Case Study competition

Thomas Farrell

3/23/2022

Contents

Cover Page

Report by:Thomas Farrell, Leo Salazar, Mia Brito, Raneen Aljamal, Adrian Joshua

Primary Contact : Thomas Farrell (512) 400-9580

Executive Summary

This data set is collected directly from a fundraising agency and contains 95,412 records broken into 142 fields. The models shown related the efficiency of the different variables to predict if a consumer will donate to the campaign and if they donate it will predict how much they are likely to donate. We will use three different modeling techniques Linear Regression, Linear discriminant analysis(LDA),Decision Tree modeling, and random forest techniques.

The main question to be answered is what type of individual will most likely donate? | The donation can be monitored by factors such as neighborhood code, Their max donation, and the time of their last donation. In the data a donation is represented by the binary number 1. ## Related Literature:

When it comes to modeling approaches, it is crucial to pick the method that best satisfies your data's needs. My team members and I have dove into the different types of modeling approaches to explore which method best fits our research. In our past case studies, we first explored logistic regression. As previously mentioned, Mitchell Dayton explains how logistic regression tackles the process of classifying problems. We also explored the use of a linear regression method. We learned that linear regression more commonly handles regression problems. To further enhance our knowledge of the different types of modeling approaches, we will be exploring the use of LDA and Decision Tree models in this case study.

Linear Discriminant Analysis, LDA, is another common modeling approach. LDA is commonly used to find a linear combination to separate more than two subjects or events. In Ricciardi, Valente, Edmund, Cantoni, Green, Fiorillo, Picone, Santini, and Cesarelli's work, they used the LDA model to explore the use of a data mining technique to help clinicians in decision-making when it comes to Coronary Artery Disease. As previously mentioned, LDA models are used to separate two or more subjects. In this case study, they used the LDA model to classify the patients to help aid the decision making process. They concluded that this accelerated and enhanced the decision making process for doctors to choose the correct treatment for their patient.

Similar to an LDA model, a Decision Tree model is used for classification purposes. A Decision Tree model builds regression by creating a model in a tree-like structure. It essentially makes the data set smaller by breaking it down into smaller subsets for an easier and quicker read. Stretch, Soares, and Moreira explored the use of a Decision Tree model when predicting the failure of students in University courses. They used the Decision Tree model to narrow down their data and make it smaller and were able to successfully explain the drop out phenomenon.

Data Anlaysis (Methodology)

Once we defined our variables in the data set, we changed all of our character variables to factor variables. We then created graphic models of significant variables in our data. This allowed us to visually see what people in our data could be beneficial as we could see which demographics may donate. When making our graphic representations we had to include Donation as the fill for the bar chart or histogram. For the Categorical variables we decided to use a Bar chart to best represent our data. The fill of the "Donation" variable allowed us to show the relationship between the variable and if they would choose to make a Donation or not.

We had limitations in the data as some of the variables in the data were highly correlated to Donation. So we had to remove these variables from our main data set. To find these highly correlated variables we ran the findcorrelation function, and set the cutoff to .9. We then ran a logistical regression and found the GVIF of each variable and ran another logistic regression with only the significant variables.. The rule that we used to determine multicollinearity was if the GVIF was over 5, the variable contained multicollinearity. We then removed the variables with multiculinqrity we then used the function ovum.sample which can balance overfitted or under-fitted data.

When we achieved our balanced data set we then performed a training and testing split on the data so we could create our models with the testing data.Once we had our data down to the correct variables we were able to see that an optimal cutoff needed to be implemented. We experimented with .1 as the cutoff first and our sensitivity was at .0102 which was telling us that our model was not good at predicting the "yes" responses. We then changed our optimal cutoff to .51 which changed the sensitivity to .588 which gave us a better model.We created an LDA model in our case study which is a modeling technique that reduces the number of variables in our data set while keeping as much information as possible to find a linear combination to predict our categories. The LDA model gave us a sensitivity of .6393 and a specificity of .5276 which showed us that our model was more accurate at predicting "yes" responses. We also observed that our data for this model had an accuracy of .5838 which was slightly higher than that of the logistic regression model. Due to the limitation of not being able to implement an optimal cutoff for the LDA model the accuracy of this model could be seen as false.

Appendix

Loading in libraries

```r
library(varImp)
library(caret)
library(lattice)
library(ggplot2)
library(gam)
library(caret)
library(ROCR)
library(ggmosaic)
library(prettydoc)
library(tinytex)
library(corrplot)
library(MASS)
library(tidyverse)
library(vip)
library(esquisse)
library(lubridate)
library(modeldata)
library(leaps)
library(ROSE)
library(randomForest)
source('/Users/thomasfarrell/Downloads/optim_threshold.R')
```

(Brownlee 2020)(Ricciardi et al. 2020)(n.d.a)(n.d.b)

Reading in data

```r
b1 = read.csv("/Users/thomasfarrell/Downloads/Customer_Analytics_TrainTest.csv", sep = ",")
```

