Machine Learning Final Project

Singh, Gurjeet

June 10, 2018

Contents

1. Introduction

The purpose of this project is to develop a machine learning model for a charitable organization to improve the cost-effectiveness of their direct marketing campaigns to previous donors.

Based on the recent mailing records, the typical overall response rate is 10%. Out of those who respond (donate) to the mailing, the average donation is $14.50. Each mailing costs $2.00 to produce and send; the mailing includes a gift of personalized address labels and the assortment of cards and envelopes. It is not cost-effective to mail everyone because the expected profit from each mailing is 14.50 x 0.10 - 2 = -$0.55. Therefore, We would like to develop a classification model using data from the most recent campaign that can effectively capture likely donors so that the expected net profit is maximized.

Table 1 provides the features in the dataset.

Table 1: Data set features

Name	Description	Name	Description
ID	ID Number	incm	In $ thousands
reg1	Geographic Regions 1	inca	In $ thousands
reg2	Geographic Regions 2	plow	Low income category
reg3	Geographic Regions 3	npro	LfTm # of promotions rec. to date
reg4	Geographic Regions 4	tgif	$ amount of LfTm gifts to date
reg5	Geographic Regions 5 - Imputed	lgif	$ amount of largest gift to date
home	1 = HmOw, 0 = Not a HmOw	rgif	amount of most recent gift
chld	# of children	tdon	# of months since last donation
hinc	Household income	tlag	# of months bet. 1st and 2nd gifts
genf	0 = Male, 1 = Female	agif	AVG $ amount of gifts to date
wrat	9 = Highest, 0 = lowest	donr	1 = Donor, 0 = Non-donor
avhv	In $ thousands	damt	Donation Amount in $

2. Data Exploration and Preparation

The first step towards any modeling project is the Exploratory Data Analysis (EDA). This helps us to understand and analyze the dataset as well as summarize the main characteristics of variables. In this section, we will discuss, visualize, and review the basic statistics to understand the data and its distributions. We will also prepare the data for the modelings.

2.1. Data Overview

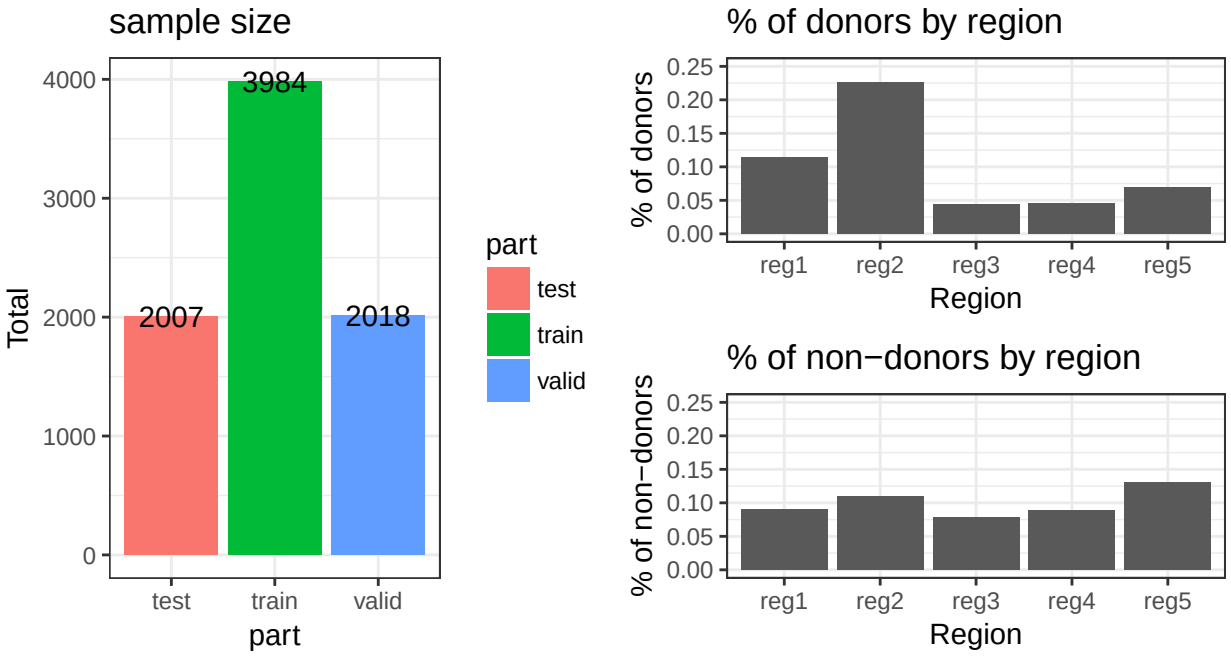
The table below shows the summary statistics of the charity dataset which includes training, validation, and test set. We notice that there are 8009 observations. We do not have any missing data for training and validation set. There are missing data for test set because we do not have any values for the response variables, donr and damt. Since the variables are not on the same scale, we will transform these variables in Section 2.3 (Data Preparation).

Table 2: Summary Statistic: Charity data

Name	Description	nobs	NAs	Minimum	Maximum	Mean	Median	Stdev
agif	AVG $ amount of gifts to date	8009	0	1.29	72.27	11.68	10.23	6.57
avhv	In $ thousands	8009	0	48.00	710.00	182.65	169.00	72.72
chld	# of children	8009	0	0.00	5.00	1.72	2.00	1.40
damt	Donation Amount in $	8009	2007	0.00	27.00	7.21	0.00	7.36

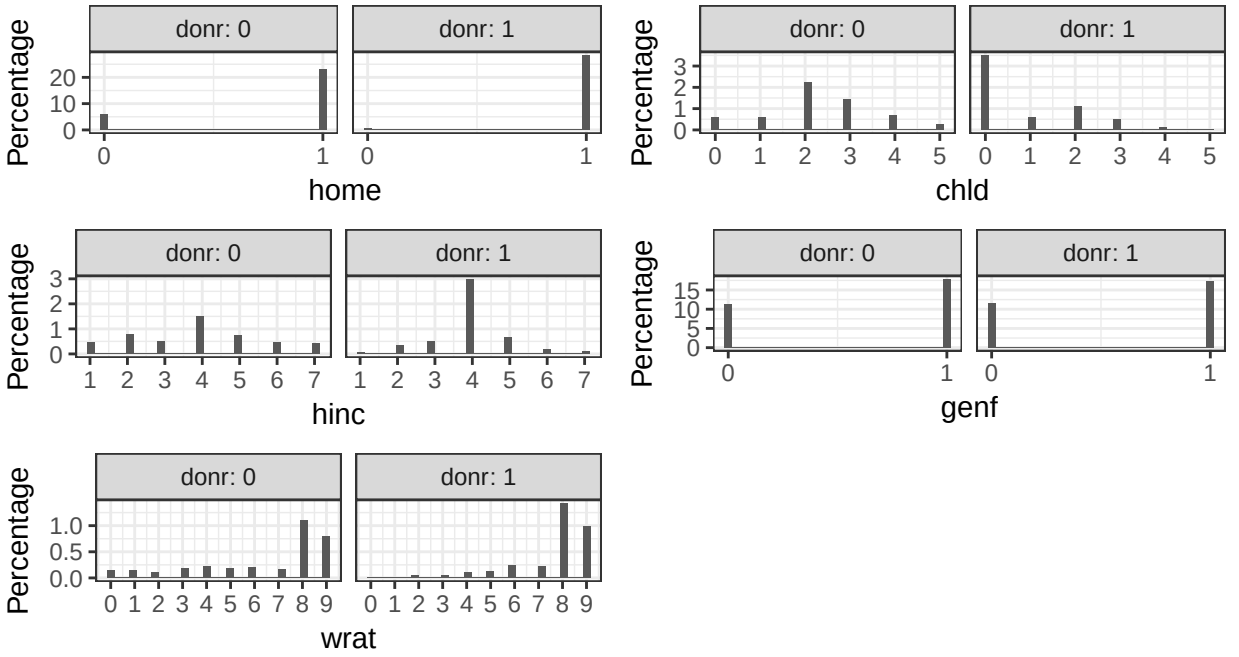
Name	Description	nobs	NAs	Minimum	Maximum	Mean	Median	Stdev
donr	1 = Donor, 0 = Non-donor	8009	2007	0.00	1.00	0.50	0.00	0.50
genf	0 = Male, 1 = Female	8009	0	0.00	1.00	0.61	1.00	0.49
hinc	Household income	8009	0	1.00	7.00	3.91	4.00	1.47
home	1 = HmOw, 0 = Not a HmOw	8009	0	0.00	1.00	0.87	1.00	0.34
ID	ID Number	8009	0	1.00	8009.00	4005.00	4005.00	2312.14
inca	In $ thousands	8009	0	12.00	305.00	56.43	51.00	24.82
incm	In $ thousands	8009	0	3.00	287.00	43.47	38.00	24.71
lgif	$ amount of largest gift to date	8009	0	3.00	681.00	22.94	16.00	29.95
npro	LfTm # of promotions rec. to date	8009	0	2.00	164.00	60.03	58.00	30.35
plow	Low income category	8009	0	0.00	87.00	14.23	10.00	13.41
reg1	Geographic Regions 1	8009	0	0.00	1.00	0.20	0.00	0.40
reg2	Geographic Regions 2	8009	0	0.00	1.00	0.32	0.00	0.47
reg3	Geographic Regions 3	8009	0	0.00	1.00	0.13	0.00	0.34
reg4	Geographic Regions 4	8009	0	0.00	1.00	0.14	0.00	0.35
reg5	Geographic Regions 5 - Imputed	8009	0	0.00	1.00	0.21	0.00	0.41
rgif	amount of most recent gift	8009	0	1.00	173.00	15.66	12.00	12.43
tdon	# of months since last donation	8009	0	5.00	40.00	18.86	18.00	5.78
tgif	$ amount of LfTm gifts to date	8009	0	23.00	2057.00	113.07	89.00	85.48
tlag	# of months bet. 1st and 2nd gifts	8009	0	1.00	34.00	6.36	5.00	3.70
wrat	9 = Highest, 0 = lowest	8009	0	0.00	9.00	6.91	8.00	2.43

