have a decent meandecreaseaccuracy.

Predictors Used For Model Building: largest_gift, home_value, avg_fam_in, med_fam_inc, pct_lt15k, last_gift, avg_gift, and months_since_donate.

Random Forest

Originally it was used to figure what predictors would be useful so the model was re-run but with a limited set of predictors.

```
cv_train <- trainControl(method="repeatedcv", number=5, repeats=3)
```

```
set.seed(12345)

randomForest <- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                          +last_gift + avg_gift + largest_gift +  months_since_donate , data = train,
                          method = "rf")

randomForest
```

```
## Random Forest
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.5301108  0.06021189
##   5     0.5277754  0.05553329
##   8     0.5262187  0.05243449
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The random forest resulted in an accuracy of 0.5301 or 53.01%

SVM-Radial

A radial SVM was among the three types of SVM's I tested since one type may have an advantage over the others.

```
set.seed(12345)

svm_Radial <- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                          +last_gift + avg_gift+ largest_gift +  months_since_donate, data = train, tr
                          method = "svmRadial")

svm_Radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.5518854  0.10381857
##   0.50  0.5501087  0.10027358
##   1.00  0.5462182  0.09248931
##
## Tuning parameter 'sigma' was held constant at a value of 0.273098
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.273098 and C = 0.25.
```

The radial SVM resulted in an accuracy of 0.5518 or 55.18%

SVM-Poly

I also decided to test a polynomial variation of a SVM to see if it would perform better.

```
set.seed(12345)

svm_Poly<- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                          +last_gift + avg_gift + largest_gift +  months_since_donate , data = train,
                          method = "svmPoly")

svm_Poly
```

```
## Support Vector Machines with Polynomial Kernel
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
## Resampling results across tuning parameters:
##
##   degree  scale  C     Accuracy   Kappa
##   1       0.001  0.25  0.5237796  0.04814559
##   1       0.001  0.50  0.5153320  0.03124031
##   1       0.001  1.00  0.5582174  0.11651398
##   1       0.010  0.25  0.5484413  0.09666277
##   1       0.010  0.50  0.5472183  0.09423796
##   1       0.010  1.00  0.5508856  0.10162579
##   1       0.100  0.25  0.5519948  0.10387740
##   1       0.100  0.50  0.5523280  0.10456059
##   1       0.100  1.00  0.5503278  0.10060066
##   2       0.001  0.25  0.5149991  0.03057236
##   2       0.001  0.50  0.5583287  0.11673278
```

```
##   2       0.001  1.00  0.5502193  0.10023689
##   2       0.010  0.25  0.5561084  0.11223294
##   2       0.010  0.50  0.5538846  0.10785745
##   2       0.010  1.00  0.5555513  0.11120107
##   2       0.100  0.25  0.5527722  0.10565961
##   2       0.100  0.50  0.5521067  0.10434127
##   2       0.100  1.00  0.5508830  0.10189214
##   3       0.001  0.25  0.5497735  0.09977278
##   3       0.001  0.50  0.5545522  0.10892245
##   3       0.001  1.00  0.5524413  0.10471887
##   3       0.010  0.25  0.5531067  0.10628923
##   3       0.010  0.50  0.5558835  0.11184268
##   3       0.010  1.00  0.5543287  0.10873205
##   3       0.100  0.25  0.5492189  0.09842847
##   3       0.100  0.50  0.5494422  0.09887937
##   3       0.100  1.00  0.5481098  0.09620249
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 2, scale = 0.001 and C = 0.5.
```

The polynomial SVM resulted in an accuracy of 0.5583 or 55.83%

SVM-Linear

I also decided to test a linear variation of a SVM to see if it would perform better.

```
set.seed(12345)

svm_Linear<- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                           +last_gift + avg_gift + largest_gift +  months_since_donate, data = train, t
                           method = "svmLinear")

svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5451069  0.09019354
##
## Tuning parameter 'C' was held constant at a value of 1
```

The polynomial SVM resulted in an accuracy of 0.5451 or 54.51%

Generalized Linear Model

I also decided to see if a generalized linear model since it seems there is a non-linear relationship in effect.

```
set.seed(12345)

glm_model<- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                          +last_gift + avg_gift + largest_gift +  months_since_donate , data = train,
                          method = "glm")

glm_model
```

```
## Generalized Linear Model
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.548218  0.09647081
```

The GLM resulted in an accuracy of 0.5482 or 54.82%

Linear Discriminant Analyis

Since this is classification a linear discriminant approach could be effective.