Changing character variables to factors

```r
b1$State = as.factor(b1$State)
b1$NeighborhoodCode = as.factor(b1$NeighborhoodCode)
b1$Gender = as.factor(b1$Gender)
b1$Donation = as.factor(b1$Donation)
b1 = subset(b1, !is.na(b1$Age))
b1 = subset(b1, !is.na(b1$NumberOfChild))
b1 = subset(b1, !is.na(b1$Income))
b1 = dplyr::select(b1, - PercMale)
b1 = dplyr::select(b1, - PercFemale)
b1 = dplyr::select(b1, - PercWhite)
b1 = dplyr::select(b1, - PercIsleAsian)
b1 = dplyr::select(b1, - AgeC1)
b1 = dplyr::select(b1, - AgeC2)
b1 = dplyr::select(b1, - AgeC3)
```

```r
b1 = dplyr::select(b1, - AgeC4)
b1 = dplyr::select(b1, - AgeC5)
b1 = dplyr::select(b1, - AgeC6)
b1 = dplyr::select(b1, - AgeC7)
b1 = dplyr::select(b1, - HHN1)
b1 = dplyr::select(b1, - HHN2)
b1 = dplyr::select(b1, - HHN4)
b1 = dplyr::select(b1, - PercMarr)
b1 = dplyr::select(b1, - PercSingle)
b1 = dplyr::select(b1, - MedHomeVal)
b1 = dplyr::select(b1, - MedFamInc)
b1 = dplyr::select(b1, - PerCapInc)
b1 = dplyr::select(b1, - IC2H)
b1 = dplyr::select(b1, - IC1F)
b1 = dplyr::select(b1, - IC2F)
b1 = dplyr::select(b1, - IC3F)
b1 = dplyr::select(b1, - IC4F)
b1 = dplyr::select(b1, - IC5F)
b1 = dplyr::select(b1, - IC6F)
b1 = dplyr::select(b1, - IC9F)
b1 = dplyr::select(b1, - EMP3)
b1 = dplyr::select(b1, - EMP4)
b1 = dplyr::select(b1, - EMP8)
b1 = dplyr::select(b1, - EMP9)
b1 = dplyr::select(b1, - EMP13)
b1 = dplyr::select(b1, - EMP14)
b1 = dplyr::select(b1, - Edu1)
b1 = dplyr::select(b1, - Edu2)
b1 = dplyr::select(b1, - Edu3)
b1 = dplyr::select(b1, - Edu4)
b1 = dplyr::select(b1, - Edu5)
b1 = dplyr::select(b1, - Edu6)
b1 = dplyr::select(b1, - Edu7)
b1 = dplyr::select(b1, - PercForgnBr)
b1 = dplyr::select(b1, - LangEng)
b1 = dplyr::select(b1, - LangSpa)
b1 = dplyr::select(b1, - LangAs)
b1 = dplyr::select(b1, - LangOth)
b1 = dplyr::select(b1, - AmDonated)
b1 = dplyr::select(b1, - NumberOfChild)
b1 = dplyr::select(b1, - PercEmpState)


str(b1)
```

```
## ’data.frame':    11359 obs. of  94 variables:
##  $ DateOfFirstGift : int  9401 8601 8701 8901 8801 9501 9101 9601 9101 9101 ...
##  $ State           : Factor w/ 57 levels "AA","AE","AK",..: 9 14 21 29 20 29 9 20 49 20 ...
##  $ Zip             : int  91326 33176 46755 56475 62376 55044 -93527 60091 77381 62249 ...
##  $ DOB             : int  5202 2001 6001 2603 5201 5201 4307 5601 4809 6001 ...
##  $ NeighborhoodCode: Factor w/ 17 levels " ","C1","C2",..: 8 9 12 7 6 11 11 8 11 12 ...
##  $ Age             : int  46 78 38 72 46 46 54 42 49 38 ...
##  $ Home            : int  1 1 1 1 0 1 1 1 1 0 ...
##  $ Income          : int  6 3 4 4 7 7 7 7 7 4 ...
##  $ Gender          : Factor w/ 7 levels " ","A","C","F",..: 6 4 4 4 4 4 6 4 4 4 ...
##  $ PercVets        : int  15 28 33 30 30 35 48 32 28 33 ...
##  $ PercEmpLoc      : int  6 26 7 9 4 3 8 8 7 6 ...
##  $ PercEmpFed      : int  1 2 1 6 3 6 39 4 0 2 ...
##  $ Persons         : int  3611 2520 1067 1435 4429 1134 3142 1666 1200 2196 ...
##  $ Fams            : int  940 627 245 384 1257 305 884 523 331 542 ...
##  $ Households      : int  998 761 348 483 1593 366 1237 604 377 792 ...
##  $ PercPopUrban    : int  99 99 0 0 0 99 0 99 0 0 ...
##  $ PercPopRural    : int  0 0 0 99 99 0 99 0 0 3 ...
##  $ PercBlack       : int  0 98 0 0 0 0 1 0 2 0 ...
##  $ PercNatAm       : int  0 0 0 0 0 0 2 0 0 0 ...
##  $ PercHisp        : int  6 1 1 0 0 0 6 2 5 1 ...
##  $ PercAsianInd    : int  4 0 0 0 0 0 0 0 1 0 ...
##  $ PercJapa        : int  2 0 0 0 0 0 0 3 0 0 ...
##  $ PercChin        : int  6 0 0 0 0 0 0 2 0 0 ...
##  $ PercPhil        : int  4 0 0 0 0 0 0 0 0 0 ...
##  $ PercKor         : int  14 0 0 0 0 1 0 4 0 0 ...
##  $ PercViet        : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ PercHaw         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PercMex         : int  2 0 1 0 0 0 5 1 3 1 ...
##  $ PercPurRic      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PercCuba        : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ MedAge          : int  34 33 35 31 35 31 36 42 30 38 ...
##  $ AvgAge          : int  32 36 40 34 37 31 37 40 29 41 ...
##  $ ChildC1         : int  11 13 15 13 14 15 14 12 13 14 ...
##  $ ChildC2         : int  16 15 14 16 16 13 16 17 16 15 ...
##  $ ChildC3         : int  36 34 35 34 33 31 34 27 35 36 ...
##  $ PercElder       : int  12 31 24 27 28 10 25 28 7 32 ...
##  $ HHN3            : int  75 57 45 49 46 64 38 51 67 40 ...
##  $ HHN5            : int  23 24 11 18 12 14 10 8 15 10 ...
##  $ HHN6            : int  9 14 4 7 3 4 4 0 3 3 ...
##  $ PercSepDiv      : int  4 17 8 6 5 8 13 5 7 7 ...
##  $ PercWid         : int  3 9 18 6 9 2 6 6 2 18 ...
##  $ AvgHomeVal      : int  5218 594 519 428 377 1259 862 2628 791 677 ...
##  $ MedRent         : int  12 4 3 2 2 6 3 11 6 2 ...
##  $ AvgRent         : int  10 3 3 1 1 4 3 9 6 2 ...
##  $ MedHousInc      : int  1088 240 251 225 245 566 313 711 538 246 ...
##  $ AvgHousInc      : int  1026 293 278 267 280 595 367 827 575 335 ...
##  $ AvgFamInc       : int  1037 321 311 305 313 627 401 895 588 444 ...
##  $ IC1H            : int  2 24 18 31 27 2 24 3 8 30 ...
##  $ IC3H            : int  2 23 25 21 20 19 11 7 8 13 ...
##  $ IC4H            : int  5 13 20 15 18 22 17 8 24 18 ...
##  $ IC5H            : int  15 4 4 8 8 31 21 27 36 10 ...
##  $ IC6H            : int  14 4 1 1 1 15 6 12 15 2 ...
##  $ IC7H            : int  13 0 0 1 1 2 1 10 8 4 ...
##  $ IC8H            : int  10 0 0 0 0 4 1 6 0 1 ...
##  $ IC9H            : int  33 2 0 0 0 2 0 19 0 1 ...
##  $ IC7F            : int  14 0 0 1 1 3 2 9 7 7 ...
##  $ IC8F            : int  10 0 0 0 0 4 1 7 0 2 ...
##  $ PercSocSec      : int  6 35 31 34 33 7 24 21 10 33 ...
##  $ LFA             : int  70 61 54 65 65 84 62 68 80 58 ...
##  $ LFAM            : int  83 73 65 72 74 88 70 85 93 68 ...
##  $ LFAF            : int  58 51 46 58 56 78 55 54 69 50 ...
##  $ LFAMEmp        : int  81 65 61 68 70 81 68 83 91 66 ...
##  $ LFAFEmp        : int  57 49 45 53 54 73 54 53 67 47 ...
##  $ PercProf       : int  22 17 6 9 9 15 14 27 22 12 ...
##  $ PercManage     : int  24 8 8 6 7 17 13 26 24 15 ...
##  $ PercTech       : int  4 2 3 2 3 1 5 3 8 5 ...
##  $ PercSales      : int  21 6 14 9 6 12 9 19 24 14 ...
##  $ PercCler       : int  13 15 19 12 13 19 16 12 14 12 ...
##  $ PercFarm       : int  0 2 0 17 13 3 1 0 0 0 ...
##  $ PercCraft      : int  4 9 15 15 12 11 16 5 5 9 ...
##  $ PercMech       : int  1 0 19 8 9 1 3 1 0 13 ...
##  $ PercTran       : int  0 7 3 6 8 2 4 1 1 6 ...
##  $ PercLab        : int  3 2 1 4 5 4 3 0 0 1 ...
##  $ EMP1           : int  1 2 0 18 15 1 1 0 0 0 ...
##  $ EMP2           : int  0 0 0 0 0 0 0 0 4 0 ...
##  $ EMP5           : int  1 5 0 4 7 14 4 6 3 4 ...
##  $ EMP6           : int  2 2 1 1 1 3 2 0 0 3 ...
##  $ EMP7           : int  8 2 9 2 3 5 2 2 10 4 ...
##  $ EMP10          : int  4 7 3 4 3 5 10 7 3 4 ...
##  $ EMP11          : int  3 6 0 2 3 0 2 2 1 3 ...
##  $ EMP12          : int  4 4 3 1 1 3 1 1 0 1 ...
##  $ EMP15          : int  11 4 4 4 5 8 9 12 6 6 ...
##  $ EMP16          : int  1 3 0 7 1 1 26 3 1 2 ...
##  $ MedEdu         : int  6 26 10 9 4 3 10 8 7 2 ...
##  $ PercStateBr    : int  39 65 77 89 88 73 49 65 44 80 ...
##  $ LifeGiftDol    : num  47 254 107 48 100 36 107 50 102 57 ...
##  $ LifeGiftNum    : int  3 37 14 9 4 4 6 1 12 7 ...
##  $ LifeGiftPromNum : int  1 8 8 5 1 2 4 1 6 3 ...
##  $ MinDol         : num  10 3 3 4 15 5 10 50 5 5 ...
##  $ MaxDol         : num  25 15 12 7 45 11 30 50 13 10 ...
##  $ LastDol        : num  25 15 11 6 45 10 15 50 10 9 ...
##  $ AvgDol         : num  15.67 6.86 7.64 5.33 25 ...
##  $ CustID         : int  148535 7112 62117 85548 98090 82229 160963 89160 122772 97870 ...
##  $ Donation       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Removing highly correlated variables

