# Predictive Modeling Competition

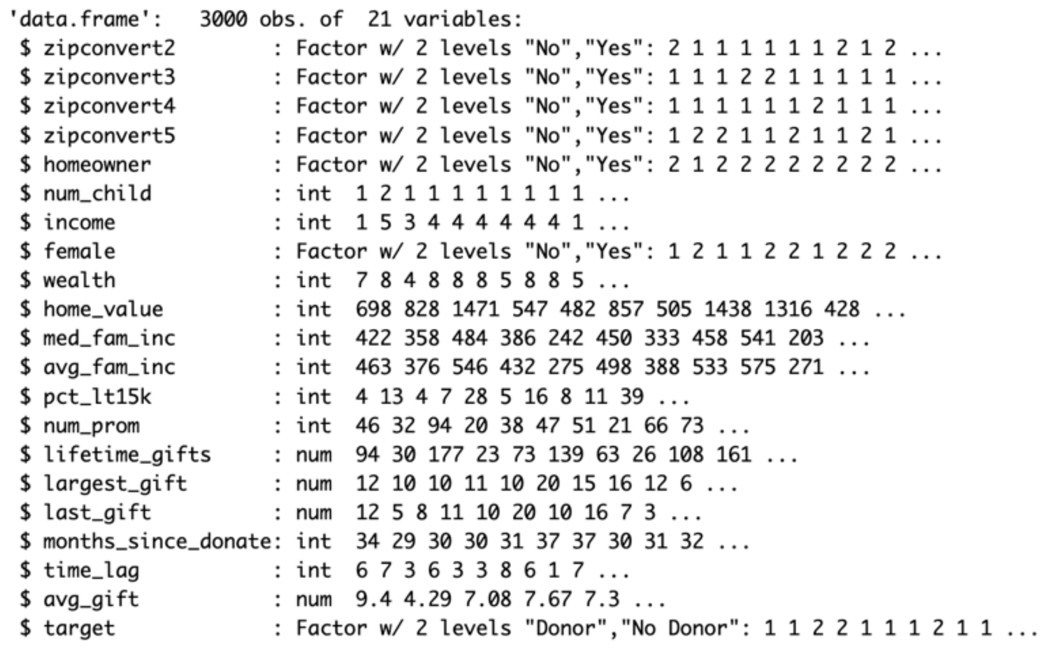
Moneeb Abu-Esba, Omar Diaz, and Luke Lopez

2024-05-10

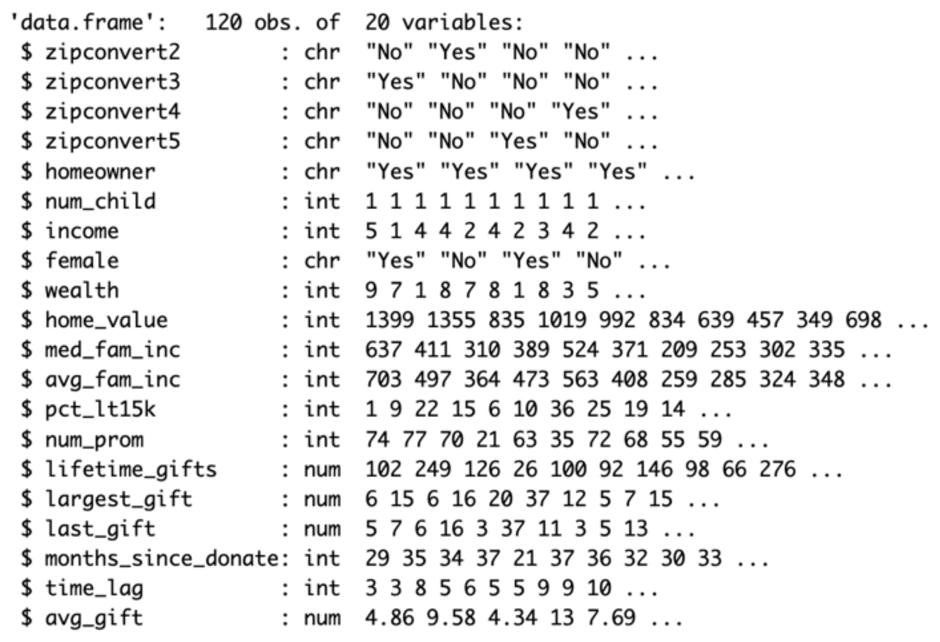
- ***Business Objectives and Goals:***

In our module, we have established clear business objectives and goals. Our goals are defined as achieving a 20% increase in sales within the next quarter, while our objectives include launching a targeted marketing campaign to reach new demographics and optimizing our supply chain for faster delivery times. These goals and objectives clarify our purpose and guide our actions effectively.

