Fundraising Project

Chelsea Nowlin

8/7/2020

Business Objective and Goals ### This project will be focused on analyzing the fundraising dataset for the National Veteran's Organization that wants to determine the cost effectiveness of their direct marketing campaign via direct-mail. According to recent records, the overall response from their massive database of donors is only 5.1%. Out of the 5% who respond to the direct-mail who donated, the average donation is about $13.00. It costs the organization about $0.68 in marketing costs. The goal is to develop a classification model that maiximize profits by targeting households that are most likely to donate during the fundraising campaign. ###

Loading the packages

```
library(tidyverse)

## -- Attaching packages ------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

Data Sources and Data Used Loading the dataset

```
fundraising = readRDS("C:/Users/razzb/OneDrive/Documents/UTSA Graduate School
Classes/Data Algorithms/Final Project/fundraising.rds")
summary(fundraising)
```

```
##  zipconvert2 zipconvert3 zipconvert4 zipconvert5 homeowner
##  No :2352    Yes: 551    No :2357    No :1846    Yes:2312
##  Yes: 648    No :2449    Yes: 643    Yes:1154    No : 688
##
##
##
##
##    num_child         income       female         wealth
##  Min.   :1.000   Min.   :1.000   Yes:1831   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:3.000   No :1169   1st Qu.:5.000
##  Median :1.000   Median :4.000              Median :8.000
##  Mean   :1.069   Mean   :3.899              Mean   :6.396
##  3rd Qu.:1.000   3rd Qu.:5.000              3rd Qu.:8.000
##  Max.   :5.000   Max.   :7.000              Max.   :9.000
##    home_value      med_fam_inc      avg_fam_inc       pct_lt15k
##  Min.   :   0.0   Min.   :   0.0   Min.   :   0.0   Min.   : 0.00
##  1st Qu.: 554.8   1st Qu.: 278.0   1st Qu.: 318.0   1st Qu.: 5.00
##  Median : 816.5   Median : 355.0   Median : 396.0   Median :12.00
##  Mean   :1143.3   Mean   : 388.4   Mean   : 432.3   Mean   :14.71
##  3rd Qu.:1341.2   3rd Qu.: 465.0   3rd Qu.: 516.0   3rd Qu.:21.00
##  Max.   :5945.0   Max.   :1500.0   Max.   :1331.0   Max.   :90.00
##     num_prom      lifetime_gifts    largest_gift       last_gift
##  Min.   : 11.00   Min.   :  15.0   Min.   :   5.00   Min.   :  0.00
##  1st Qu.: 29.00   1st Qu.:  45.0   1st Qu.:  10.00   1st Qu.:  7.00
##  Median : 48.00   Median :  81.0   Median :  15.00   Median : 10.00
##  Mean   : 49.14   Mean   : 110.7   Mean   :  16.65   Mean   : 13.48
##  3rd Qu.: 65.00   3rd Qu.: 135.0   3rd Qu.:  20.00   3rd Qu.: 16.00
##  Max.   :157.00   Max.   :5674.9   Max.   :1000.00   Max.   :219.00
##  months_since_donate    time_lag         avg_gift             target
##  Min.   :17.00       Min.   : 0.000   Min.   :  2.139   Donor   :1499
##  1st Qu.:29.00       1st Qu.: 3.000   1st Qu.:  6.333   No Donor:1501
##  Median :31.00       Median : 5.000   Median :  9.000
##  Mean   :31.13       Mean   : 6.876   Mean   : 10.669
##  3rd Qu.:34.00       3rd Qu.: 9.000   3rd Qu.: 12.800
##  Max.   :37.00       Max.   :77.000   Max.   :122.167
```

Methodology 1. Partition the dataset 2. Check for missing values 3. Check summary statistics and look for outliers 4. Determine significance of model and parameters 5. Check Collinearity 6. Model Selection 7. Model prediction and validation 8. Test data

Data Partitioning

```
set.seed(12345)
trainIndex <- createDataPartition(fundraising$target, p = .8,
                                  list = FALSE,
                                  times = 1)
```

Training and Test data split

```
fundraisingTrain <- fundraising[ trainIndex,]
fundraisingTest <- fundraising[-trainIndex,]
```

Model Building

1. Exploratory Data Analysis Asking questions,

- Are they any significant paramters in the dataset that will be useful?
- Is there any collinearity present among the predictors?

Listing the variable names