```
b1_num = dplyr::select_if(b1, is.numeric)
M = cor(b1_num)
highcorr = findCorrelation(M, cutoff = .9, names = TRUE)
b1 = dplyr::select(b1, - highcorr)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(highcorr)` instead of `highcorr` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

Checking summary of target variable

```
summary(b1$Donation)
```

```
##     0     1
## 10838   521
```

Balancing the data

```
balancedb1 = ovun.sample(Donation~.,data= b1, seed=123, method= "over")$data
```

```
table(balancedb1$Donation)
```

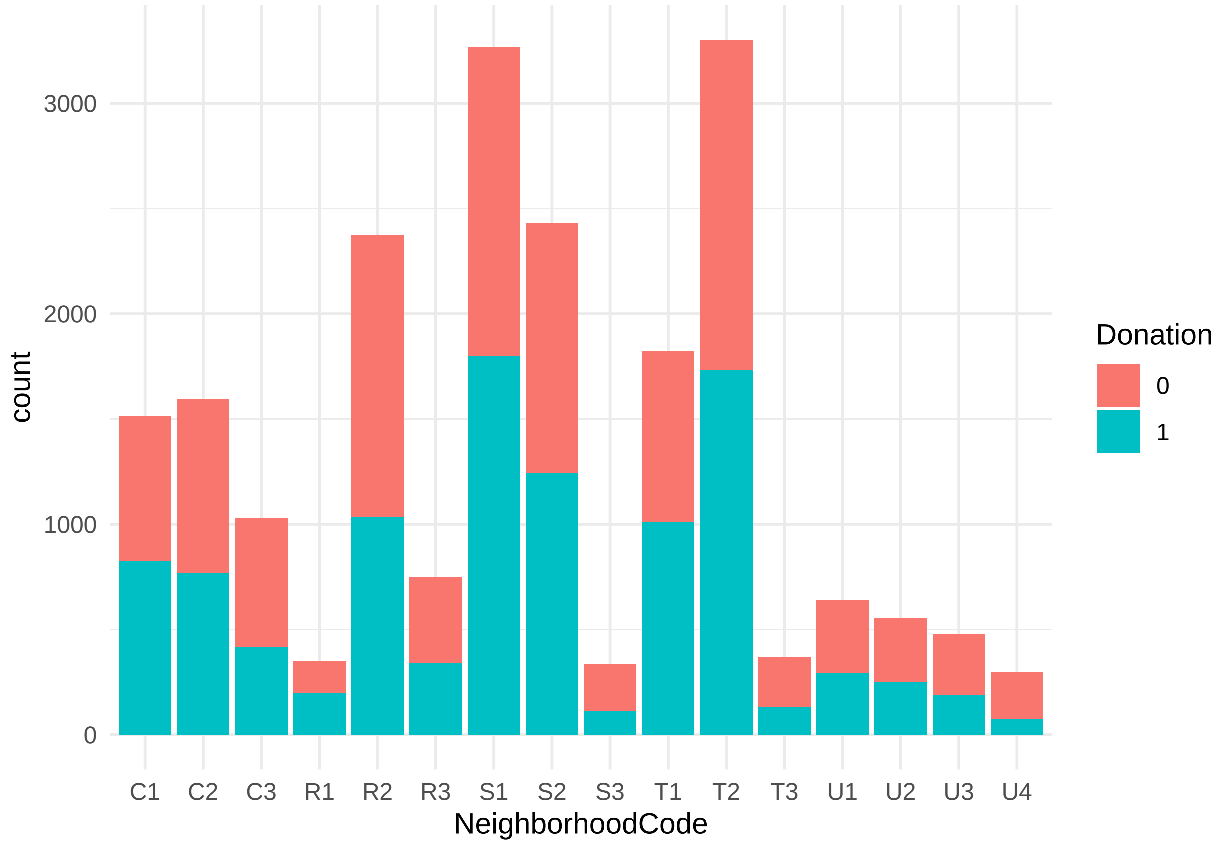
```
##
##     0     1
## 10838 10676
```