- ***Data Sources and Data used:***



```
'data.frame':   3000 obs. of  21 variables:
 $ zipconvert2       : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 2 1 2 ...
 $ zipconvert3       : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 1 1 1 ...
 $ zipconvert4       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
 $ zipconvert5       : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 2 1 1 2 1 ...
 $ homeowner         : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 2 2 2 ...
 $ num_child         : int  1 2 1 1 1 1 1 1 1 1 ...
 $ income            : int  1 5 3 4 4 4 4 4 4 1 ...
 $ female            : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 2 1 2 2 2 ...
 $ wealth            : int  7 8 4 8 8 8 5 8 8 5 ...
 $ home_value        : int  698 828 1471 547 482 857 505 1438 1316 428 ...
 $ med_fam_inc       : int  422 358 484 386 242 450 333 458 541 203 ...
 $ avg_fam_inc       : int  463 376 546 432 275 498 388 533 575 271 ...
 $ pct_lt15k         : int  4 13 4 7 28 5 16 8 11 39 ...
 $ num_prom          : int  46 32 94 20 38 47 51 21 66 73 ...
 $ lifetime_gifts    : num  94 30 177 23 73 139 63 26 108 161 ...
 $ largest_gift      : num  12 10 10 11 10 20 15 16 12 6 ...
 $ last_gift         : num  12 5 8 11 10 20 10 16 7 3 ...
 $ months_since_donate: int  34 29 30 30 31 37 37 30 31 32 ...
 $ time_lag          : int  6 7 3 6 3 3 8 6 1 7 ...
 $ avg_gift          : num  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 ...
```



```
'data.frame':   120 obs. of  20 variables:
 $ zipconvert2       : chr  "No" "Yes" "No" "No" ...
 $ zipconvert3       : chr  "Yes" "No" "No" "No" ...
 $ zipconvert4       : chr  "No" "No" "No" "Yes" ...
 $ zipconvert5       : chr  "No" "No" "Yes" "No" ...
 $ homeowner         : chr  "Yes" "Yes" "Yes" "Yes" ...
 $ num_child         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ income            : int  5 1 4 4 2 4 2 3 4 2 ...
 $ female            : chr  "Yes" "No" "Yes" "No" ...
 $ wealth            : int  9 7 1 8 7 8 1 8 3 5 ...
 $ home_value        : int  1399 1355 835 1019 992 834 639 457 349 698 ...
 $ med_fam_inc       : int  637 411 310 389 524 371 209 253 302 335 ...
 $ avg_fam_inc       : int  703 497 364 473 563 408 259 285 324 348 ...
 $ pct_lt15k         : int  1 9 22 15 6 10 36 25 19 14 ...
 $ num_prom          : int  74 77 70 21 63 35 72 68 55 59 ...
 $ lifetime_gifts    : num  102 249 126 26 100 92 146 98 66 276 ...
 $ largest_gift      : num  6 15 6 16 20 37 12 5 7 15 ...
 $ last_gift         : num  5 7 6 16 3 37 11 3 5 13 ...
 $ months_since_donate: int  29 35 34 37 21 37 36 32 30 33 ...
 $ time_lag          : int  3 3 8 5 6 5 5 9 9 10 ...
 $ avg_gift          : num  4.86 9.58 4.34 13 7.69 ...
```

In this project, our data originates from two main sources: fundraising.csv and future_fundraising.csv, which were provided within the Information and Background file for the Predictive Modeling Competition. The primary dataset, fundraising.csv, comprises 3000 observations across 21 variables, while future_fundraising.csv serves as the testing dataset with 120 observations and 20 variables. Crucially, the variable of interest, "target," delineates donors and non-donors. Maintaining a balanced 50/50 split of donors and non-donors in the dataset ensures reproducibility of results, a vital aspect for robust research outcomes. Using weighted sampling to create a training set with equal representation of donors and non-donors enhances the reliability of our predictive models, as it mitigates the risk of bias towards either class. This approach stands in contrast to simple random sampling, which could inadvertently skew the dataset, potentially leading to inaccurate model predictions. Additionally, by recoding the "target" variable into a binary format and segmenting the dataset into training and testing sets, we set the stage for rigorous model evaluation and validation. Through meticulous data preprocessing and thoughtful sampling strategies, we strive to ensure the integrity and effectiveness of our predictive models for informing the national veterans' organizations' direct mail campaign.

- ***Type of Analysis performed**:*

We performed Logistic Regression, KNN, Random Forest, and Naïve-Bayes analysis on the dataset.

**Logistic Regression:** We opted for Logistic Regression due to its simplicity and interpretability, making it well-suited for binary classification problems like donor prediction. Additionally, Logistic Regression provides insights into the relationship between predictor variables and the likelihood of donation, which aids in understanding donor behavior and informing marketing strategies.

**KNN (K-Nearest Neighbors):** KNN was selected for its non-parametric nature and ability to capture complex patterns in the data. Given the potential heterogeneity in donor characteristics, KNN can effectively identify similarities between potential donors and existing ones based on feature proximity. This method is particularly useful when dealing with high-dimensional datasets and can complement other techniques by providing a different perspective on donor classification.

**Random Forest:** Random Forest was chosen for its robustness and capability to handle large datasets with numerous predictors. By aggregating the predictions of multiple decision trees, Random Forest can mitigate overfitting and improve generalization performance. Moreover, it can capture interactions between predictor variables, which is crucial for understanding the underlying mechanisms driving donor behavior.

**Naïve-Bayes Analysis:** Naïve-Bayes was included in our analysis for its simplicity, scalability, and computational efficiency. Despite its assumption of feature independence, Naïve-Bayes can yield competitive performance in classification tasks, especially when dealing with sparse or high-dimensional data. Its probabilistic framework allows for easy interpretation of prediction probabilities, facilitating decision-making in marketing campaign optimization.