```
names(fundraising)
```

```
##  [1] "zipconvert2"        "zipconvert3"        "zipconvert4"
##  [4] "zipconvert5"        "homeowner"          "num_child"
##  [7] "income"             "female"             "wealth"
## [10] "home_value"         "med_fam_inc"        "avg_fam_inc"
## [13] "pct_lt15k"          "num_prom"           "lifetime_gifts"
## [16] "largest_gift"       "last_gift"          "months_since_donate"
## [19] "time_lag"           "avg_gift"           "target"
```

```
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 3.6.3

## Loading required package: survival

## Warning: package 'survival' was built under R version 3.6.3

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
describe(fundraisingTrain)
```

```
## fundraisingTrain
##
##  21  Variables      2401  Observations
## ------------------------------------------------------------------------
-
```

```
## zipconvert2
##        n  missing distinct
##     2401        0        2
##
## Value         No   Yes
## Frequency   1893   508
## Proportion 0.788 0.212
## ---------------------------------------------------------------------------
-
## zipconvert3
##        n  missing distinct
##     2401        0        2
##
## Value        Yes    No
## Frequency    444  1957
## Proportion 0.185 0.815
## ---------------------------------------------------------------------------
-
## zipconvert4
##        n  missing distinct
##     2401        0        2
##
## Value         No   Yes
## Frequency   1884   517
## Proportion 0.785 0.215
## ---------------------------------------------------------------------------
-
## zipconvert5
##        n  missing distinct
##     2401        0        2
##
## Value         No   Yes
## Frequency   1472   929
## Proportion 0.613 0.387
## ---------------------------------------------------------------------------
-
## homeowner
##        n  missing distinct
##     2401        0        2
##
## Value        Yes    No
## Frequency   1850   551
## Proportion 0.771 0.229
## ---------------------------------------------------------------------------
-
## num_child
##        n  missing distinct     Info     Mean      Gmd
##     2401        0        5    0.145    1.072   0.1393
##
## lowest : 1 2 3 4 5, highest: 1 2 3 4 5
```

```
##
## Value          1     2     3     4     5
## Frequency   2279    83    27    11     1
## Proportion 0.949 0.035 0.011 0.005 0.000
## --------------------------------------------------------------------------
-
## income
##        n  missing distinct     Info     Mean      Gmd
##     2401        0        7    0.948    3.919    1.811
##
## lowest : 1 2 3 4 5, highest: 3 4 5 6 7
##
## Value          1     2     3     4     5     6     7
## Frequency    212   346   224   832   410   184   193
## Proportion 0.088 0.144 0.093 0.347 0.171 0.077 0.080
## --------------------------------------------------------------------------
-
## female
##        n  missing distinct
##     2401        0        2
##
## Value        Yes    No
## Frequency   1470   931
## Proportion 0.612 0.388
## --------------------------------------------------------------------------
-
## wealth
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0       10    0.832    6.429    2.505        1        2
##      .25      .50      .75      .90      .95
##        5        8        8        8        9
##
## lowest : 0 1 2 3 4, highest: 5 6 7 8 9
##
## Value          0     1     2     3     4     5     6     7     8     9
## Frequency     89   106   102   119   113   141   125   136  1321   149
## Proportion 0.037 0.044 0.042 0.050 0.047 0.059 0.052 0.057 0.550 0.062
## --------------------------------------------------------------------------
-
## home_value
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0     1341        1     1147    902.4      343      418
##      .25      .50      .75      .90      .95
##      560      815     1335     2395     3200
##
## lowest :    0  171  200  209  212, highest: 5855 5888 5908 5926 5945
## --------------------------------------------------------------------------
-
## med_fam_inc
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0      604        1    389.1    177.5      188      220 
##      .25      .50      .75      .90      .95 
##      279      355      464      593      683 
## 
## lowest :    0   68   71   72   77, highest: 1299 1340 1469 1496 1500
## --------------------------------------------------------------------------
-
## avg_fam_inc 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2401        0      632        1    433.3    179.4      232      264 
##      .25      .50      .75      .90      .95 
##      319      396      518      651      761 
## 
## lowest :    0   89   90  121  125, highest: 1217 1228 1236 1273 1331
## --------------------------------------------------------------------------
-
## pct_lt15k 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2401        0       67    0.999    14.74    12.98        0        2 
##      .25      .50      .75      .90      .95 
##        5       12       21       31       39 
## 
## lowest :  0  1  2  3  4, highest: 66 68 69 85 90
## --------------------------------------------------------------------------
-
## num_prom 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2401        0      121        1    48.75    25.32       20       22 
##      .25      .50      .75      .90      .95 
##       29       47       64       77       85 
## 
## lowest :  11  12  13  14  15, highest: 135 140 141 147 157
## --------------------------------------------------------------------------
-
## lifetime_gifts 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2401        0      390        1    110.4    96.57       25       30 
##      .25      .50      .75      .90      .95 
##       45       80      133      213      283 
## 
## lowest :   15.0   16.0   18.0   19.0   20.0, highest:  946.0 1012.0 1174.0
2200.0 5674.9
## --------------------------------------------------------------------------
-
## largest_gift 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2401        0       52    0.988    16.45    10.08        6        7 
##      .25      .50      .75      .90      .95 
##       10       15       20       25       30 
## 
```