Create a training and testing split

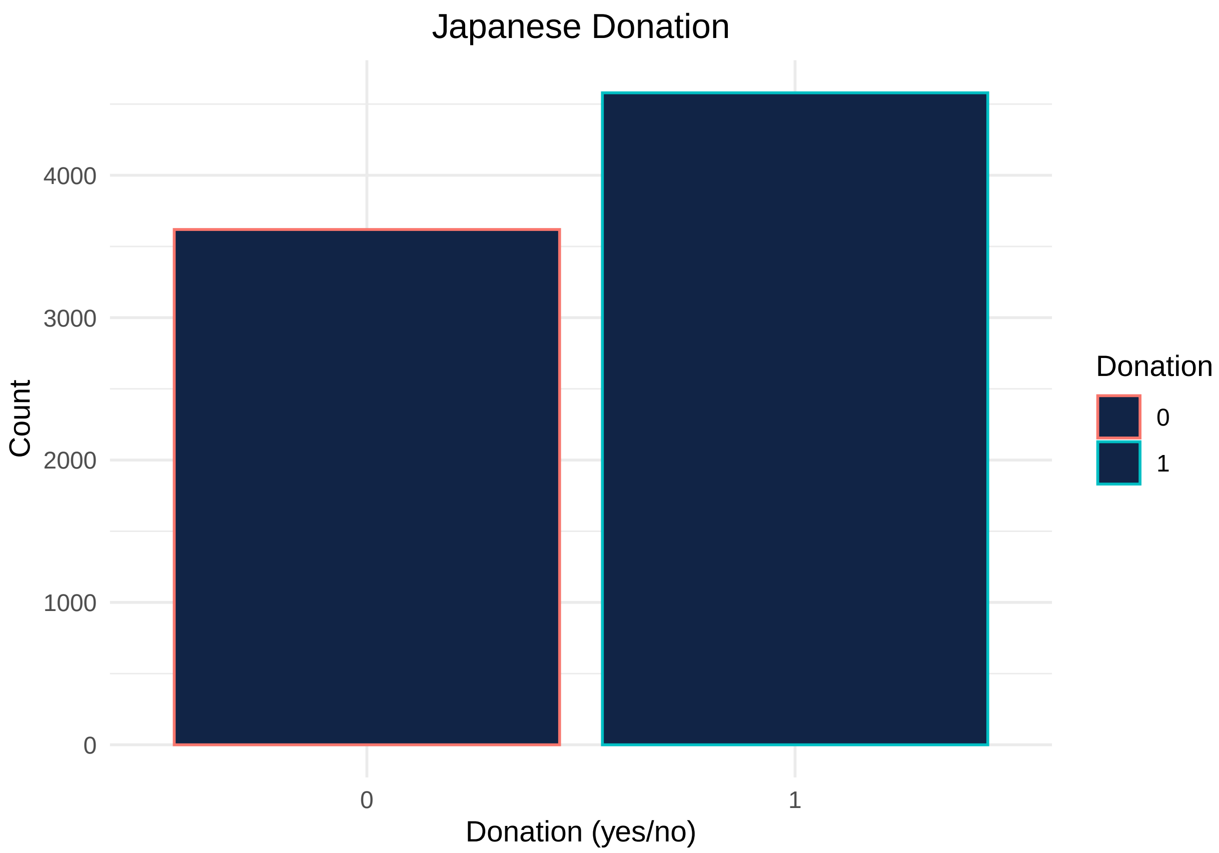
```
set.seed(1)
tr_ind = sample(nrow(balancedb1),.8*nrow(balancedb1), replace = F)
b1train = balancedb1[tr_ind,]
b1test = balancedb1[-tr_ind,]
```

Creating graphic models of important variables

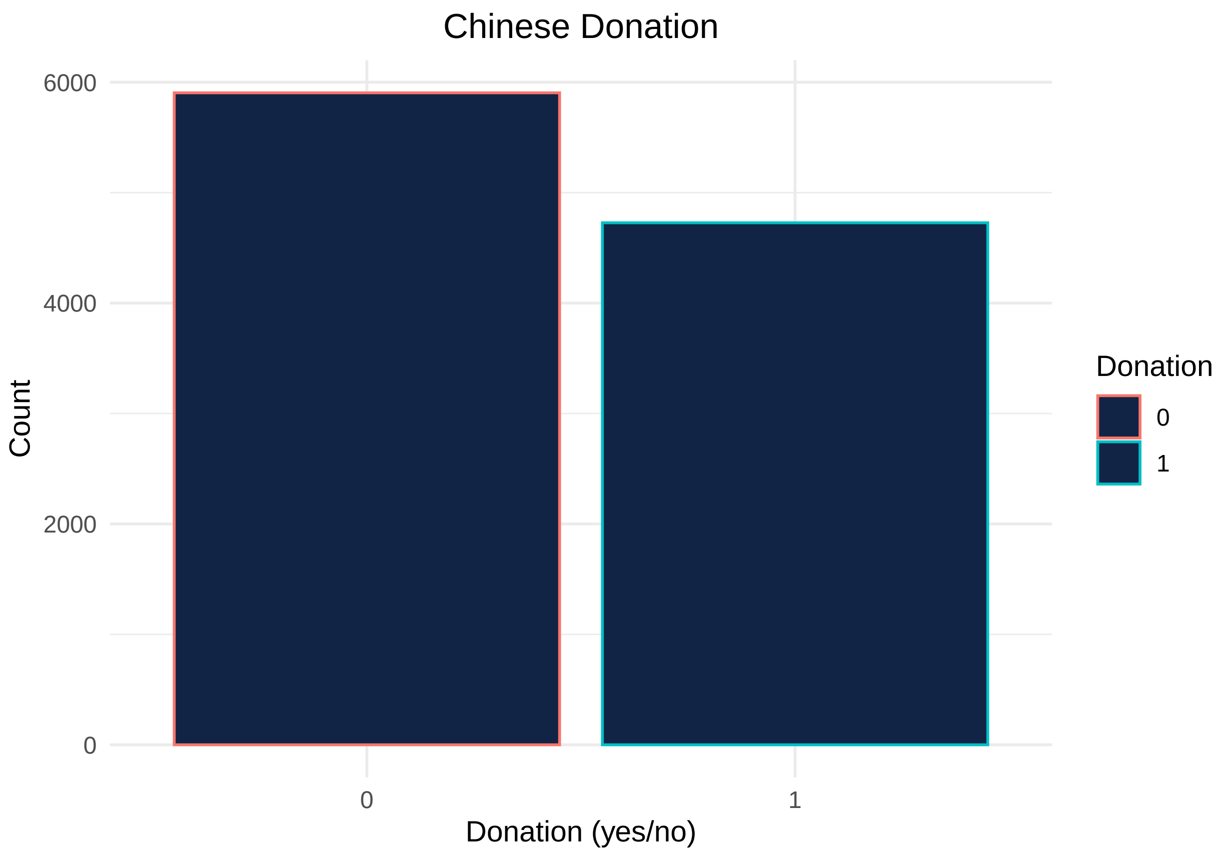
```
balancedb1 %>%
 filter(!(NeighborhoodCode %in% " ")) %>%
 ggplot() +
  aes(x = NeighborhoodCode, fill = Donation) +
  geom_bar() +
  scale_fill_hue(direction = 1) +
  theme_minimal()
```



```
balancedb1 %>%
 filter(!(NeighborhoodCode %in% " ")) %>%
 ggplot() +
  aes(x = Donation, colour = Donation, weight = PercJapa) +
  geom_bar(fill = "#112446") +
  scale_color_hue(direction = 1) +
  labs(
    x = "Donation (yes/no)",
    y = "Count",
    title = "Japanese Donation"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