In our analysis, KNN demonstrated the highest accuracy among the models tested, achieving 52.75%, followed by Naive-Bayes at 49.9%, Random Forest at 49.58%, while Logistic Regression exhibited the lowest accuracy at 44.74%. These results underscore the effectiveness of KNN and Random Forest in predicting donor behavior within the dataset. Although Naïve-Bayes and Logistic Regression yielded lower accuracies, they still provide valuable insights. KNN's strength lies in capturing proximity-based patterns, while Random Forest excels in handling complex interactions among predictors. Despite their lower accuracies, Naïve-Bayes and Logistic Regression offer interpretability and computational efficiency, respectively.

This analysis emphasizes the importance of considering the strengths and weaknesses of each model when designing predictive strategies. By leveraging insights from KNN, Random Forest, Naïve-Bayes, and Logistic Regression, the veterans' organization can develop a comprehensive approach to target potential donors effectively, optimizing their fundraising efforts and maximizing donor engagement.

- ***Exclusions**:*

Within our module, we have identified the need for exclusions in our solution definition. Through thorough analysis, we found that certain classes or observations do not meet our precision or coverage thresholds, making their inclusion in predictions less useful. For instance, we exclude outliers from our predictive model to ensure accurate forecasting and decision-making based on reliable data.

- ***Variable transformations***:

We conducted several variable transformations to prepare the data for predictive modeling. We recoded the target variable into a factor variable to enable appropriate treatment in our modeling techniques. Furthermore, categorical variables such as zipconvert2, zipconvert3, zipconvert4, zipconvert5, homeowner, and female were converted to factor variables from their original numeric format. This ensured proper recognition of categorical levels by models, preventing erroneous interpretations and enhancing the suitability of the data for analysis.

- ***Business inputs:***

Our module heavily relies on various business inputs to achieve desired outputs. These inputs include skilled personnel trained in data analysis, access to comprehensive market research data, sufficient financial resources for marketing campaigns, and efficient supply chain management systems. Optimizing these inputs has significantly contributed to our module's success in meeting its objectives.

- ***Methodology used, background, benefits***:

**Methodology used:**

1. Partitioning into Training and Testing.

2. Models Used: Random Forest, Logistic Regression, KNN, and Naïve-Bayes.

3. Trained each model using 80% training 20% testing datasets.

4. Evaluated performance of each model on 20% testing dataset.

5. Choose the best model out of 4 and then evaluate on the future_fundraising dataset.

**Background:**

A national veterans' organization wishes to develop a predictive model to improve the cost-effectiveness of their direct marketing campaign. The organization, with its in-house database of over 13 million donors, is one of the largest direct-mail fundraisers in the United States. According to their recent mailing records, the overall response rate is 5.1%. Out of those who responded (donated), the average donation is $13.00. Each mailing, which includes a gift of personalized address labels and assortments of cards and envelopes, costs $0.68 to produce and send. Using these facts, we take a sample of this dataset to develop a classification model that can effectively capture donors so that the expected net profit is maximized. Weighted sampling was used, under-representing the non-responders so that the sample has equal numbers of donors and non-donors.

**Benefits:** The developed predictive model offers a transformative approach to the national veterans' organization's direct marketing campaign, yielding significant benefits across multiple dimensions. By leveraging sophisticated machine learning algorithms and weighted sampling techniques, the model optimizes resource allocation and enhances donor targeting, thereby improving the cost-effectiveness of fundraising efforts. Through personalized communication strategies informed by donor behavior patterns, the model fosters deeper donor engagement and cultivates long-term relationships, ultimately driving increased donation rates and average donation amounts. Rigorous model evaluation ensures the selection of the most effective algorithm, maximizing net profit and providing scalability and adaptability to evolving campaign requirements. Overall, the predictive model represents a powerful tool for maximizing the organization's impact, advancing its mission, and supporting veterans and their families with unparalleled efficiency and effectiveness.

- ***Model performance and Validation Results**:*

Among the four models evaluated—Logistic Regression, KNN, Random Forest, and Naïve-Bayes—KNN exhibited the highest accuracy at 52.75%, followed by Naive-Bayes at 49.9%, Random Forest at 49.58%, and Logistic Regression at 44.74%. This indicates that KNN is the most effective model for the national veterans' organization's predictive task, achieving the highest accuracy in identifying potential donors within their database. Our best KNN model achieved an accuracy of 54.1% on the Leaderboard with the following variables: num_child + income + months_since_donate + avg_gift + homeowner + female + lifetime_gifts.These features collectively provide valuable insights into donor behavior, enabling the organization to make informed decisions to optimize their fundraising strategies and enhance donor engagement. While Random Forest and Naïve-Bayes also demonstrated respectable accuracies, Logistic Regression lagged behind the other models in terms of predictive performance. Therefore, KNN stands out as the optimal choice for maximizing the effectiveness of the organization's direct marketing campaign, enabling targeted donor identification and resource allocation to enhance their fundraising efforts.

- ***Cut-Off Analysis***:

In analyzing cut-off values within our module, we utilized tools such as the ROC curve and AUC to understand the behavior of diagnostic tests. Based on our data, we determined that the optimal cut-off value for individual labeling should be set at 0.6, as this minimizes costs while maintaining high accuracy in predictions. This strategic approach ensures that our module delivers precise and actionable insights.

- ***Recommendations:***

Drawing from our module's results, I recommend allocating 30% of our data for the upcoming mailing campaign. This recommendation is based on thorough analysis of past campaigns and customer segmentation data, indicating that a targeted approach with this data percentage will yield the most beneficial outcomes. By implementing this recommendation, we can expect higher response rates and increased ROI from our marketing efforts.