```
## lowest :   5   6   7   8   9, highest: 125 140 175 250 375
## ---------------------------------------------------------------------------
-
## last_gift
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0       50    0.988    13.57    9.205        4        5
##      .25      .50      .75      .90      .95
##        7       10       16       25       25
##
## lowest :   0   1   2   3   4, highest:  80  90 100 125 219
## ---------------------------------------------------------------------------
-
## months_since_donate
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0       21    0.985    31.19    4.263       24       28
##      .25      .50      .75      .90      .95
##       29       31       34       37       37
##
## lowest : 17 18 19 20 21, highest: 33 34 35 36 37
## ---------------------------------------------------------------------------
-
## time_lag
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0       41    0.991     6.86    5.332        1        2
##      .25      .50      .75      .90      .95
##        3        5        9       13       17
##
## lowest :  0  1  2  3  4, highest: 37 38 44 48 62
## ---------------------------------------------------------------------------
-
## avg_gift
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2401        0     1081        1    10.72    6.568    4.000    4.667
##      .25      .50      .75      .90      .95
##    6.364    9.071   12.842   18.571   22.692
##
## lowest :   2.138889   2.354839   2.439815   2.445946   2.463415
## highest:  77.571429  80.000000  85.000000 100.000000 122.166667
## ---------------------------------------------------------------------------
-
## target
##        n  missing distinct
##     2401        0        2
##
## Value         Donor No Donor
## Frequency      1200     1201
## Proportion      0.5      0.5
## ---------------------------------------------------------------------------
-
```

Checking for missing values

```
sum(is.na(fundraisingTrain))

## [1] 0
```

There are no missing values present in the dataset.

Creating summary statistics for the variables in the training dataset. This will give us an idea about the metrics of our targeted household population. In summary, based on the skim function used, the typical house is: - Middle class (based on income levels, home value, average and median family income) - 1 child - High wealth rating - Donates infrequently - Smaller donations - Majority female

```
library(skimr)

## Warning: package 'skimr' was built under R version 3.6.3

skim(fundraisingTrain)
```

Data summary

Name	fundraisingTrain
Number of rows	2401
Number of columns	21
_______________________	
Column type frequency:	
factor	7
numeric	14
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
zipconvert2	0	1	FALSE	2	No: 1893, Yes: 508
zipconvert3	0	1	FALSE	2	No: 1957, Yes: 444
zipconvert4	0	1	FALSE	2	No: 1884, Yes: 517
zipconvert5	0	1	FALSE	2	No: 1472, Yes: 929
homeowner	0	1	FALSE	2	Yes: 1850, No: 551
female	0	1	FALSE	2	Yes: 1470, No: 931
target	0	1	FALSE	2	No : 1201, Don: 1200

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
num_child	0	1	1.07	0.35	1.00	1.00	1.00	1.00	5.00	▇▁▁▁▁
income	0	1	3.92	1.63	1.00	3.00	4.00	5.00	7.00	▆▂▇▃▃
wealth	0	1	6.43	2.54	0.00	5.00	8.00	8.00	9.00	▁▁▂▃▇
home_value	0	1	1146.67	956.27	0.00	560.00	815.00	1335.00	5945.00	▇▃▁▁▁
med_fam_inc	0	1	389.06	174.76	0.00	279.00	355.00	464.00	1500.00	▅▇▁▁▁
avg_fam_inc	0	1	433.32	169.12	0.00	319.00	396.00	518.00	1331.00	▁▇▃▁▁
pct_lt15k	0	1	14.74	12.16	0.00	5.00	12.00	21.00	90.00	▇▃▁▁▁
num_prom	0	1	48.75	22.67	11.00	29.00	47.00	64.00	157.00	▇▇▃▁▁
lifetime_gifts	0	1	110.44	157.49	15.00	45.00	80.00	133.00	5674.90	▇▁▁▁▁
largest_gift	0	1	16.45	14.17	5.00	10.00	15.00	20.00	375.00	▇▁▁▁▁
last_gift	0	1	13.57	10.71	0.00	7.00	10.00	16.00	219.00	▇▁▁▁▁
months_since_donate	0	1	31.19	4.08	17.00	29.00	31.00	34.00	37.00	▁▁▅▇▅
time_lag	0	1	6.86	5.52	0.00	3.00	5.00	9.00	62.00	▇▁▁▁▁
avg_gift	0	1	10.72	7.48	2.14	6.36	9.07	12.84	122.17	▇▁▁▁▁

Determining the data types for each of the variables. This will help when building the classification model.