```
balancedb1 %>%
 filter(!(NeighborhoodCode %in% " ")) %>%
 ggplot() +
  aes(
    x = Donation,
    colour = Donation,
    group = Donation,
    weight = PercChin
  ) +
  geom_bar(fill = "#112446") +
  scale_color_hue(direction = 1) +
  labs(
    x = "Donation (yes/no)",
    y = "Count",
    title = "Chinese Donation"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```


Chinese Donation
Count
6000
4000
2000
0
0
1
Donation (yes/no)
Donation
0
1

Results

Logistic Regression

Shows the accuracy that our logistic regression model has in predicting Yes and no Responses. The sensitivity of our model shows the accuracy we have for predicting "yes" responses and the specificity of our model shows the accuracy we have in predicting "no" responses. This model has an accuracy of 58.28%. We used .51 as our optimal cutoff point which gave us the best sensitivity and specificity which shows us that our model was accurate at predicting both yes and no responses.

```
a1 = glm(formula = Donation ~ NeighborhoodCode + PercJapa + PercChin + AvgRent + IC3H + LFAF + MaxDol +


car::vif(a1)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## NeighborhoodCode 2.405257 16        1.027806
## PercJapa         1.183531  1        1.087902
## PercChin         1.360207  1        1.166279
## AvgRent          2.108765  1        1.452159
## IC3H             1.346386  1        1.160339
## LFAF             1.209120  1        1.099600
## MaxDol           2.760343  1        1.661428
## LastDol          2.758851  1        1.660979
```