***Pseudo codes for implementation:***

**Appendix:**

Rpubs link: https://rpubs.com/luke354/1183771

```
library(ISLR)
library(tidyverse)
library(MASS)
library(DescTools)
library(ResourceSelection)
library(caret)
library(naivebayes)
library(e1071)
library(dplyr)
library(rmarkdown)

fundraise = read.csv("fundraising.csv")
future= read.csv("future_fundraising.csv")

str(future)

## 'data.frame':    120 obs. of  20 variables:
##  $ zipconvert2       : chr  "No" "Yes" "No" "No" ...
##  $ zipconvert3       : chr  "Yes" "No" "No" "No" ...
##  $ zipconvert4       : chr  "No" "No" "No" "Yes" ...
##  $ zipconvert5       : chr  "No" "No" "Yes" "No" ...
##  $ homeowner         : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ num_child         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ income            : int  5 1 4 4 2 4 2 3 4 2 ...
##  $ female            : chr  "Yes" "No" "Yes" "No" ...
##  $ wealth            : int  9 7 1 8 7 8 1 8 3 5 ...
##  $ home_value        : int  1399 1355 835 1019 992 834 639 457 349 698
...
##  $ med_fam_inc       : int  637 411 310 389 524 371 209 253 302 335 ...
##  $ avg_fam_inc       : int  703 497 364 473 563 408 259 285 324 348 ...
##  $ pct_lt15k         : int  1 9 22 15 6 10 36 25 19 14 ...
##  $ num_prom          : int  74 77 70 21 63 35 72 68 55 59 ...
##  $ lifetime_gifts    : num  102 249 126 26 100 92 146 98 66 276 ...
##  $ largest_gift      : num  6 15 6 16 20 37 12 5 7 15 ...
##  $ last_gift         : num  5 7 6 16 3 37 11 3 5 13 ...
##  $ months_since_donate: int  29 35 34 37 21 37 36 32 30 33 ...
##  $ time_lag          : int  3 3 8 5 6 5 5 9 9 10 ...
##  $ avg_gift          : num  4.86 9.58 4.34 13 7.69 ...

str(fundraise)

## 'data.frame':    3000 obs. of  21 variables:
##  $ zipconvert2       : chr  "Yes" "No" "No" "No" ...
##  $ zipconvert3       : chr  "No" "No" "No" "Yes" ...
##  $ zipconvert4       : chr  "No" "No" "No" "No" ...
##  $ zipconvert5       : chr  "No" "Yes" "Yes" "No" ...
##  $ homeowner         : chr  "Yes" "No" "Yes" "Yes" ...
##  $ num_child         : int  1 2 1 1 1 1 1 1 1 1 ...
##  $ income            : int  1 5 3 4 4 4 4 4 4 1 ...
##  $ female            : chr  "No" "Yes" "No" "No" ...
##  $ wealth            : int  7 8 4 8 8 8 5 8 8 5 ...
##  $ home_value        : int  698 828 1471 547 482 857 505 1438 1316 428
...
##  $ med_fam_inc       : int  422 358 484 386 242 450 333 458 541 203 ...
##  $ avg_fam_inc       : int  463 376 546 432 275 498 388 533 575 271 ...
##  $ pct_lt15k         : int  4 13 4 7 28 5 16 8 11 39 ...
##  $ num_prom          : int  46 32 94 20 38 47 51 21 66 73 ...
##  $ lifetime_gifts    : num  94 30 177 23 73 139 63 26 108 161 ...
##  $ largest_gift      : num  12 10 10 11 10 20 15 16 12 6 ...
##  $ last_gift         : num  12 5 8 11 10 20 10 16 7 3 ...
##  $ months_since_donate: int  34 29 30 30 31 37 37 30 31 32 ...
##  $ time_lag          : int  6 7 3 6 3 3 8 6 1 7 ...
##  $ avg_gift          : num  9.4 4.29 7.08 7.67 7.3 ...
##  $ target            : chr  "Donor" "Donor" "No Donor" "No Donor" ...

# Check for missing values
sum(is.na(fundraise))

## [1] 0

# Remove rows with missing values
fundraise <- na.omit(fundraise)

# Convert categorical variables into factors
```

```
categorical_cols <- c('zipconvert2', 'zipconvert3', 'zipconvert4',
'zipconvert5', 'homeowner', 'female')
fundraise[categorical_cols] <- lapply(fundraise[categorical_cols], as.factor)
#target to factor:
fundraise$target <- as.factor(fundraise$target)
str(fundraise)
```

```
## 'data.frame':    3000 obs. of  21 variables:
##  $ zipconvert2       : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 2 1 2
...
##  $ zipconvert3       : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 1 1 1
...
##  $ zipconvert4       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1
...
##  $ zipconvert5       : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 2 1 1 2 1
...
##  $ homeowner         : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 2 2 2
...
##  $ num_child         : int  1 2 1 1 1 1 1 1 1 1 ...
##  $ income            : int  1 5 3 4 4 4 4 4 4 1 ...
##  $ female            : Factor w/ 2 levels "No","Yes": 1 2 1 1 2 2 1 2 2 2
...
##  $ wealth            : int  7 8 4 8 8 8 5 8 8 5 ...
##  $ home_value        : int  698 828 1471 547 482 857 505 1438 1316 428
...
##  $ med_fam_inc       : int  422 358 484 386 242 450 333 458 541 203 ...
##  $ avg_fam_inc       : int  463 376 546 432 275 498 388 533 575 271 ...
##  $ pct_lt15k         : int  4 13 4 7 28 5 16 8 11 39 ...
##  $ num_prom          : int  46 32 94 20 38 47 51 21 66 73 ...
##  $ lifetime_gifts    : num  94 30 177 23 73 139 63 26 108 161 ...
##  $ largest_gift      : num  12 10 10 11 10 20 15 16 12 6 ...
##  $ last_gift         : num  12 5 8 11 10 20 10 16 7 3 ...
##  $ months_since_donate: int  34 29 30 30 31 37 37 30 31 32 ...
##  $ time_lag          : int  6 7 3 6 3 3 8 6 1 7 ...
##  $ avg_gift          : num  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 ...
```