```
str(fundraisingTrain)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2401 obs. of  21 variables:
##  $ zipconvert2       : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 2 1 2 1
...
##  $ zipconvert3       : Factor w/ 2 levels "Yes","No": 2 2 2 1 2 2 2 2 2 2
...
##  $ zipconvert4       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1
```

```
...
##  $ zipconvert5        : Factor w/ 2 levels "No","Yes": 1 2 2 1 2 1 1 2 1 2
...
##  $ homeowner          : Factor w/ 2 levels "Yes","No": 1 2 1 1 1 1 1 1 1 2
...
##  $ num_child          : num  1 2 1 1 1 1 1 1 1 1 ...
##  $ income             : num  1 5 3 4 4 4 4 4 1 2 ...
##  $ female             : Factor w/ 2 levels "Yes","No": 2 1 2 2 1 2 1 1 1 2
...
##  $ wealth             : num  7 8 4 8 8 5 8 8 5 8 ...
##  $ home_value         : num  698 828 1471 547 857 ...
##  $ med_fam_inc        : num  422 358 484 386 450 333 458 541 203 337 ...
##  $ avg_fam_inc        : num  463 376 546 432 498 388 533 575 271 402 ...
##  $ pct_lt15k          : num  4 13 4 7 5 16 8 11 39 5 ...
##  $ num_prom           : num  46 32 94 20 47 51 21 66 73 27 ...
##  $ lifetime_gifts     : num  94 30 177 23 139 63 26 108 161 50 ...
##  $ largest_gift       : num  12 10 10 11 20 15 16 12 6 20 ...
##  $ last_gift          : num  12 5 8 11 20 10 16 7 3 20 ...
##  $ months_since_donate: num  34 29 30 30 37 37 30 31 32 37 ...
##  $ time_lag           : num  6 7 3 6 3 8 6 1 7 7 ...
##  $ avg_gift           : num  9.4 4.29 7.08 7.67 10.69 ...
##  $ target             : Factor w/ 2 levels "Donor","No Donor": 1 1 2 2 1 1
2 1 1 2 ...
```

Detmining if the zipcode variables are worth keeping in the model.

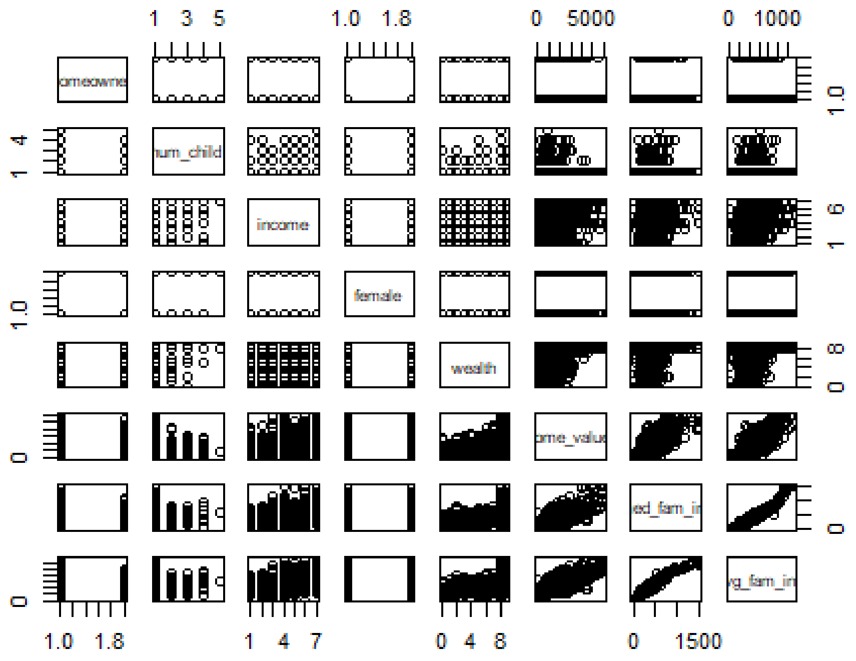
```
library(plyr)

## ------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:Hmisc':
##
##     is.discrete, summarize

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```
donor.count = subset(fundraising, target == "Donor")
dcount = count(donor.count, c('zipconvert2', 'zipconvert3', 'zipconvert4',
'zipconvert5'))
dcount