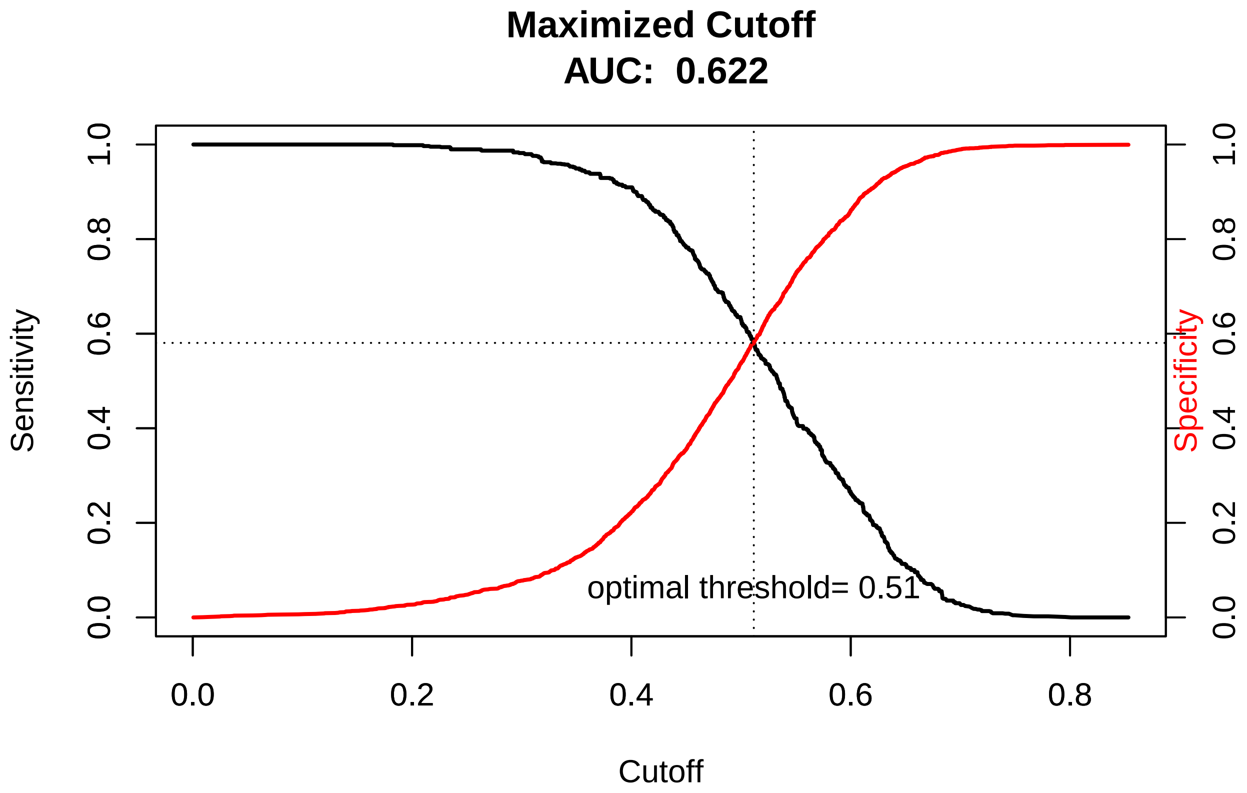
```
summary(a1)
```

```
##
## Call:
## glm(formula = Donation ~ NeighborhoodCode + PercJapa + PercChin +
##     AvgRent + IC3H + LFAF + MaxDol + LastDol, family = binomial,
##     data = b1test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9591  -1.1587   0.7686   1.1215   1.8448
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.311249   0.314318   4.172 3.02e-05 ***
## NeighborhoodCodeC1 -0.591886   0.275844  -2.146 0.031895 *
## NeighborhoodCodeC2 -0.818897   0.269836  -3.035 0.002407 **
## NeighborhoodCodeC3 -1.097742   0.290142  -3.783 0.000155 ***
## NeighborhoodCodeR1 -0.549097   0.366936  -1.496 0.134540
## NeighborhoodCodeR2 -0.997766   0.265537  -3.758 0.000172 ***
## NeighborhoodCodeR3 -0.781701   0.299902  -2.607 0.009147 **
## NeighborhoodCodeS1 -0.552635   0.263000  -2.101 0.035617 *
## NeighborhoodCodeS2 -0.666402   0.263147  -2.532 0.011328 *
## NeighborhoodCodeS3 -1.514217   0.375201  -4.036 5.44e-05 ***
## NeighborhoodCodeT1 -0.408988   0.271025  -1.509 0.131288
## NeighborhoodCodeT2 -0.699372   0.260006  -2.690 0.007149 **
## NeighborhoodCodeT3 -1.419982   0.343900  -4.129 3.64e-05 ***
## NeighborhoodCodeU1 -0.900740   0.319637  -2.818 0.004832 **
## NeighborhoodCodeU2 -1.065282   0.317121  -3.359 0.000782 ***
## NeighborhoodCodeU3 -1.221400   0.315352  -3.873 0.000107 ***
## NeighborhoodCodeU4 -2.138585   0.420360  -5.088 3.63e-07 ***
## PercJapa            0.034442   0.013977   2.464 0.013731 *
## PercChin           -0.054431   0.023496  -2.317 0.020524 *
## AvgRent            -0.008877   0.020739  -0.428 0.668633
## IC3H               -0.018312   0.005689  -3.219 0.001287 **
## LFAF                0.007138   0.002852   2.503 0.012329 *
## MaxDol              0.001412   0.006342   0.223 0.823831
## LastDol            -0.042504   0.007058  -6.022 1.72e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5965.1  on 4302  degrees of freedom
## Residual deviance: 5734.4  on 4279  degrees of freedom
## AIC: 5782.4
##
## Number of Fisher Scoring iterations: 4
```

```
predprob = predict.glm(a1, newdata = b1test, type = "response")
predclass_log = ifelse(predprob >= .51, "1", "0")
caret::confusionMatrix(as.factor(predclass_log), as.factor(b1test$Donation), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1235  892
##          1  903 1273
##
##                Accuracy : 0.5828
##                  95% CI : (0.5679, 0.5976)
##     No Information Rate : 0.5031
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.1656
##
##  Mcnemar’s Test P-Value : 0.8134
##
##             Sensitivity : 0.5880
##             Specificity : 0.5776
##          Pos Pred Value : 0.5850
##          Neg Pred Value : 0.5806
##              Prevalence : 0.5031
##          Detection Rate : 0.2958
##    Detection Prevalence : 0.5057
##       Balanced Accuracy : 0.5828
##
##        ’Positive’ Class : 1
##
```

```
optim_threshold(a1,b1test, b1test$Donation)
```



Running the optimal threshold function showed up that .51 was the best number to reach the highest sensitivity and specificity within our logistic regression model.

```
m2.log = step(a1, direction = "backward")
```

```
## Start:  AIC=5782.36
## Donation ~ NeighborhoodCode + PercJapa + PercChin + AvgRent +
##     IC3H + LFAF + MaxDol + LastDol
##
##                      Df Deviance    AIC
## - MaxDol              1   5734.4 5780.4
## - AvgRent             1   5734.5 5780.5
## <none>                    5734.4 5782.4
## - PercChin            1   5740.4 5786.4
## - LFAF                1   5740.6 5786.6
## - PercJapa            1   5741.2 5787.2
## - IC3H                1   5744.8 5790.8
## - LastDol             1   5769.9 5815.9
## - NeighborhoodCode 16   5811.3 5827.3
##
## Step:  AIC=5780.41
## Donation ~ NeighborhoodCode + PercJapa + PercChin + AvgRent +
##     IC3H + LFAF + LastDol
```

```
##
##                    Df Deviance    AIC
## - AvgRent           1   5734.6 5778.6
## <none>                  5734.4 5780.4
## - PercChin          1   5740.5 5784.5
## - LFAF              1   5740.7 5784.7
## - PercJapa          1   5741.2 5785.2
## - IC3H              1   5744.8 5788.8
## - NeighborhoodCode 16   5811.4 5825.4
## - LastDol           1   5840.9 5884.9
##
## Step:  AIC=5778.59
## Donation ~ NeighborhoodCode + PercJapa + PercChin + IC3H + LFAF +
##     LastDol
##
##                    Df Deviance    AIC
## <none>                  5734.6 5778.6
## - LFAF              1   5740.7 5782.7
## - PercJapa          1   5741.3 5783.3
## - PercChin          1   5741.6 5783.6
## - IC3H              1   5744.9 5786.9
## - NeighborhoodCode 16   5812.5 5824.5
## - LastDol           1   5842.1 5884.1
```

```
summary(m2.log)
```

```
##
## Call:
## glm(formula = Donation ~ NeighborhoodCode + PercJapa + PercChin +
##     IC3H + LFAF + LastDol, family = binomial, data = b1test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9530  -1.1594   0.7681   1.1247   1.8477
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.293006   0.305325   4.235 2.29e-05 ***
## NeighborhoodCodeC1 -0.608613   0.273745  -2.223 0.026197 *
## NeighborhoodCodeC2 -0.824739   0.269620  -3.059 0.002222 **
## NeighborhoodCodeC3 -1.093165   0.289563  -3.775 0.000160 ***
## NeighborhoodCodeR1 -0.543617   0.365647  -1.487 0.137087
## NeighborhoodCodeR2 -0.987478   0.263605  -3.746 0.000180 ***
## NeighborhoodCodeR3 -0.768911   0.297025  -2.589 0.009634 **
## NeighborhoodCodeS1 -0.571267   0.260233  -2.195 0.028148 *
## NeighborhoodCodeS2 -0.674388   0.262739  -2.567 0.010265 *
## NeighborhoodCodeS3 -1.522272   0.374893  -4.061 4.90e-05 ***
## NeighborhoodCodeT1 -0.414229   0.270960  -1.529 0.126327
## NeighborhoodCodeT2 -0.696967   0.259505  -2.686 0.007237 **
## NeighborhoodCodeT3 -1.408846   0.342335  -4.115 3.87e-05 ***
## NeighborhoodCodeU1 -0.919240   0.317409  -2.896 0.003779 **
## NeighborhoodCodeU2 -1.076587   0.316344  -3.403 0.000666 ***
## NeighborhoodCodeU3 -1.226940   0.315257  -3.892 9.95e-05 ***
## NeighborhoodCodeU4 -2.138502   0.420060  -5.091 3.56e-07 ***
## PercJapa            0.034097   0.013922   2.449 0.014320 *
## PercChin           -0.056992   0.022801  -2.500 0.012434 *
## IC3H               -0.017866   0.005574  -3.205 0.001351 **
## LFAF                0.006956   0.002820   2.466 0.013655 *
## LastDol            -0.041371   0.004320  -9.578  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5965.1  on 4302  degrees of freedom
## Residual deviance: 5734.6  on 4281  degrees of freedom
## AIC: 5778.6
##
## Number of Fisher Scoring iterations: 4
```

```
car::vif(m2.log)
```

```
##                      GVIF Df GVIF^(1/(2*Df))
## NeighborhoodCode 1.649119 16        1.015755
## PercJapa         1.180751  1        1.086623
## PercChin         1.269170  1        1.126575
## IC3H             1.292857  1        1.137039
## LFAF             1.182873  1        1.087600
## LastDol          1.030925  1        1.015345
```