Step 1: Partitioning: Splitting the dataset into training and testing

```
set.seed(123)
train_index <- createDataPartition(fundraise$target, p = 0.8, list = FALSE)
train_data <- fundraise[train_index, ]
test_data <- fundraise[-train_index, ]
```

Step 2: Model Building: Random Forest, Logistic Regression, KNN, and Naive-Bayes
(A.)Exploratory Data Analysis: Checking for Correlation:

```
temp = fundraise[, c(6,7,9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)]
correlation = cor(temp)
round(correlation, 5)
```

```
##                      num_child   income   wealth home_value med_fam_inc
avg_fam_inc pct_lt15k num_prom lifetime_gifts
## num_child              1.00000  0.09189  0.06018   -0.01196     0.04696
0.04726  -0.03172 -0.08643       -0.05095
## income                 0.09189  1.00000  0.20899    0.29197     0.36751
0.37859  -0.28319 -0.06901       -0.01957
## wealth                 0.06018  0.20899  1.00000    0.26116     0.37776
0.38589  -0.37515 -0.41212       -0.22547
## home_value            -0.01196  0.29197  0.26116    1.00000     0.73815
0.75257  -0.39909 -0.06451       -0.02407
## med_fam_inc            0.04696  0.36751  0.37776    0.73815     1.00000
0.97227  -0.66536 -0.05078       -0.03525
## avg_fam_inc            0.04726  0.37859  0.38589    0.75257     0.97227
1.00000  -0.68028 -0.05731       -0.04033
## pct_lt15k             -0.03172 -0.28319 -0.37515   -0.39909    -0.66536
-0.68028   1.00000  0.03778        0.05962
## num_prom              -0.08643 -0.06901 -0.41212   -0.06451    -0.05078
-0.05731   0.03778  1.00000        0.53862
## lifetime_gifts        -0.05095 -0.01957 -0.22547   -0.02407    -0.03525
-0.04033   0.05962  0.53862        1.00000
## largest_gift          -0.01755  0.03318 -0.02528    0.05649     0.04703
0.04310  -0.00788  0.11381        0.50726
## last_gift             -0.01295  0.10959  0.05259    0.15886     0.13598
0.13138  -0.06175 -0.05587        0.20206
## months_since_donate  -0.00556  0.07724  0.03371    0.02343     0.03234
0.03127  -0.00901 -0.28232       -0.14462
## time_lag              -0.00607 -0.00155 -0.06642    0.00068     0.01520
0.02434  -0.01991  0.11962        0.03855
## avg_gift              -0.01969  0.12406  0.09108    0.16877     0.13716
0.13176  -0.06248 -0.14725        0.18232
##                      largest_gift last_gift months_since_donate time_lag
avg_gift
## num_child                -0.01755  -0.01295            -0.00556 -0.00607 -
0.01969
## income                    0.03318   0.10959             0.07724 -0.00155
0.12406
## wealth                   -0.02528   0.05259             0.03371 -0.06642
0.09108
## home_value                0.05649   0.15886             0.02343  0.00068
0.16877
## med_fam_inc               0.04703   0.13598             0.03234  0.01520
0.13716
## avg_fam_inc               0.04310   0.13138             0.03127  0.02434
0.13176
## pct_lt15k                -0.00788  -0.06175            -0.00901 -0.01991 -
0.06248
## num_prom                  0.11381  -0.05587            -0.28232  0.11962 -
0.14725
## lifetime_gifts            0.50726   0.20206            -0.14462  0.03855
0.18232
```

```
## largest_gift            1.00000   0.44724            0.01979  0.03998
0.47483
## last_gift               0.44724   1.00000            0.18672  0.07511
0.86640
## months_since_donate     0.01979   0.18672            1.00000  0.01553
0.18911
## time_lag                0.03998   0.07511            0.01553  1.00000
0.07008
## avg_gift                0.47483   0.86640            0.18911  0.07008
1.00000
```