##   zipconvert2 zipconvert3 zipconvert4 zipconvert5 freq
## 1          No         Yes          No          No  269
## 2          No          No          No         Yes  592
## 3          No          No         Yes          No  318
## 4         Yes          No          No          No  320
```

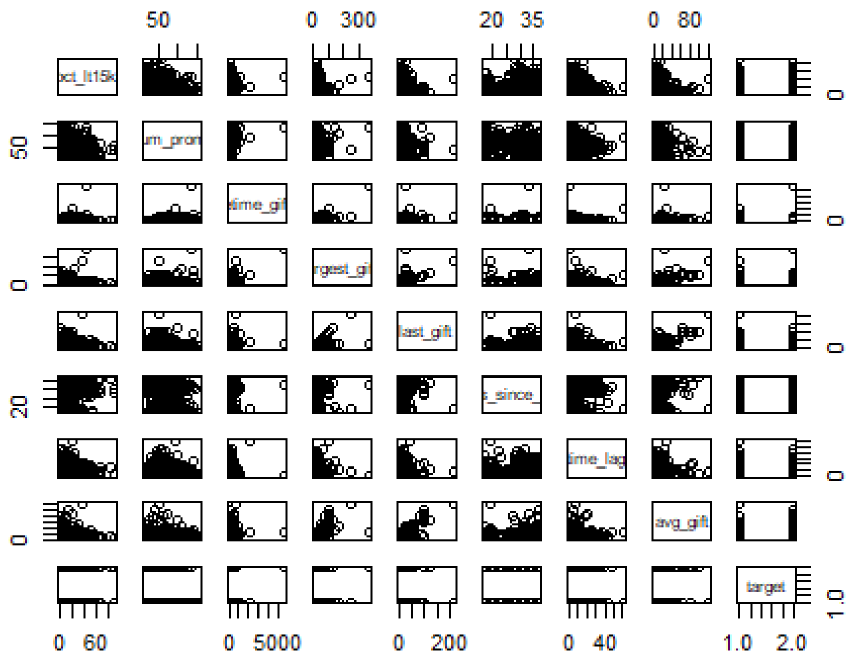
The zipcode variables will be excluded as they are not significant to the model. All the zip code zones have donors, so this will make it difficult to determine the zipcode with the most donors. Other variables may determine an easier method for determining the target population.

Checking for collinearity

```
model <- lm(target ~ homeowner + wealth + income + avg_fam_inc, data=
fundraisingTrain)

## Warning in model.response(mf, "numeric"): using type = "numeric" with a
## factor response will be ignored

## Warning in Ops.factor(y, z$residuals): '-' not meaningful for factors

pairs(fundraisingTrain[5:12])
```



```
pairs(fundraisingTrain[13:21])
```



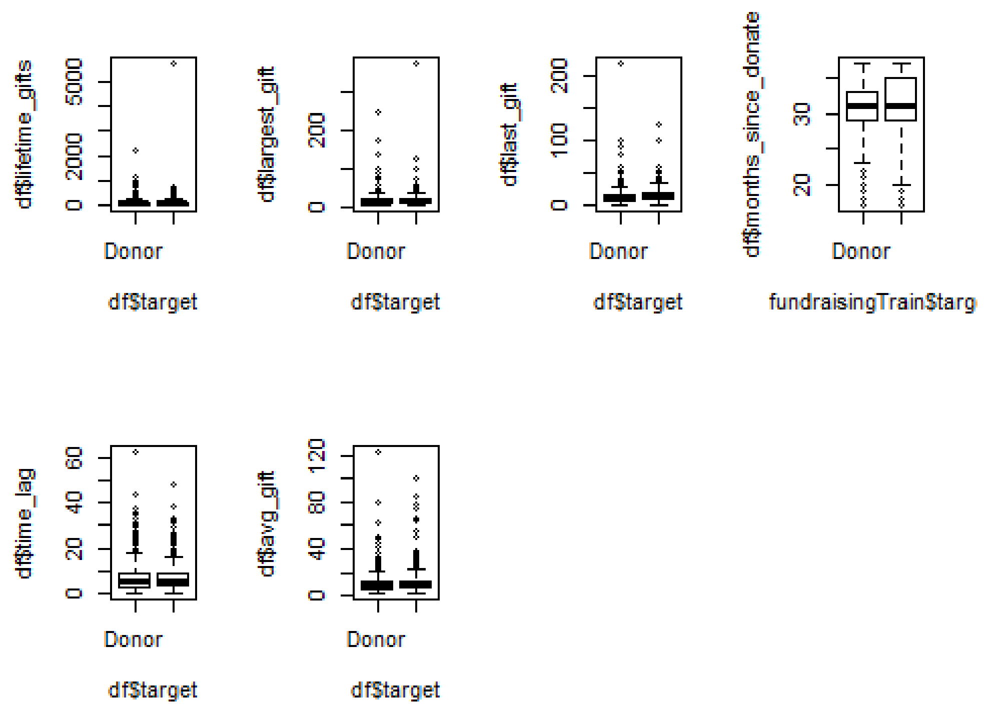
Based on the collinearity matrix, the variables, avg family income, median family income, and the home value have positive collinearity. This makes sense as homeownership can be tied to income.

Dropping the zipcodes from the training set