The left plot in the figure below shows the sample size. We have 2007 observations in the test set, 3984 observations in training set, and 2018 observations in the validation set. The right plots in the figure below provide us with the information of donors and non-donors by each region. We can see that 'reg2' is highly predictive variable and it has the most percentage of donors.

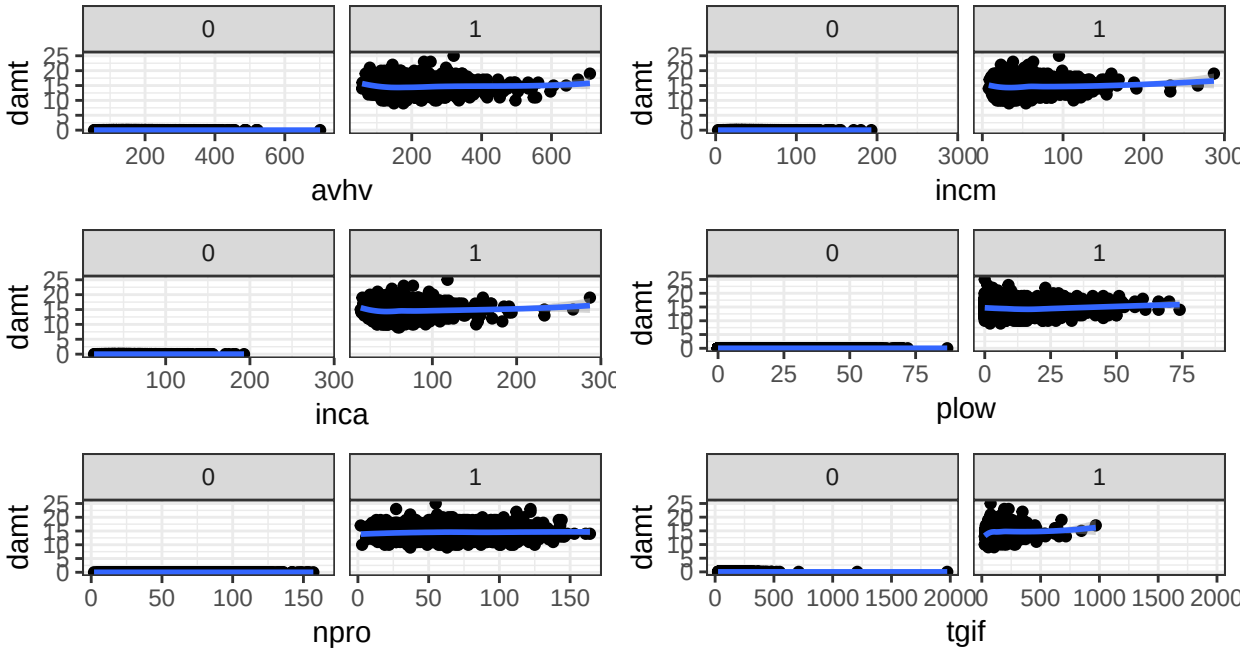


2.2. Visualizations

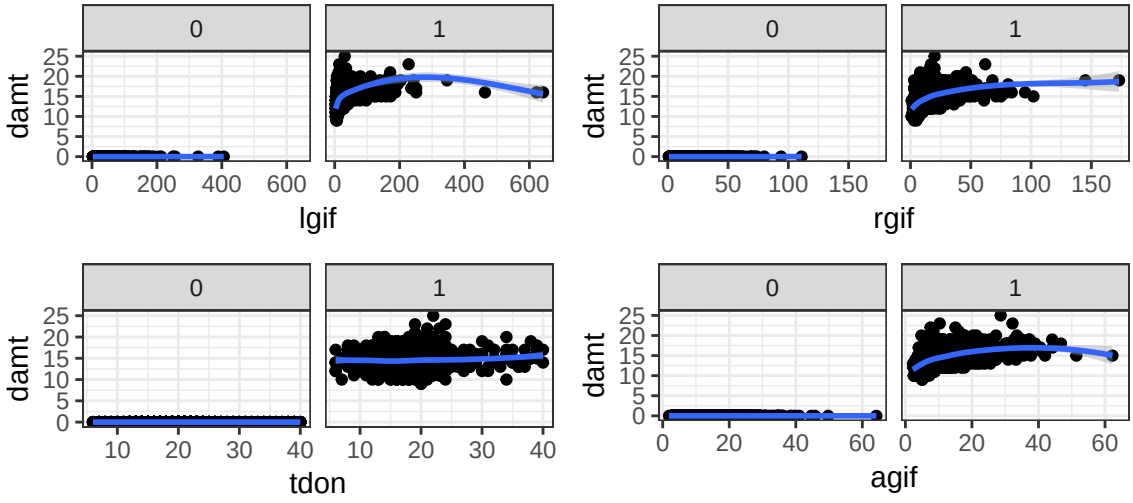
The figure below shows the distribution of donors and non-donors by various predictor variables. In the figure below, we see that high percentage donors are homeowners, have no children, with a household income of category 4, and wealth rating of 8 or 9. The figure also suggests that percentage of female donors are more than male donors.