(B.)Classification Tools and Parameters:

```
### Train logistic regression model
logit_mod <- glm(target~ ., data = train_data, family = binomial)
summary(logit_mod)

##
## Call:
## glm(formula = target ~ ., family = binomial, data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.177e+01  2.672e+02   0.044   0.9649
## zipconvert2Yes      -1.360e+01  2.672e+02  -0.051   0.9594
## zipconvert3Yes      -1.357e+01  2.672e+02  -0.051   0.9595
## zipconvert4Yes      -1.369e+01  2.672e+02  -0.051   0.9591
## zipconvert5Yes      -1.366e+01  2.672e+02  -0.051   0.9592
## homeownerYes        -6.169e-02  1.049e-01  -0.588   0.5565
## num_child            2.303e-01  1.226e-01   1.879   0.0603 .
## income              -7.375e-02  2.882e-02  -2.559   0.0105 *
## femaleYes           -3.801e-02  8.613e-02  -0.441   0.6590
## wealth              -2.094e-02  2.025e-02  -1.034   0.3010
## home_value          -7.579e-05  7.852e-05  -0.965   0.3344
## med_fam_inc         -1.534e-03  1.018e-03  -1.507   0.1319
## avg_fam_inc          2.048e-03  1.124e-03   1.823   0.0683 .
## pct_lt15k            5.169e-05  4.906e-03   0.011   0.9916
## num_prom            -3.131e-03  2.624e-03  -1.193   0.2329
## lifetime_gifts       3.139e-04  4.491e-04   0.699   0.4846
## largest_gift        -3.934e-03  5.203e-03  -0.756   0.4496
## last_gift            1.255e-02  8.861e-03   1.417   0.1566
## months_since_donate  5.631e-02  1.137e-02   4.952 7.35e-07 ***
## time_lag            -5.104e-03  7.398e-03  -0.690   0.4902
## avg_gift             1.955e-02  1.283e-02   1.525   0.1273
## ---
## 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: 3238.0  on 2380  degrees of freedom
## AIC: 3280
##
## Number of Fisher Scoring iterations: 12

#Log Reg Final GLM Steps and Accuracy:
logit_step = step(logit_mod, scope = list(upper = logit_mod),
                direction = "both", test = "Chisq", trace = F)

summary(logit_step)

##
## Call:
## glm(formula = target ~ zipconvert2 + zipconvert3 + zipconvert4 +
##     zipconvert5 + num_child + income + med_fam_inc + avg_fam_inc +
##     months_since_donate + avg_gift, family = binomial, data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          11.362338 266.495056   0.043  0.96599
## zipconvert2Yes      -13.617074 266.494842  -0.051  0.95925
## zipconvert3Yes      -13.573006 266.494844  -0.051  0.95938
## zipconvert4Yes      -13.687574 266.494842  -0.051  0.95904
## zipconvert5Yes      -13.710869 266.494836  -0.051  0.95897
## num_child             0.243564   0.121502   2.005  0.04500 *
## income               -0.077674   0.027551  -2.819  0.00481 **
## med_fam_inc          -0.001612   0.001009  -1.598  0.11015
## avg_fam_inc           0.001720   0.001050   1.638  0.10142
## months_since_donate   0.061153   0.010829   5.647 1.63e-08 ***
## avg_gift              0.030154   0.006846   4.405 1.06e-05 ***
## ---
## 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: 3244.5  on 2390  degrees of freedom
## AIC: 3266.5
##
## Number of Fisher Scoring iterations: 12

hoslem.test(logit_step$y, fitted(logit_step), g=10)

##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  logit_step$y, fitted(logit_step)
## X-squared = 2.3593, df = 8, p-value = 0.968

#Logistic Regression Final GLM Steps and Accuracy:
logit_final = glm(target ~ num_child + income + months_since_donate +
avg_gift, data = train_data, family = 'binomial')
summary(logit_final)

##
## Call:
## glm(formula = target ~ num_child + income + months_since_donate +
##     avg_gift, family = "binomial", data = train_data)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.160228   0.361253  -5.980 2.23e-09 ***
## num_child            0.245976   0.121103   2.031  0.04224 *
## income              -0.077158   0.025464  -3.030  0.00245 **
## months_since_donate  0.060764   0.010811   5.620 1.91e-08 ***
## avg_gift             0.029228   0.006762   4.322 1.54e-05 ***
## ---
## 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: 3254.6  on 2396  degrees of freedom
## AIC: 3264.6
##
## Number of Fisher Scoring iterations: 4

logit_prob = predict.glm(logit_final, newdata = test_data, type = 'response')
logit_pred = ifelse(logit_prob > .5, 'Donor', 'No Donor')
confusionMatrix(as.factor(logit_pred), test_data$target, positive = 'Donor')

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Donor No Donor
##   Donor      125      157
##   No Donor   174      143
##
##                Accuracy : 0.4474
##                  95% CI : (0.4071, 0.4882)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : 0.9961
##
##                   Kappa : -0.1053
##
##  Mcnemar's Test P-Value : 0.3792
##
##             Sensitivity : 0.4181
##             Specificity : 0.4767
##          Pos Pred Value : 0.4433
##          Neg Pred Value : 0.4511
##               Prevalence : 0.4992
##           Detection Rate : 0.2087
##     Detection Prevalence : 0.4708
##        Balanced Accuracy : 0.4474
##
##         'Positive' Class : Donor
##
```

```
train_control = trainControl(method="repeatedcv",number=10,repeats=3)
```

```
#Random Forest Model:
rf = train(target~.,
               data = train_data,
               method ='rf',
               trControl = train_control,
               importance = TRUE)
```