```
set.seed(12345)

lda_model<- train(target ~ home_value + avg_fam_inc + med_fam_inc + pct_lt15k
                          +last_gift + avg_gift + largest_gift +  months_since_donate, data = train,
                          method = "lda")

lda_model
```

```
## Linear Discriminant Analysis
##
## 3000 samples
##    8 predictor
##    2 classes: 'Donor', 'No Donor'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 2400, 2400, 2400, 2400, 2400, 2399, ...
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.5471065  0.09425052
```

The LDA resulted in an accuracy of 0.5471 or 54.71%

Step 3: Testing

SVM-Poly was the most accurate with an accuracy of 0.5583 or 55.83%

```
set.seed(12345)

svm_Poly_Preds <- predict(svm_Poly, newdata = test)
svm_Poly_Preds <- data.frame(svm_Poly_Preds)
write.csv(svm_Poly_Preds, file = "target.csv", row.names = FALSE)
```

Documentation

Business Objectives and Goals

A national veterans organizations wants to increase the effectiveness of their direct-mail campaign. Currently they have a response rate of 5.1% and the average donation is 13USD but that does not account for the cost of each letter which is .68USD. The goals of this analysis is to improve that rate by analyzing a sample of data from their larger data set and look for variables that might influence someone's decision to donate. By isolating these variables and building a classification model the organization should be able to raise more money whilst spending less money on mass mailing.

Data Sources and Data

The data is originally sourced from the organization's internal database which has the information of over 13 million donors. The sample provided has 3,000 records but is split into 2 files for training and then testing. The original sample provided was weighted so non-donors are underrepresented due to high amount of them in the original data source.

Type of Analyis

The initial analysis step was building a model using bagging since it would tell me which predictor variables were useful for further analysis and which ones were not. Due to this all the zip convert variables were removed from the data set.

A correlation table was then generated to see which predictors variables had some form of correlation with each other regardless if the correlation is positive or negative. The table was not extremely useful due to low-readability so it was then processed into 2 correlation plots that provide a superior graphical representation of the correlation between predictor variables in the data. The first correlation plot is just colored squares colored on a scale from red to blue with red = -1 (strong negative correlation) and blue = 1 (strong positive correlation). This first chart is too quickly see correlations because the second correlation plot also has the number representing the correlation number in each box so you can drill deeper into correlation between predictor variables.

From the importance table from the bagging model and the correlation plot I generated my list of predictor variables: largest_gift, home_value, avg_fam_in, med_fam_inc, pct_lt15k, last_gift, avg_gift, and months_since_donate.

I then decided to build a handful of models to explore different approaches to see which approach would perform the best.

The first model built was a random forest since I originally used bagging to generate an importance table I figured the random forest model would perform well in this classification problem.It performed the worst of all the models.

The second model built was a Support Vector Machine utilizing a polynomial function for the kernel. It performed the second best only losing to the SVM-Poly.

The third model built was a Support Vector Machine utilizing a radial function for the kernel. It performed the best and was the model that was used on the test data.

The fourth model built was a Support Vector Machine utilizing a linear function for the kernel. It performed in the middle of the pack and was beat out by the non-linear versions of SVM.

The fifth model built was generalized linear model since I was curious to how it would perform considering by this point it seemed certain the relationship was non-linear. The GLM's use of link functions allowed for this to work and the model performed okay.

The sixth and final model was a linear discriminant analysis since GLM performed okay I wanted to see how an LDA would fare on this problem. LDA was used instead of linear regression since this is a classification problem

---

Exclusions

Zip codes were removed from the data since they were not relevant to the analysis that was being run.

Variable Transformations

Female and homeowner had to be converted to the numeric data type so correlation functions could be executed on them.

Model Performance and Result Validation

For each model I built I used the accuracy calculated by the function due to cross-validation.

SVM-Polynomial was the most accurate so it was the model that was ran on the test data.

Model	Accuracy
Random Forests	53.01%
SVM-Radial	55.18%
SVM-Poly	55.83%
SVM-Linear	54.51%
GLM	54.82%
LDA	54.71%

Recommendation

The model performed rather well in absolute terms since the final accuracy was 55.83% of predicting people who would donate. This a giant leap compared to the mass mailing campaigns response rate of 5.1%. The model should be tried on a limited set of the overall larger data set to make sure real world performance mirrors test data performance. Also, the creation of a new data set that is more robust could help if it had more predictor variables. Especially in this context a predictor variable based on if someone is a veteran or is close to a veteran could help strengthen the model.