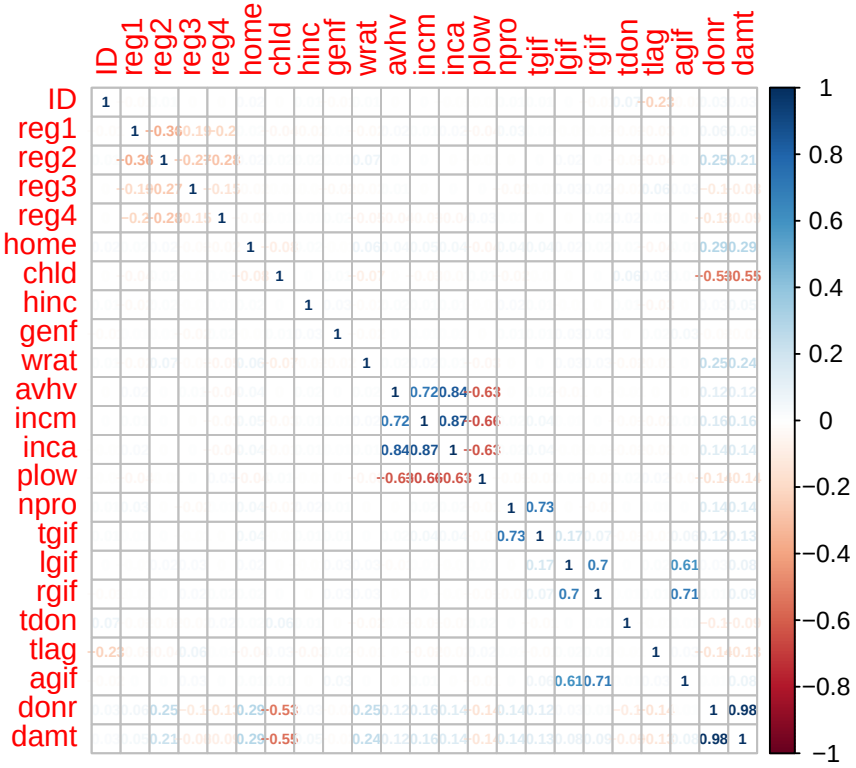
The figure below shows various scatterplots for each continuous predictor variables. These are split by donors vs. non-donors. None of the predictor variables show any linear relationship with the response variable, damt. This provides us with the hint that we have to use flexible and non-linear methods during modeling to obtain the maximum profit.



The figure below again shows the similar trend as the previous figure. We notice that these variables do not have a strong linear relationship with the response variable, damt.



The figure below shows the correlation plot of the data. We notice that average home value (avhv) is highly correlated with median family income (incm), average family income (inca), and percentage categorized as low income (plow). There is over 70% correlation between avhv and incm, 80% between avhv and inca, and a negative correlation between avhv. The average family income (inca) is also highly correlated with median family income (incm). It has over 86% correlation. Therefore, we will remove average family income (inca) from our data and retain the other two. There is over 70% correlation between lifetime number of promotions (npro) and lifetime dollar gifts (tgif). We will remove npro from the dataset. Lastly, there is about 70% correlation between most recent gifts (rgif) and largest gift(lgif). rgif variable also has 71% correlation with the average gift (agif). Therefore, we will remove rgif variable and retain the other two in our final dataset.



2.3. Data Preparation

In order to prepare our data for the modeling, we have retained only the variables listed in the table below. Apart from removing the correlated variables, we have also transformed our variables using log transformation. We have used the 'scale' function in R to perform this transformation. Lastly, we split the dataset into training, validation, and test set. For prediction modeling, we only retain observations for the donors i.e. donr = 1.

Table 3: List of variables retained

ID	home	avhv	tdon
reg1	chld	incm	tlag
reg2	hinc	plow	agif
reg3	genf	tgif	donr
reg4	wrat	lgif	damt

3. Classification Modeling

This section discusses the various models build to identify the likely donors so that we can maximize the net profit. In this section, we will develop 'Random Forest', 'Boosting', 'Support Vector Classifier', 'Support Vector Machine', and 'Lasso'.

3.1. Model Building

3.1.1 Model 1: Random Forest

The summary below shows the output of the Random Forest model for classification. The model created 800 trees with 4 variables in each split. OOB Error rate on the training set it 10.39%. It also provide us with training set confusion matrix.

```
Call:
 randomForest(formula = as.factor(donr) ~ ., data = train_df_std[,      !names(train_df_std) %in% c("ID
               Type of random forest: classification
                     Number of trees: 800
No. of variables tried at each split: 4

        OOB estimate of  error rate: 10.39%
Confusion matrix:
     0    1 class.error
0 1763  226  0.11362494
1  188 1807  0.09423559
```

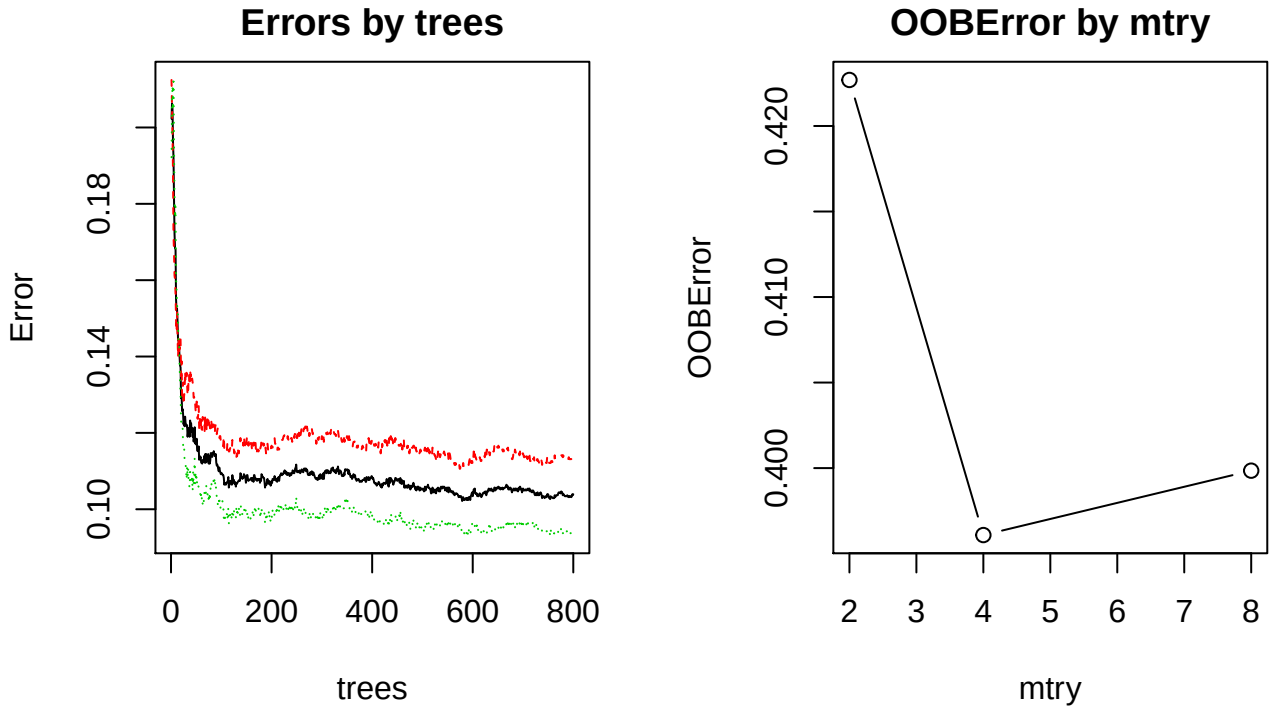
The table below show us the confusion matrix result obtained using the validation test set. We see that the accuracy is 89.64%. The accuracy interval is between 88% and 90% for this model. There are 935 mails that our model correctly predicted.