```
drop <- c("zipconvert2", "zipconvert3", "zipconvert4", "zipconvert5")
df = fundraisingTrain[,!(names(fundraisingTrain) %in%drop)]
```

Next, looking at the pedictors tied to actual donations.

```
par(mfrow = c(2,4))
boxplot(df$lifetime_gifts ~ df$target, data = df)
boxplot(df$largest_gift ~ df$target, data = df)
boxplot(df$last_gift ~ df$target, data = df)
boxplot(df$months_since_donate ~ fundraisingTrain$target, data = df)
boxplot(df$time_lag ~ df$target, data = df)
boxplot(df$avg_gift ~ df$target, data = df)
```



The boxplots represent the distribution among the donor related varaibles for donations. One the far right, the boxplot showing the distribution of months since the last donation, it has more predictive power based on the amount of time has passed since the last donation that could determine how likely someone is to donate again.

Exclusions - Removing paramters determined to not be significant to the model and would not contribute to a more accurate final model. Paramters are removed based on p-values determination of a score equal to or less than 0.05.

General Linear Model

```
fund.glm = glm(target ~ homeowner + num_child + income + female + wealth +
med_fam_inc + home_value + pct_lt15k + num_prom + lifetime_gifts +
largest_gift + last_gift + months_since_donate + time_lag + avg_gift, data =
df, family = binomial)
summary(fund.glm)

##
## Call:
## glm(formula = target ~ homeowner + num_child + income + female +
##     wealth + med_fam_inc + home_value + pct_lt15k + num_prom +
##     lifetime_gifts + largest_gift + last_gift + months_since_donate +
##     time_lag + avg_gift, family = binomial, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8835  -1.1511   0.5628   1.1517   1.8379
```

```
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)    
## (Intercept)         -1.937e+00  4.968e-01  -3.899 9.67e-05 ***
## homeownerNo          8.099e-02  1.050e-01   0.771  0.44061    
## num_child            2.728e-01  1.259e-01   2.167  0.03022 *  
## income              -8.031e-02  2.907e-02  -2.763  0.00574 ** 
## femaleNo             3.004e-02  8.605e-02   0.349  0.72703    
## wealth              -1.497e-02  2.009e-02  -0.745  0.45620    
## med_fam_inc          2.928e-04  4.503e-04   0.650  0.51550    
## home_value          -1.396e-04  6.603e-05  -2.114  0.03449 *  
## pct_lt15k           -3.948e-03  4.776e-03  -0.827  0.40845    
## num_prom            -3.563e-03  2.651e-03  -1.344  0.17891    
## lifetime_gifts      -1.178e-04  5.257e-04  -0.224  0.82272    
## largest_gift         3.478e-03  7.742e-03   0.449  0.65324    
## last_gift            9.886e-03  8.966e-03   1.103  0.27021    
## months_since_donate  6.845e-02  1.135e-02   6.030 1.64e-09 ***
## time_lag            -3.654e-04  7.726e-03  -0.047  0.96228    
## avg_gift            -8.020e-04  1.237e-02  -0.065  0.94832    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3328.5  on 2400  degrees of freedom
## Residual deviance: 3240.9  on 2385  degrees of freedom
## AIC: 3272.9
##
## Number of Fisher Scoring iterations: 4
```

The p-values help determine which variables left in the model are significant. All variables with p-values <= 0.05 will remain in the model and the non-significant variables will not be included.

2. Model Classification

Support Vector Machine Model 1: SVM Using training set of 2401 obs method = SVM Polynomial Kernel

```
library(caret)
library(kernlab)

##
## Attaching package: 'kernlab'

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

```
Model <- train(target ~ ., data = df,
               method = "svmPoly",
               na.action = na.omit,
               preProcess = c("scale", "center"),
               trControl= trainControl(method = "none"),
               tuneGrid = data.frame(degree=1, scale=1, C=1))
```