This step function would take out any variables that could have a GVIF over 5 which would present multiculinarity in the dataset.We used this function to go backwards through our model and take out variables that were not significant. Our final Model consisted of the variables Percentage Japanese, percentage chinese, IC3H, LFAF, and LastDol.

LDA Model

Shows the accuracy that our LDA model has in predicting Yes and no Responses. This model is 58.38% accurate. The LDA model shows us a higher accuracy but due to the fact that we were not able to implent an optimal cutoff for the LDA model. This accuracy could be seen as false because the sensitivity and specificity are still lower that the logistic regression model.

```
m1.lda = lda(formula = as.factor(Donation) ~ NeighborhoodCode + PercJapa + PercChin + AvgRent + IC3H + 
predclass_lda = predict(m1.lda, newdata = b1test)
caret::confusionMatrix(as.factor(predclass_lda$class),as.factor(b1test$Donation), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1128  781
##          1 1010 1384
##
##                Accuracy : 0.5838
##                  95% CI : (0.5689, 0.5986)
##     No Information Rate : 0.5031
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.167
##
##  Mcnemar's Test P-Value : 7.145e-08
##
##             Sensitivity : 0.6393
##             Specificity : 0.5276
##          Pos Pred Value : 0.5781
##          Neg Pred Value : 0.5909
##              Prevalence : 0.5031
##          Detection Rate : 0.3216
##    Detection Prevalence : 0.5564
##       Balanced Accuracy : 0.5834
##
##        'Positive' Class : 1
##
```

Decision Tree

Shows the accuracy that our Decision Tree has in predicting Yes and no Responses. This model is 55.06% accurate. The Decision Tree model shows us a lower accuracy than the other two models but it is able to break down the variables and show us the importance of each variable ranked.It also holds a sensitivity of 96.97% which shows that our model can predict “yes” responses but is relatively bad at predicting “no” responses.

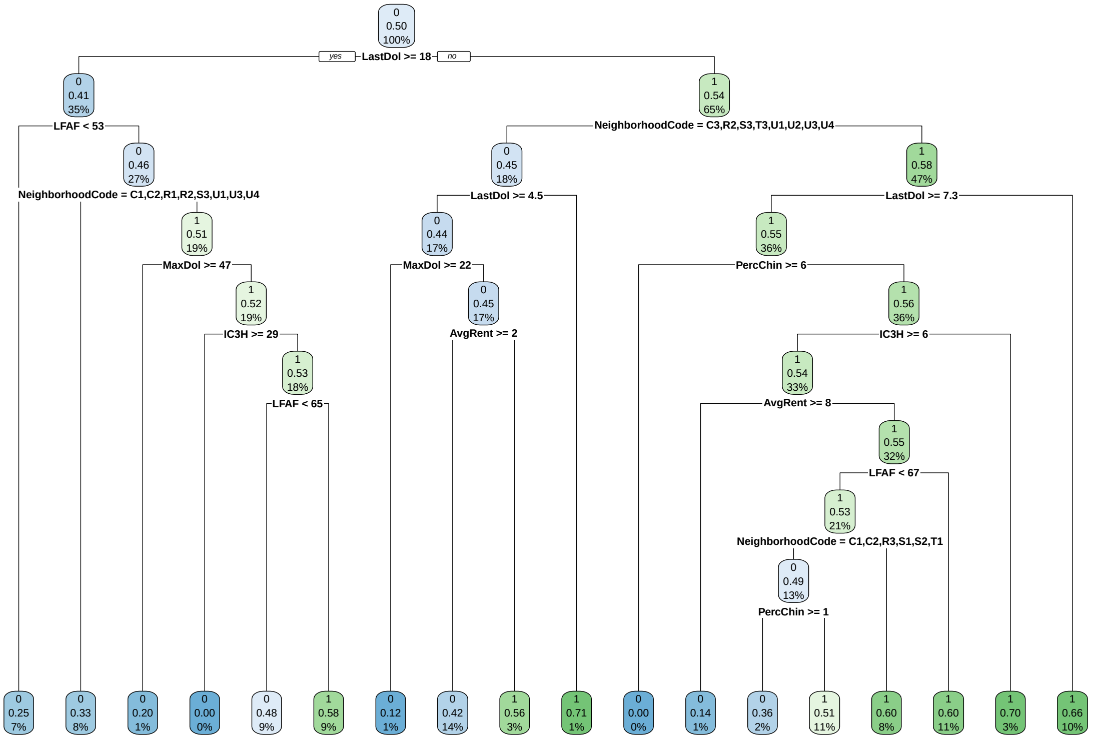
```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(rpart)
library(caTools)
set.seed(123)
split = sample.split(b1$Donation, SplitRatio = 0.8)
b1train = subset(b1, split == TRUE)
b1test = subset(b1, split == FALSE)

tree.b1train <- rpart(formula = Donation ~ NeighborhoodCode + PercJapa + PercChin + AvgRent + IC3H + LF
                      data = balancedb1,control =rpart.control(minsplit =2500,minbucket=1,cp=0))

rpart.plot(tree.b1train)
```