Table 4: Confusion Matric Result

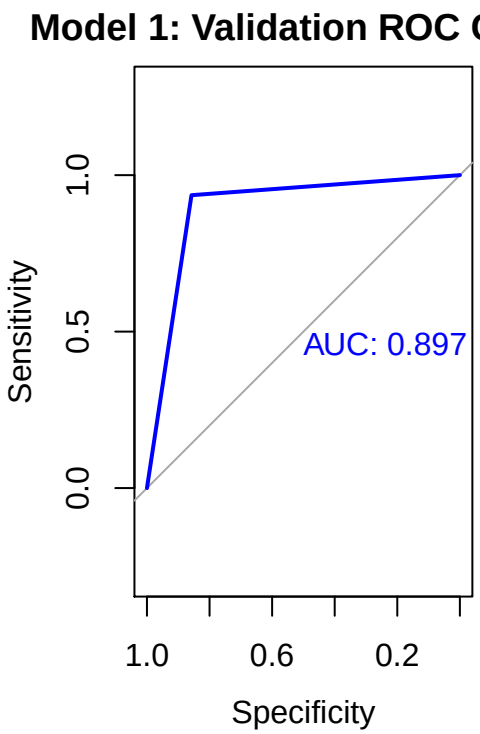
	x
Accuracy	0.89643
Kappa	0.79301
AccuracyLower	0.88231
AccuracyUpper	0.90939
AccuracyNull	0.50496
AccuracyPValue	0.00000
McnemarPValue	0.00000

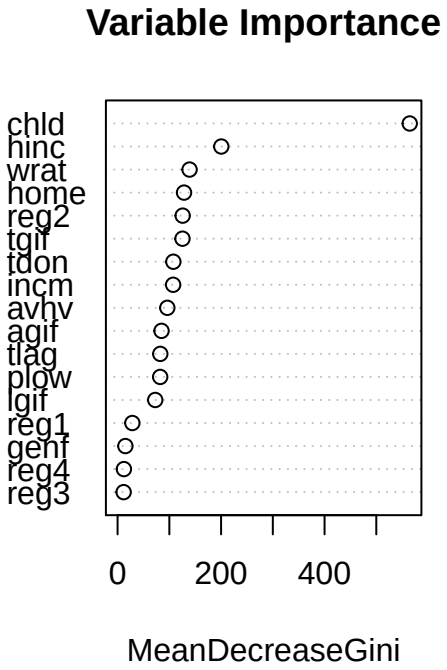
	0	1
0	874	64
1	145	935

The figure below shows the training errors by the number of trees on the left. We see that the errors significantly drops after using 100 trees. On the right, we see the results from running our Random Forest through another method to find the best variable split. It appears that this method also suggests that using 4 variables at each split is the best option. Since our model already uses the 4 variables split, we will accept and continue with this model.



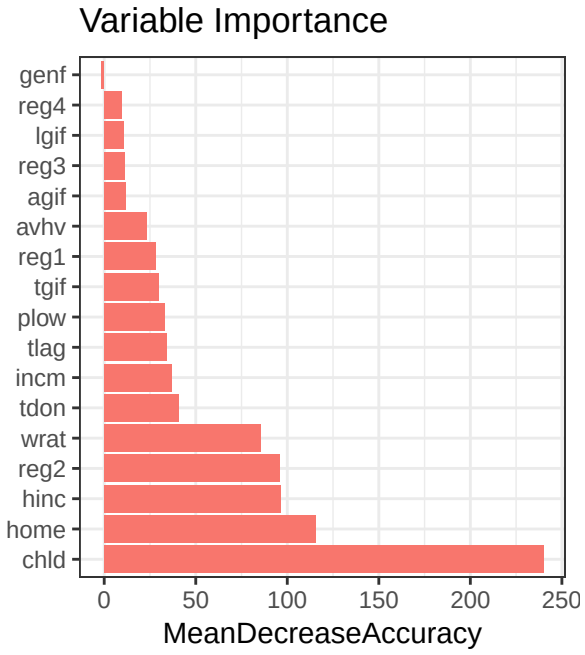
The figure below shows the validation set ROC curve on the left. This model predicts the area under the curve (AUC) to be at 89%. With the accuracy of 89.64% and AUC of 89%, this model seems to a good candidate to be the final model. On the right plot, we see that the model shows us the mean decrease in the Gini value. The model selects 'chld' as the most important variable.

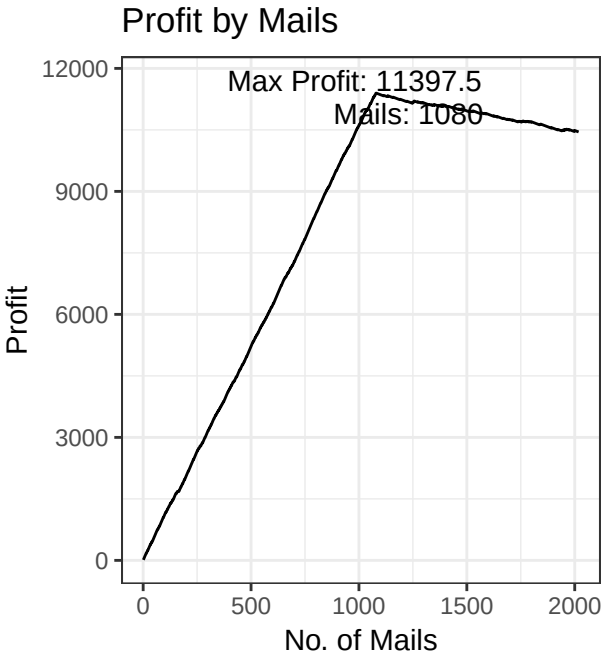




In the figure below, we see again see the variable important plot on the left. However, this plot plots the variable based on the mean decrease of accuracy. We notice that 'chld' again selected as the most important variable. However, the second most important variable now is the 'home' instead of 'hinc' as shown in the previous plot.

On the right, we have a plot that shows the profit that can be obtained by the number of mails sent. We see that the maximum profit of $11,397.50 can be obtained by sending the 1080 mails.





The table below shows this model predicts that the maximum profit of $11,397.50 can be obtained by sending 1080 mails.

Table 5: Profit and No. of mails by Classification Table

Mails	Max Profit
1080	11397.5

3.1.2 Model 2: Boosting

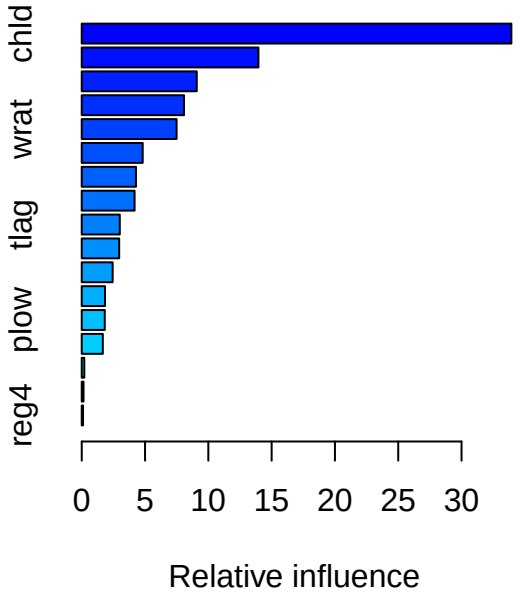
using caret package

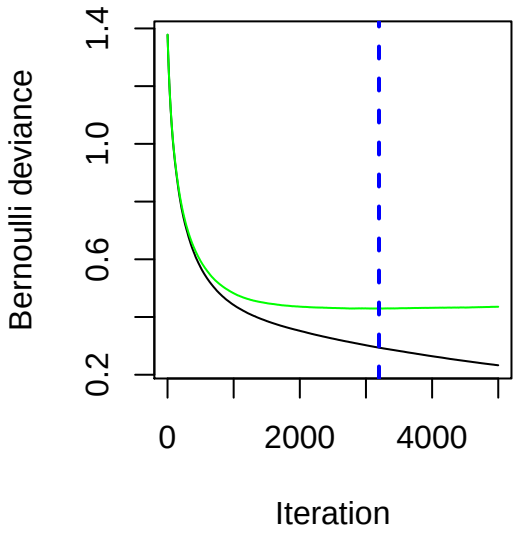
The table below shows the relative influence statistics from the boosting model. This model selects 'chld' as the most influential variable. These results are pretty consistent with the random forest variable importance results. Apart from the 'chld', model lists that home income (hinc), region 2 (reg2), homeowner (home), and wealth rating (wrat) are the next most important variables.