Cross Validation Model 2: CV Model Using training set of 2401 obs method = K fold cross validation 10 fold, ~240 obs per fold

```
Model2 <- train(target ~ ., data = df,
               method = "svmPoly",
               na.action = na.omit,
               preProcess = c("scale", "center"),
               trControl= trainControl(method = "cv", number = 10),
               tuneGrid = data.frame(degree=1, scale=1, C=1))
```

Applying prediction to models Applied to both testing and traing datasets and both models

```
Model.training <- predict(Model, df)
Model.testing <- predict(Model, fundraisingTest)
Model2 <- predict(Model2, df)
```

Applying model performance

```
Model.training.confusion <- confusionMatrix(Model.training, df$target)
Model.testing.confusion <- confusionMatrix(Model.testing,
fundraisingTest$target)
Model2.confusion <- confusionMatrix(Model2, df$target)
```

Creating confusion matrices for the two models to check their performance for selection.

```
print(Model.training.confusion)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction Donor No Donor
##   Donor      683      531
##   No Donor   517      670
##
##                Accuracy : 0.5635
##                  95% CI : (0.5434, 0.5835)
##     No Information Rate : 0.5002
##     P-Value [Acc > NIR] : 2.974e-10
##
##                   Kappa : 0.127
##
```

```
##  Mcnemar's Test P-Value : 0.688
##
##             Sensitivity : 0.5692
##             Specificity : 0.5579
##          Pos Pred Value : 0.5626
##          Neg Pred Value : 0.5644
##              Prevalence : 0.4998
##          Detection Rate : 0.2845
##    Detection Prevalence : 0.5056
##       Balanced Accuracy : 0.5635
##
##        'Positive' Class : Donor
##
```

```r
print(Model.testing.confusion)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Donor No Donor
##   Donor      170      149
##   No Donor   129      151
##
##                Accuracy : 0.5359
##                  95% CI : (0.495, 0.5764)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : 0.0469
##
##                   Kappa : 0.0719
##
##  Mcnemar's Test P-Value : 0.2545
##
##             Sensitivity : 0.5686
##             Specificity : 0.5033
##          Pos Pred Value : 0.5329
##          Neg Pred Value : 0.5393
##              Prevalence : 0.4992
##          Detection Rate : 0.2838
##    Detection Prevalence : 0.5326
##       Balanced Accuracy : 0.5359
##
##        'Positive' Class : Donor
##
```

```r
print(Model2.confusion)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction Donor No Donor
##   Donor      683      531
##   No Donor   517      670
##
##                Accuracy : 0.5635
##                  95% CI : (0.5434, 0.5835)
##     No Information Rate : 0.5002
##     P-Value [Acc > NIR] : 2.974e-10
##
##                   Kappa : 0.127
##
##  Mcnemar's Test P-Value : 0.688
##
##             Sensitivity : 0.5692
##             Specificity : 0.5579
##          Pos Pred Value : 0.5626
##          Neg Pred Value : 0.5644
##              Prevalence : 0.4998
##          Detection Rate : 0.2845
##    Detection Prevalence : 0.5056
##       Balanced Accuracy : 0.5635
##
##        'Positive' Class : Donor
##
```

As we can see from the results, the cross validation models accuracy performs at 56.35%. The testing dataset has an accuracy performance of 53.59%. However, accuracy alone is not always a good indicator of a good model for selection.