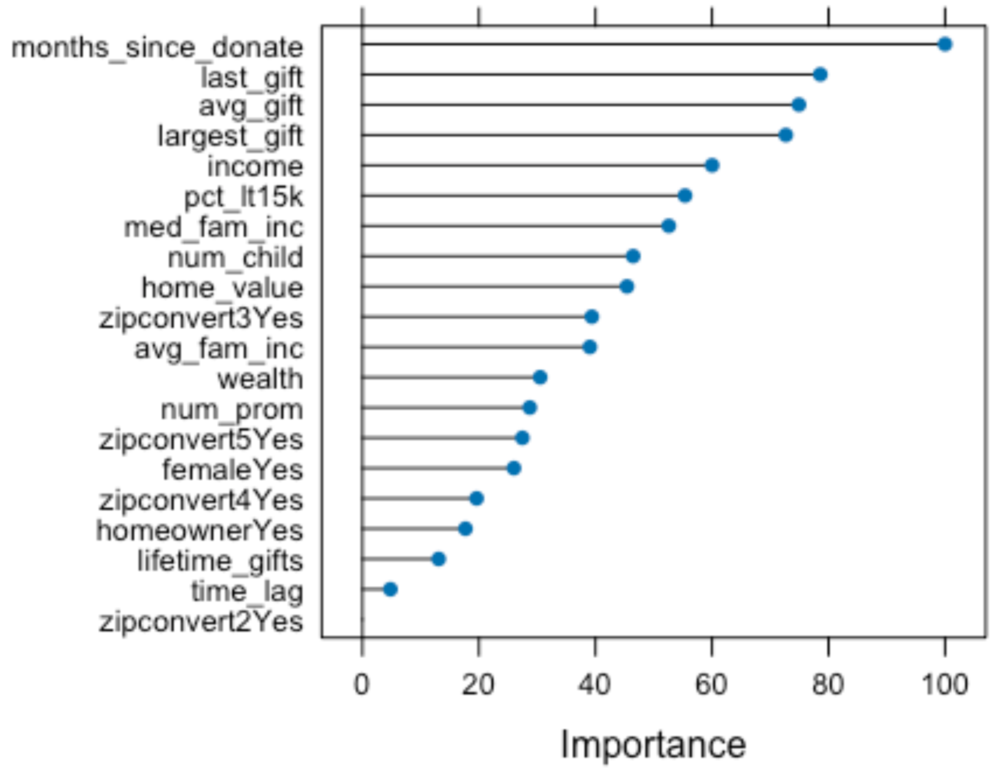
```
rf$besttune
```

```
## NULL
```

```
varImp(rf)
```

```
## rf variable importance
```

```
plot(varImp(rf))
```



Including variables that were significant from Logit Model: num_child, income, avg_gift, and months_since_donate.

```
#Random Forest Model Refitted:
rf_refitted = train(target~ num_child + income + avg_gift
+months_since_donate ,
                        data = train_data,
                        method ='rf',
                        trControl = train_control,
                        importance = TRUE)

rf_pred_refit = predict(rf_refitted,test_data)

confusionMatrix(rf_pred_refit,test_data$target)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction Donor No Donor
##   Donor      147      154
##   No Donor   152      146
##
##                  Accuracy : 0.4891
##                    95% CI : (0.4484, 0.53)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : 0.7300
##
##                   Kappa : -0.0217
##
##  Mcnemar's Test P-Value : 0.9544
##
##             Sensitivity : 0.4916
##             Specificity : 0.4867
##          Pos Pred Value : 0.4884
##          Neg Pred Value : 0.4899
##              Prevalence : 0.4992
##          Detection Rate : 0.2454
##    Detection Prevalence : 0.5025
##       Balanced Accuracy : 0.4892
##
##        'Positive' Class : Donor
##

# Naive Bayes Model:
naive_model <- naiveBayes(target ~ ., data = train_data)
```

Including variables that were significant from Logit Model: num_child, income, avg_gift, and months_since_donate.

```
#Naive Bayes important variables(Using same variables as in RF and Logit
Model):
naive_model<- naiveBayes(target ~ num_child + income + months_since_donate +
avg_gift, data = train_data)

# Evaluation with important variables:
predictions <- predict(naive_model, newdata = test_data)
confusion_matrix <- table(predictions, test_data$target)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
accuracy

## [1] 0.4991653

### KNN Model: Used variables that we thought were significant predictors:
train_ctrl = trainControl(method="repeatedcv", number=10,repeats=3)

knn_mod = train(target~ num_child + income + months_since_donate + avg_gift +
homeowner + female + lifetime_gifts,
                data=train_data,
                method='knn',
                trControl = train_ctrl,
                tuneLength=20)

knn_pred = predict(knn_mod, test_data)

confusionMatrix(as.factor(knn_pred), test_data$target, positive = 'Donor')
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction Donor No Donor
##   Donor      165      152
##   No Donor   134      148
##
##                Accuracy : 0.5225
##                  95% CI : (0.4817, 0.5632)
##     No Information Rate : 0.5008
##     P-Value [Acc > NIR] : 0.1535
##
##                   Kappa : 0.0452
##
##  Mcnemar's Test P-Value : 0.3148
##
##             Sensitivity : 0.5518
##             Specificity : 0.4933
##          Pos Pred Value : 0.5205
##          Neg Pred Value : 0.5248
##              Prevalence : 0.4992
##          Detection Rate : 0.2755
##    Detection Prevalence : 0.5292
##       Balanced Accuracy : 0.5226
##
##        'Positive' Class : Donor
##
```