Table 6: Relative influence statistics

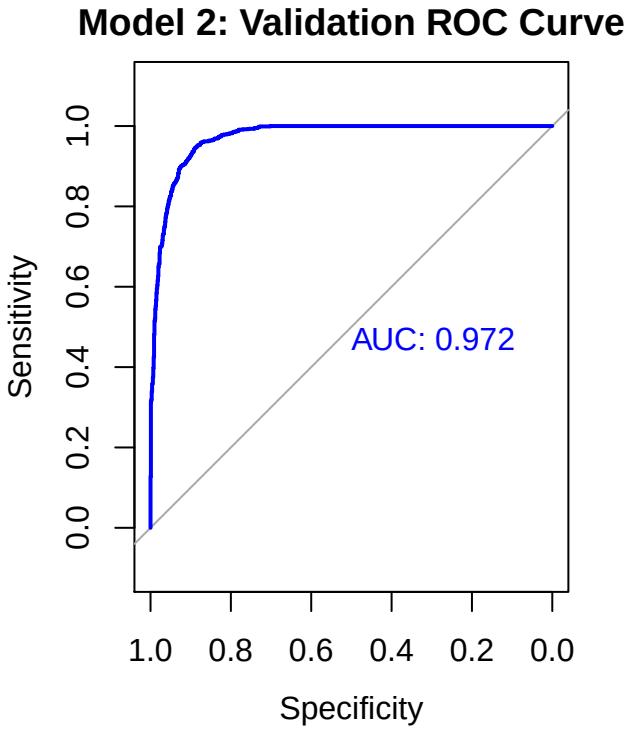
	var	rel.inf
chld	chld	33.9368131
hinc	hinc	13.9567608
reg2	reg2	9.0854435
home	home	8.0791815
wrat	wrat	7.4926458
tgif	tgif	4.8146421
incm	incm	4.2850103
tdon	tdon	4.1736844
tlag	tlag	3.0060357
avhv	avhv	2.9580047
agif	agif	2.4407118
reg1	reg1	1.8344121
plow	plow	1.8208630
lgif	lgif	1.6669460
reg3	reg3	0.2030811
genf	genf	0.1417937
reg4	reg4	0.1039706

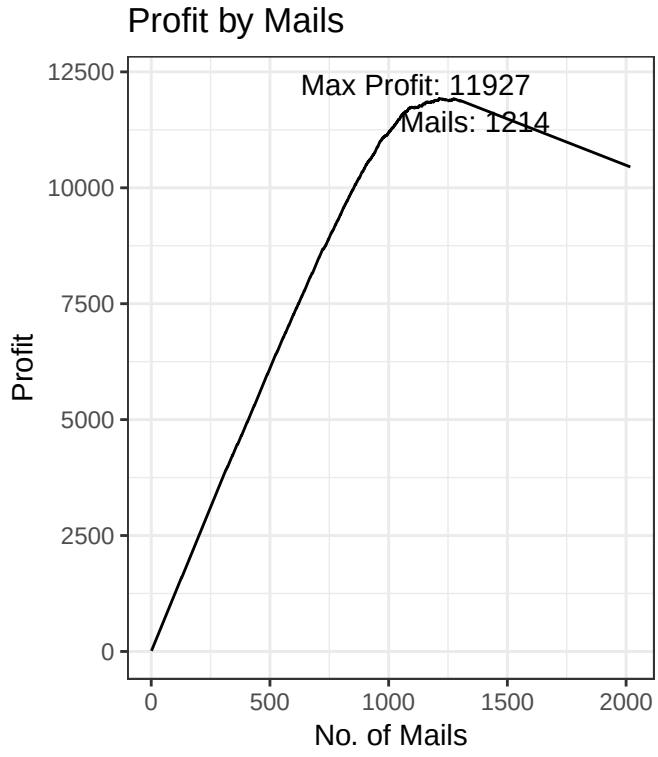
The figure below shows two plots. On the left, we have the relative influence plot that provides us with the important variables information. On the right, we have a plot that provides us best iteration of the model that produced the best result. The model best iteration was around 3100 iterations. After that, errors stay pretty same. We used this iteration information for our prediction.





In the figure below, we have validation ROC curve plot on the left for model 2. This model predicts the area under the curve (AUC) to be at 97%. This model produces metrics that way better than our random forest model. With the AUC of 97%, this model also seems to a good candidate to be the final model. However, our final selection of the model will depend on the maximum profit. On the right, we have a plot that shows the profit that can be obtained by the number of mails sent to the donors. We see that the maximum profit of $11,927 can be obtained by sending the 1214 mails.





The table below shows this model predicts that the maximum profit of $11,927 can be obtained by sending 1214 mails. This has over $500 increase in the profit by sending additional 134 mails. It would interesting to see how other models perform in the given scenario.

Table 7: Profit and No. of mails by Classification Table

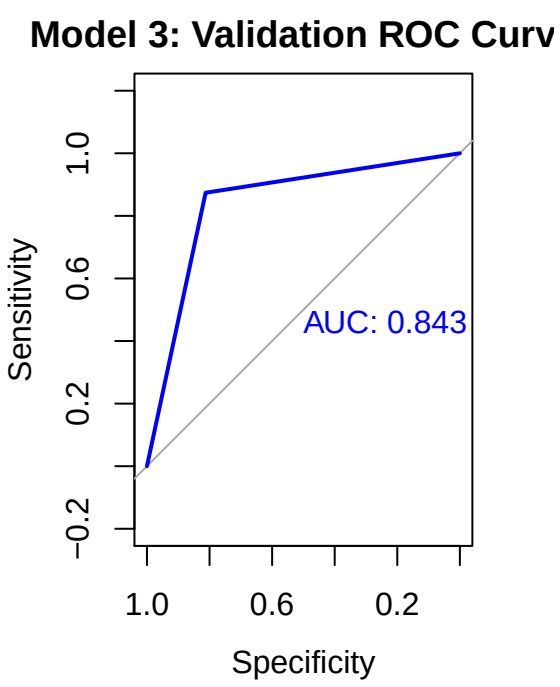
Mails	Max Profit
1214	11927

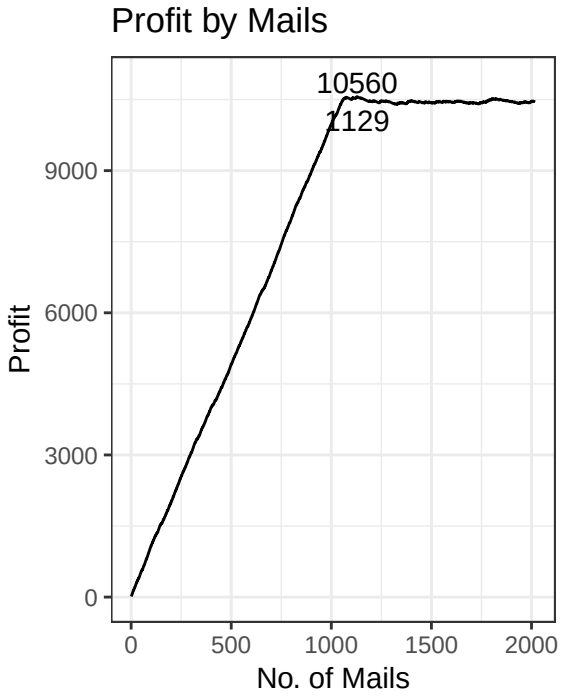
3.1.3 Model 3: Support Vector Classifier

Our next model uses the Support Vector Classifier method. We have used the cross-validation approach to select the best model. The summary below shows that best model was obtained using cost = 0.1 and gamma = 0.0588. The number of support vectors from this model is 1528. In each class, we have the equal number of vectors i.e. 764.

```
Call:
best.tune(method = svm, train.x = as.factor(donr) ~ ., data = train_df_std[,
    !names(train_df_std) %in% c("ID", "damt")], ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10)), scale = FALSE, kernel = "linear",
    parallel = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.05882353

Number of Support Vectors:  1528

 ( 764 764 )


Number of Classes:  2

Levels:
 0 1
```

In the figure below, we have validation ROC curve plot on the left for model 3 i.e. Support Vector Classifier. This model predicts the area under the curve (AUC) to be at 84.3%. This model does not perform as good as our previous two models. On the right, we have a plot that shows the profit that can be obtained by the number of mails sent to the donors. We see that the maximum profit of $10,460 can be obtained by sending the 1129 mails.