3. Classification under asymmetric conditions

- Why use weighted samples instead of a random sample? Weighted sampling was used to ensure the model would have almost the same number of donors as non-donors so one class was not given more "weight" than the other that could otherwise cause potential problems like skewness in the data.

```
futurefundraising = readRDS("C:/Users/razzb/OneDrive/Documents/UTSA Graduate
School Classes/Data Algorithms/Final Project/fundraising.rds")
summary(futurefundraising)

##  zipconvert2 zipconvert3 zipconvert4 zipconvert5 homeowner
##  No :2352    Yes: 551    No :2357    No :1846    Yes:2312
##  Yes: 648    No :2449    Yes: 643    Yes:1154    No : 688
##
##
##
##
##    num_child         income       female         wealth
##  Min.   :1.000   Min.   :1.000   Yes:1831   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:3.000   No :1169   1st Qu.:5.000
##  Median :1.000   Median :4.000              Median :8.000
##  Mean   :1.069   Mean   :3.899              Mean   :6.396
##  3rd Qu.:1.000   3rd Qu.:5.000              3rd Qu.:8.000
##  Max.   :5.000   Max.   :7.000              Max.   :9.000
##    home_value       med_fam_inc      avg_fam_inc       pct_lt15k
##  Min.   :   0.0   Min.   :   0.0   Min.   :   0.0   Min.   : 0.00
##  1st Qu.: 554.8   1st Qu.: 278.0   1st Qu.: 318.0   1st Qu.: 5.00
##  Median : 816.5   Median : 355.0   Median : 396.0   Median :12.00
##  Mean   :1143.3   Mean   : 388.4   Mean   : 432.3   Mean   :14.71
##  3rd Qu.:1341.2   3rd Qu.: 465.0   3rd Qu.: 516.0   3rd Qu.:21.00
##  Max.   :5945.0   Max.   :1500.0   Max.   :1331.0   Max.   :90.00
##     num_prom       lifetime_gifts    largest_gift       last_gift
##  Min.   : 11.00   Min.   :  15.0   Min.   :   5.00   Min.   :  0.00
##  1st Qu.: 29.00   1st Qu.:  45.0   1st Qu.:  10.00   1st Qu.:  7.00
##  Median : 48.00   Median :  81.0   Median :  15.00   Median : 10.00
##  Mean   : 49.14   Mean   : 110.7   Mean   :  16.65   Mean   : 13.48
##  3rd Qu.: 65.00   3rd Qu.: 135.0   3rd Qu.:  20.00   3rd Qu.: 16.00
##  Max.   :157.00   Max.   :5674.9   Max.   :1000.00   Max.   :219.00
##  months_since_donate    time_lag          avg_gift              target
##  Min.   :17.00        Min.   : 0.000   Min.   :  2.139   Donor   :1499
##  1st Qu.:29.00        1st Qu.: 3.000   1st Qu.:  6.333   No Donor:1501
##  Median :31.00        Median : 5.000   Median :  9.000
##  Mean   :31.13        Mean   : 6.876   Mean   : 10.669
##  3rd Qu.:34.00        3rd Qu.: 9.000   3rd Qu.: 12.800
##  Max.   :37.00        Max.   :77.000   Max.   :122.167
```

4. Evaluating the Fit of the model Let’s try other models to see if we can get better accuracy.

KNN (Nearest Neighbor Model)

```
#library(class)
#set.seed(12345)

#futureTrain = sample(120, 120)
#future.test.X = cbind(futurefundraising$num_child, futurefundraising$income,
futurefundraising$months_since_donate)[futureTrain,]

# KNN model with k = 10
#futureDonors = knn(fund.train.X, future.test.X, train.target, k = 10)
#futureDonors_value = as.character(futureDonors)
#futureDonors_value

#library(class)
#set.seed(12345)

#futureTrain = sample(120, 120)
#future.test.X = cbind(futurefundraising$num_child, futurefundraising$income,
futurefundraising$months_since_donate)[futureTrain,]

# KNN model with k = 100
#futureDonors = knn(fund.train.X, future.test.X, train.target, k = 100)
#futureDonors_value = as.character(futureDonors)
#futureDonors_value
```

```
#KNN k = 10
#write.table(futureDonors_value, file = "modelfund.csv", col.names =
c("value"), row.names = FALSE)

#KNN k = 100
#write.table(futureDonors_value, file = "fund_model.csv", col.names =
c("value"), row.names = FALSE)
```

5. Best Model After checking the scoreboard from uploading the csv files for the knn model k=10, the knn model with k=10 was determined to yield the best results in terms of overall accuracy. The final model was a KNN model with k =10 classification model was the most accuracte. Total accuracy was 57.5% The model performed a little better than the Support Vector and K-Fold Cross Validation Classification Models.

Research Methodology - Preferred method for projects research based such as this one is to have descriptions, explanations, and reasonings written in conjunction to the statistical analysis and model training on the same document to make it easier for readers and other researchers or data scientists to follow train of thought during the enitre model training process to ensure better replicability.

Model Performance - Although the KNN model was the most accuracte in this project, and many models were tested, not all models and algorithms were tested. There are many ways to train a model, and it is very likely that there is a model with even greater accuracy that was not explored yet. Further analysis would likely yield a more accurate model with further experimentation and testing on other models, including classification models. For the purposes of this project, the confusion matrix was used to explain the performance of the models tested.

Reccomendations - Further analysis of target population would most likely yield better results for more donations. Target populations should be focused on households with higher incomes and children and have a history of frequent or recent donations. Determining the rate of frequency of the average donor would significantly help determine how often to send out direct marketing donation campaigns. Also looking at the possibility of other marketing media trends, such as marketing to the target audience online, social media, TV, etc. may yield better response than direct mail marketing.