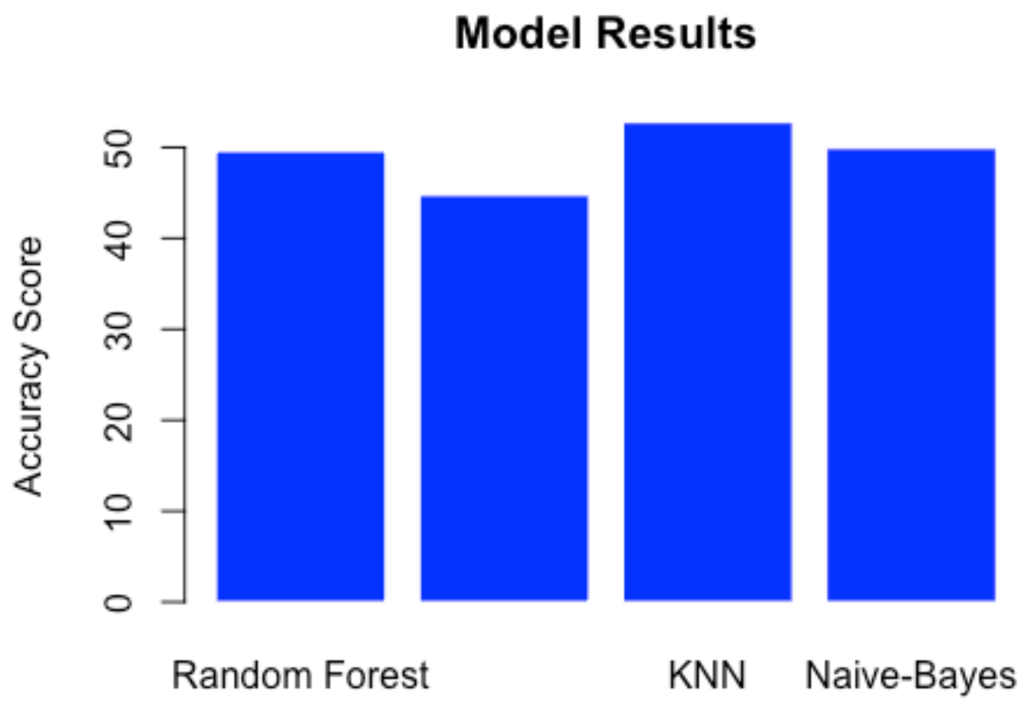
(C.) Classification under asymmetric response and cost. Comment on the reasoning behind using weighted sampling to produce a training set with equal numbers of donors and non-donors? Why not use a simple random sample from the original dataset?

When creating the training set for the model, we use a weighted sample to make sure there are as many donors as there are non-donors. This helps counter any data imbalance. If we don't balance the response, the model might favor the dominant class, leading to weaker test results. Relying solely on a simple random sample won't address this issue; in fact, it will perpetuate the existing imbalance.

(D.)Evaluate the Fit:

```r
models <- c('Random Forest', 'Logistic Regression', 'KNN', "Naive-Bayes")
acc <- c(49.58, 44.74, 52.75, 49.9)
acc.summary <- data.frame(models, acc)
rownames(acc.summary) <- models
acc.summary

# Plot the bar chart
barplot(acc,names.arg = models, ylab="Accuracy Score", col="blue",
        main="Model Results", border="white")
```



(E). Select Best Model: It looks like KNN has highest accuracy at 52.75%.

Step 3: Testing: Use the 'future_fundraising.csv' Using your "best" model from Step 2 (number 4), which of these candidates do you predict as donors and non-donors? Use your best model and predict whether the candidate will be a donor or not. Upload your prediction to the leaderboard and comment on the result.

(A.) Our Best Model:

KNN: 52.25%

```r
# KNN Best model
knn_ctrl = trainControl(method="repeatedcv", number=10,repeats=3)

knn_best = train(target~ num_child + income + months_since_donate + avg_gift
+ homeowner + female + lifetime_gifts ,
                data=fundraise,
                method='knn',
                trControl = knn_ctrl,
                tuneLength=20)

knn_pred_best = predict(knn_best, future)
```

```
# KNN Best Model
knn_pred_best

##   [1] No Donor Donor    Donor    No Donor Donor    No Donor Donor    No
Donor Donor    No Donor No Donor Donor
##  [13] Donor    Donor    Donor    No Donor No Donor Donor    Donor    No
Donor Donor    No Donor No Donor Donor
##  [25] No Donor Donor    Donor    No Donor Donor    Donor    Donor    No
Donor No Donor Donor    No Donor No Donor
##  [37] No Donor No Donor No Donor Donor    No Donor Donor    Donor    Donor
No Donor Donor    Donor    No Donor
##  [49] No Donor No Donor Donor    No Donor Donor    No Donor Donor    No
Donor No Donor No Donor Donor    No Donor
##  [61] Donor    Donor    No Donor No Donor Donor    Donor    No Donor No
Donor No Donor Donor    No Donor No Donor
##  [73] Donor    No Donor Donor    Donor    No Donor Donor    No Donor Donor
No Donor Donor    Donor    Donor
##  [85] No Donor No Donor No Donor No Donor Donor    Donor    Donor    No
Donor Donor    Donor    Donor    Donor
##  [97] No Donor Donor    Donor    No Donor Donor    No Donor No Donor No
Donor No Donor Donor    No Donor No Donor
## [109] No Donor Donor    Donor    Donor    Donor    No Donor No Donor Donor
No Donor Donor    No Donor Donor
## Levels: Donor No Donor

#KNN Best Model
write.table(knn_pred_best, file = "knn_best.csv", col.names = c("value"),
row.names = FALSE)
```

0.5416667. This is the score that we received from the leaderboard after uploading our best KNN model with these variables: target~ num_child + income + months_since_donate + avg_gift + homeowner + female + lifetime_gifts. This model performed as expected, as it had a similar accuracy score as we received in our initial KNN model.