The table below shows this model predicts that the maximum profit of $10,530 can be obtained by sending 1064 mails. The profit from this model did not beat the second model. Hence, we will not select this model as our final model.

Note, the numbers in the table differs a little from the above plot. This is because, in the plot, we are capturing the maximum profit and associated index point. At 1129, we found the profit would increase by $30 at an expense of sending 65 mails. Therefore, we would select the maximum profit and mail from the classification table.

Table 8: Profit and No. of mails by Classification Table

Mails	Max Profit
1064	10530.5

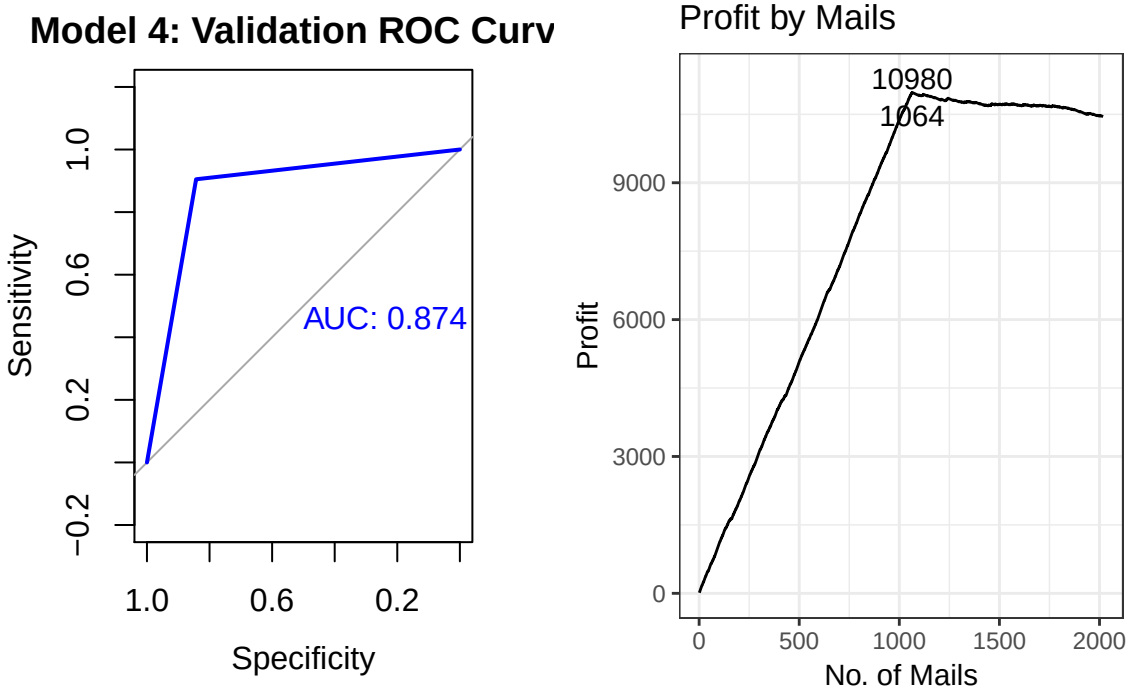
3.1.4 Model 4: Support Vector Machine

Our next model uses the Support Vector Machine method. We again have used the cross-validation approach to select the best model. The summary below shows that best model was obtained using cost = 1 and gamma = 0.1. The number of support vectors from this model is 1648. Two classes are split with 842 and 806 vectors.

```
Call:
best.tune(method = svm, train.x = as.factor(donr) ~ ., data = train_df_std[,
    !names(train_df_std) %in% c("ID", "damt")], ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5), gamma = c(0.1, 0.5, 1, 2)), scale = FALSE,
    kernel = "radial", parallel = TRUE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
```

```
       cost:  1
      gamma:  0.1

Number of Support Vectors:  1648

 ( 842 806 )


Number of Classes:  2

Levels:
 0 1
```

In the figure below, we have validation ROC curve plot on the left for model 4 i.e. Support Vector Machine. This model predicts the area under the curve (AUC) to be at 84.3%. This model does not perform as good as our first two models, random forest and boosting. However, it did perform better than Support Vector Classifier. This is because we do not have linear trend in the data. On the right, we have have plot that shows the profit that can be obtained by the number of mails sent to the donors. We see that the maximum profit of $10,980 can be obtained by sending the 1064 mails.



The table below shows this model predicts that the maximum profit of $10,980 can be obtained by sending 1064 mails. The profit from this model did not beat the second model. Hence, we will not select this model as our final model.

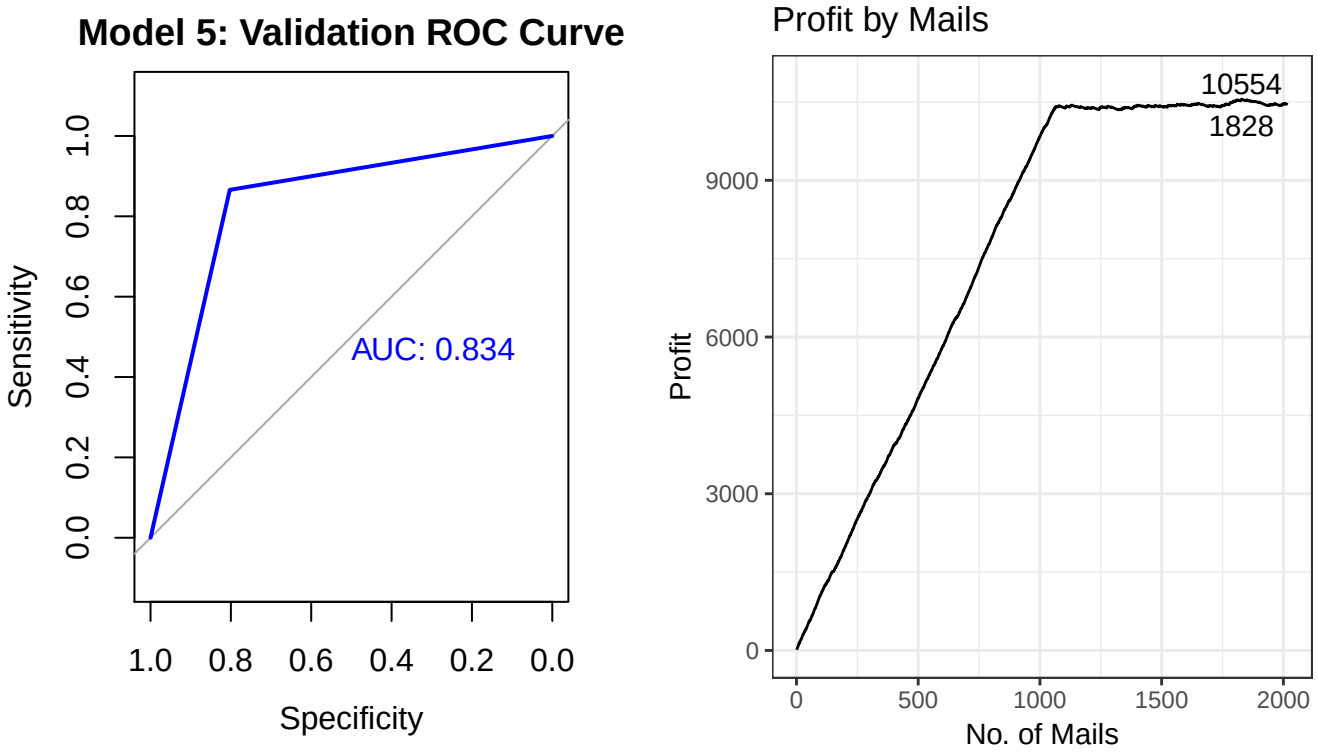
Table 9: Profit and No. of mails by Classification Table

Mails	Max Profit
1064	10980

3.1.5 Model 5: Lasso

Our last model for Classification Modeling was built using Lasso. We used the cross-validation approach to select the best lamda value for our prediction.