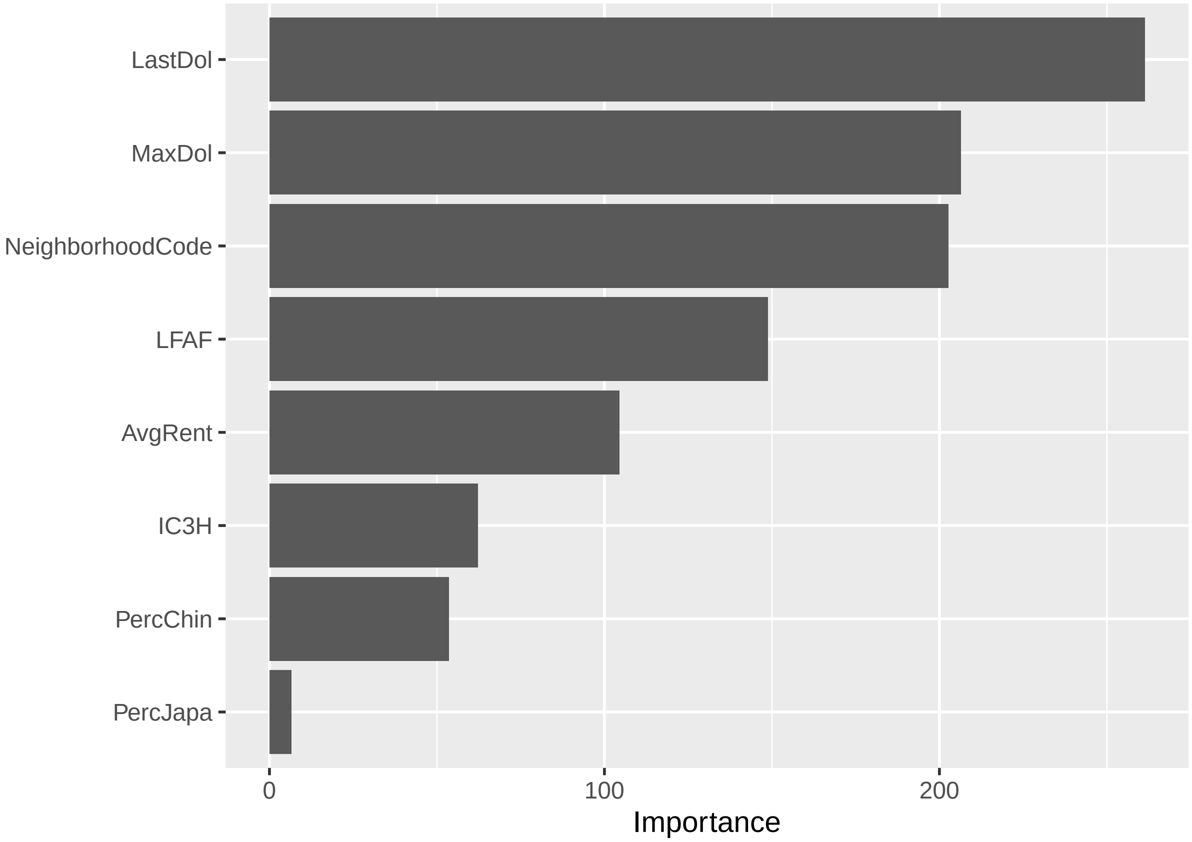


```
pred_b1tree = predict(tree.b1train, newdata = b1test, type = "class")
confusionMatrix(b1test$Donation, pred_b1tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1184  984
##          1   37   67
##
##                Accuracy : 0.5506
##                  95% CI : (0.5299, 0.5712)
##     No Information Rate : 0.5374
##     P-Value [Acc > NIR] : 0.1072
##
##                   Kappa : 0.0357
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96970
##             Specificity : 0.06375
##          Pos Pred Value : 0.54613
##          Neg Pred Value : 0.64423
##              Prevalence : 0.53741
##          Detection Rate : 0.52113
##    Detection Prevalence : 0.95423
##       Balanced Accuracy : 0.51672
##
##        'Positive' Class : 0
##
```

Using VIP function to plot important variables

```
vip(tree.b1train)
```



Conclusions and Recommendations

Within this case study, we explored the use of utilizing different modeling techniques to determine the likelihood of a consumer donating to a campaign. As previously mentioned, Ricciardi, Valente, Edmund, Cantoni, Green, Fiorillo, Picone, Santini, and Cesarelli's work concluded that the use of an LDA model accurately predicted which treatment a doctor should use on their patient with Coronary Artery disease. Thus, we hypothesized that this model would be the most accurate for our case study since we have similar variables.

We concluded that our hypothesis was correct, and our LDA model yielded an accuracy of 58.38% which was slightly more significant than our Logistical Regression Model. The LDA model had a higher accuracy percentage and held a higher sensitivity level of 63.93% which tells us that this model is more accurate than the logistic regression at predicting "yes" responses. We also observed in this model that the decision tree model held the lowest accuracy of 55.06% which was relatively low but it showed us a breakdown of the importance for each variable. The specificity was 6.37% which means that it was not a very good indicator of identifying "no" responses. This model we observed had a high rate of variance due to the fact that decision trees commonly overfit the model and have a high rate of variance. In the future, if this study were to be continued, we recommend accurately implementing an optimal cutoff to ensure precise results. An alternative methodology that we could have used in this case study is Ridge Regression. Ridge Regression is typically used when analyzing multiple regression variables that encounter multicollinearity.

References

n.d.a. *Decision Tree Regression.* https://www.saedsayad.com/decision_tree_reg.htm#:~:text=Decision%20tree%20builds%20regression%20or,decision%20nodes%20and%20leaf%20nodes.

n.d.b. https://www.researchgate.net/publication/271214468_Merging_Decision_Trees_A_Case_Study_In_Predicting_Student_Performance.

Brownlee, Jason. 2020. "Linear Discriminant Analysis for Machine Learning." *Machine Learning Mastery*, August. https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/.

Ricciardi, Carlo, Antonio Saverio Valente, Kyle Edmund, Valeria Cantoni, Roberta Green, Antonella Fiorillo, Ilaria Picone, Stefania Santini, and Mario Cesarelli. 2020. "Linear Discriminant Analysis and Principal Component Analysis to Predict Coronary Artery Disease." *Health Informatics Journal* 26 (3): 2181–92. https://doi.org/10.1177/1460458219899210.