The figure below shows the validation set ROC curve plot on the left for model 5 i.e. Lasso. This model predicts the area under the curve (AUC) to be at 83.4%. This model does not perform as good as our first two models, random forest and boosting. On the right, we have a plot that shows the profit that can be obtained by the number of mails sent to the donors. We see that the maximum profit of $10,554 can be obtained by sending the 1828 mails. However, this is not the correct prediction because there's one point that predicted the highest maximum number. We will calculate the maximum profit from the classification table.



The table below shows this model predicts that the maximum profit of $10,410.50 can be obtained by sending 1066 mails. This is the worst performing model of all.

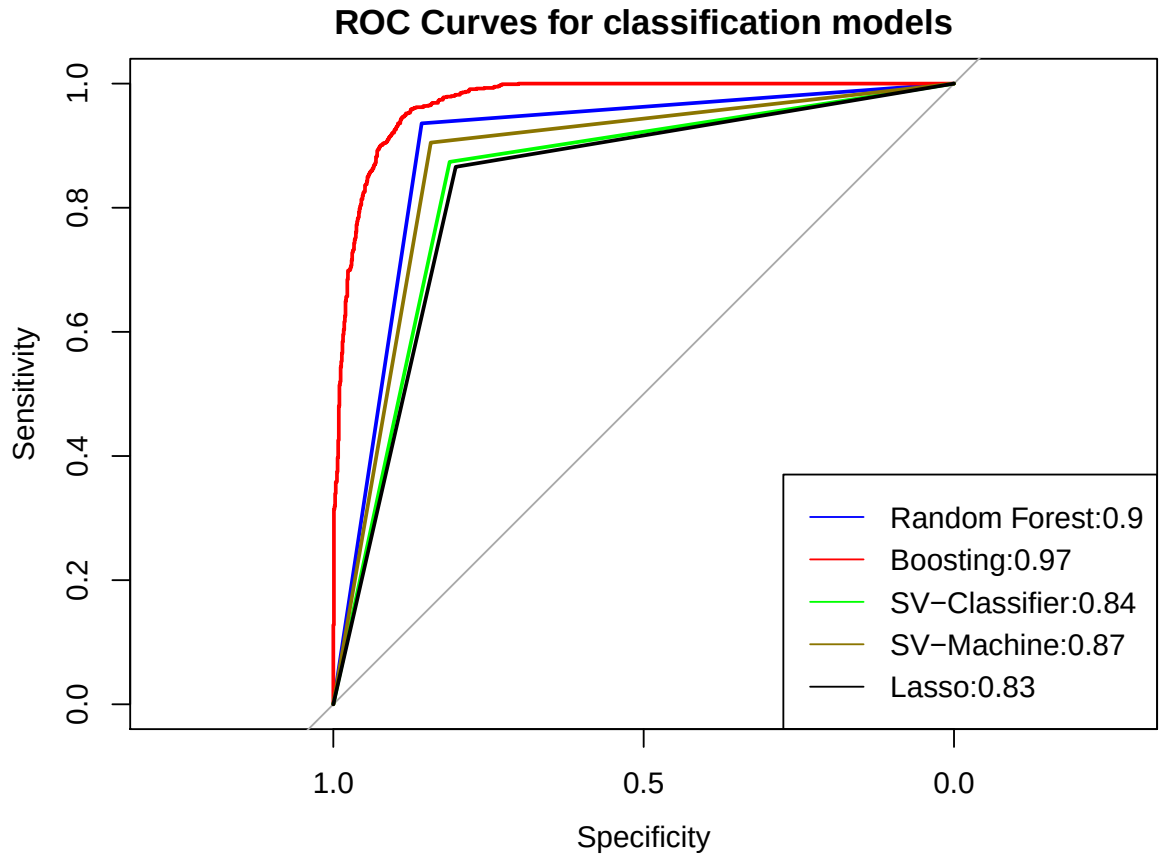
Table 10: Profit and No. of mails by Classification Table

Mails	Max Profit
1066	10410.5

3.2. Model Comparison

3.2.1. ROC Curve

The figure below shows the ROC curves from each model. On the right bottom, we can see that Booting performed better than any other model in terms of AUC value. The ROC curve also looks pretty good as compared to other curves.



3.2.2. Accuracy

The table below lists the accuracy values from each model. An interesting thing to note here is that our Boosting model did not get the highest accuracy number. However, it performed better from all the models. Random Forest got the highest accuracy value but it got the second highest profit. In my opinion, there might be some over-fitting in the random forest model.

Table 11: Accuracy values for each model

	Accuracy
Random Forest	0.8964321
Boosting	0.8845391
SV-Classifier	0.8429138
SV-Machine	0.8736373
Lasso	0.8339941

3.2.3. Profit

The table below lists the number of mails and maximum profit by each model. We can see that boosting model is the clear winner here. Therefore, we will select that as our final model.

Table 12: Number of mails and profit by each model

	Mails	Profit
Random Forest	1080	11397.5
Boosting	1214	11927
Support Vector Classifier	1064	10530.5
Support Vector Machine	1064	10980
Lasso for Classification	1066	10410.5

3.3. Model Selection

The table below shows the number of mails that was predicted by our model that will be sent. This is obtained using the test dataset. Our model predicts that 288 mails will be enough to maximize the profit in the test dataset.

Table 13: Prediction on the mails

	# of Mails
Mail = No	1719
Mail = Yes	288

4. Prediction Modeling

This section discusses the various models and their results to to predict the donation amount by each donor. In this section, we will develop 'Random Forest', 'Boosting', 'Ridge and Lasso', 'Principal Components Regression (PCR)', and 'Generalized Additive Model (GAM)'.

We will use the mean predicted error (MSE) to select our final model. The lower MSE value means the better performing model.

4.1. Model Building

4.1.1 Model 6: Random Forest

The summary below shows the output of the Random Forest model for regression. The model created 800 trees with 5 variables in each split. The mean squared error on the training set is 1.49 and R-squared value is 59.98%.

```
Call:
 randomForest(formula = damt ~ ., data = train_df_std_reg[, !names(train_df_std_reg) %in%      c("ID")]
               Type of random forest: regression
                     Number of trees: 800
No. of variables tried at each split: 5

          Mean of squared residuals: 1.499597
                    % Var explained: 59.98
```

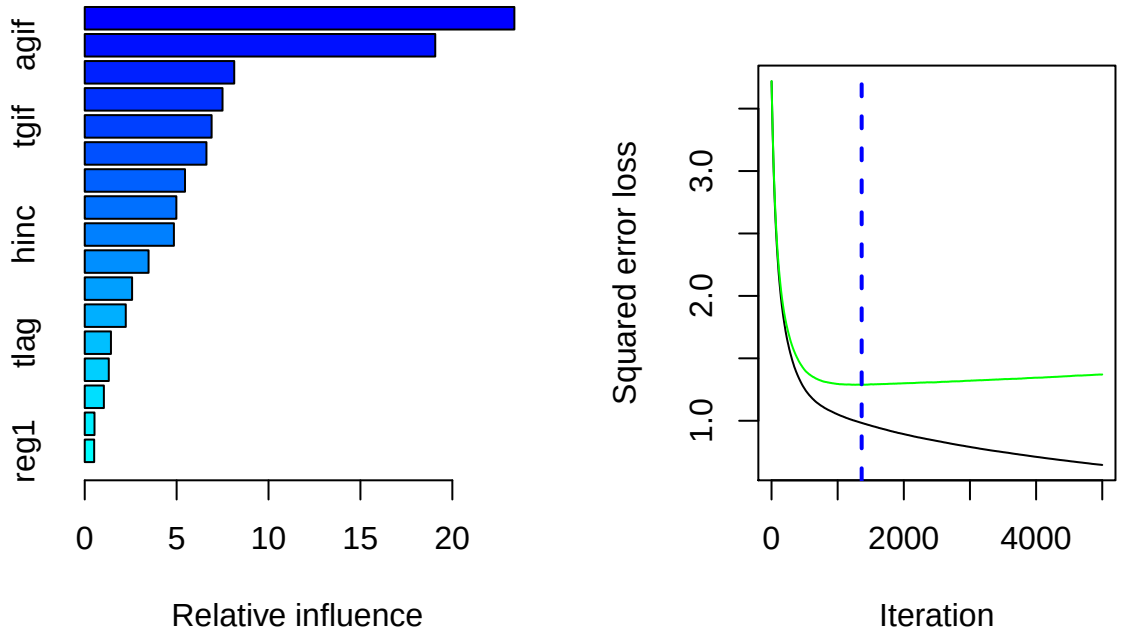
The table below shows the mean predicted error (MSE) and R-squared value on validation set. We have MSE of 1.699 and validation R-squared value of 61.12%.

Table 14: Random Forest: Metrics

	Values
MSE	1.699828
Valid R-Squared	0.611243

4.1.2 Model 7: Boosting

The figure below shows two plots. On the left, we have the relative influence plot that provides us with the important variables information. On the right, we have a plot that provides us best iteration of the model that produced the best result. The model best iteration was around 1350 iterations. After that, errors stay pretty same or increases. We used this iteration information for our prediction.



The table below shows the mean predicted error (MSE) and R-squared value on validation set. We have MSE of 1.463 and validation R-squared value of 66.04%. This is way better than Random Forest model.

Table 15: Boosting: Metrics

	Values
MSE	1.4637172
Valid R-Squared	0.6604076

4.1.3 Model 8: Ridge & Lasso Regression

Next, we created two model using the cross-validation approach. The table below shows the MSE and

R-squared for each model from the validation set. We notice that both did not perform that well. We have MSE of 1.95 and validation R-squared value of 55.35% for ridge regression. We have MSE of 1.95 and validation R-squared value of 55.347 for Lasso.

Table 16: Ridge & Lasso: Metrics

	Ridge	Lasso
MSE	1.9577431	1.9567562
Valid R-Squared	0.5535613	0.5547735

4.1.4 Model 9: Principal Components Regression (PCR)

Our next model was created using Principal Components Regression approach. Below is the summary statistics of this model. We have 17 components. It seems like that lowest cross-validation error is received using all the components. This would not be ideal step to perform. We also notice that after component 14, there not much decrease in the error and R-squared is pretty good for this component. Hence, we will use 14 components for prediction.

```
Data:   X dimension: 1995 17
    Y dimension: 1995 1
Fit method: svdpc
Number of components considered: 17

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           1.937    1.937    1.708    1.660    1.559    1.556    1.501
adjCV        1.937    1.937    1.704    1.659    1.555    1.559    1.500
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       1.481    1.474    1.472     1.465     1.419     1.420     1.362
adjCV    1.480    1.474    1.472     1.465     1.419     1.425     1.362
       14 comps  15 comps  16 comps  17 comps
CV        1.352     1.347     1.349     1.330
adjCV     1.352     1.346     1.348     1.329

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X     16.9895    30.16    42.00    49.39    56.44    63.25    68.90
damt   0.1449    23.03    27.19    36.09    36.19    40.68    42.38
      8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
X       74.13    78.83     82.83     86.44     89.54     92.49     94.96
damt    42.92    43.20     43.77     47.27     47.32     51.37     52.09
      15 comps  16 comps  17 comps
X        97.22     98.75    100.00
damt     52.54     52.59     53.97
```

The table below shows the mean predicted error (MSE) and R-squared value on validation set. We have MSE of 1.98 and validation R-squared value of 54.83%. This is one of the worst performing model so far.

Table 17: PCR: Metrics

	Values
MSE	1.9808197
Valid R-Squared	0.5483566

Values

4.1.5 Model 10: Generalized Additive Model (GAM)

Our last model for prediction is built using Generalized Additive Modeling method. We used the stepwise selection method to select the variables. Below is the summary statistics of this model. The anova table shows that all the variables are significant. We have used different degree of freedom on various variables to make it more non-linear.

```
Call: gam(formula = damt ~ reg4 + s(lgif, 12) + chld + s(agif, 4) +
    hinc + s(tgif, 6) + s(incm, 6), data = train_df_std_reg[,
    !names(train_df_std_reg) %in% c("ID")], trace = FALSE)
Deviance Residuals:
    Min      1Q  Median      3Q     Max
-3.3766 -0.8064 -0.1603  0.5307  8.6873

(Dispersion Parameter for gaussian family taken to be 1.5355)

    Null Deviance: 7474.744 on 1994 degrees of freedom
Residual Deviance: 3014.143 on 1962.999 degrees of freedom
AIC: 6550.846

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
              Df  Sum Sq Mean Sq F value                Pr(>F)
reg4           1  764.55  764.55 497.921 < 0.00000000000000022 ***
s(lgif, 12)    1 1128.27 1128.27 734.798 < 0.00000000000000022 ***
chld           1  434.48  434.48 282.963 < 0.00000000000000022 ***
s(agif, 4)     1  171.48  171.48 111.682 < 0.00000000000000022 ***
hinc           1  271.75  271.75 176.978 < 0.00000000000000022 ***
s(tgif, 6)     1   67.05   67.05  43.669   0.00000000004995760 ***
s(incm, 6)     1   88.93   88.93  57.919   0.00000000000004213 ***
Residuals   1963 3014.14    1.54
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova for Nonparametric Effects
            Npar Df  Npar F                 Pr(F)
(Intercept)
reg4
s(lgif, 12)      11 27.1037 < 0.00000000000000022 ***
chld
s(agif, 4)        3  7.9649            0.00002806 ***
hinc
s(tgif, 6)        5  6.1972            0.00001041 ***
s(incm, 6)        5  3.7956              0.002005 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The table below shows the mean predicted error (MSE) and R-squared value on validation set. We have MSE of 1.825 and validation R-squared value of 58.24%. This model did not beat our boosting model.

Table 18: Stepwise GAM: Metrics

	Values
MSE	1.8257340
Valid R-Squared	0.5824045

4.2. Model Comparison and Selection

The table below shows the mean predicted error (MSE) and R-squared value for all the models. We see that Boosting is the clear winner here with MSE of 1.464 and validation R-squared of 66%. Hence, we will again select boosting as our final model.

Table 19: MSE and R-Squared (validation set) for each model

	MSE	Valid R-Squared
Random Forest	1.7	0.611
Boosting	1.464	0.66
Ridge	1.958	0.554
Lasso	1.957	0.555
PCR	1.981	0.548
GAM	1.826	0.582

5. Conclusion

In conclusion, I would like to state that Boosting performed the best in both classification and prediction methods. It provided us with the maximum profit in classification modeling and gave the lowest MSE in the prediction modeling.

The table below shows the mean donation amount in our final report. Also, the mean value for donors only i.e. when Chat = 1.

Overall, this was an interesting project to work on. A unique way to select the models. Instead of relying on the metrics solely, we made the decision on the profit.

	Average donation Amount
All yhat values	14.23
Donors only (chat = 1)	14.76

6. Appendix I: R Code

```
knitr::opts_chunk$set(echo = F, warning = F, message = F, fig.show = "asis")
library(dplyr)
library(readr)
library(tibble)
library(tidyr)
library(ggplot2)
library(fBasics)
```

```
library(randomForest)
library(gbm)
library(e1071)
library(pROC)
library(glmnet)
library(pls)
library(gam)
load("ML_Project_Data.RData")
options(scipen = 999)

charity <- read_csv("charity.csv")
description <- read_csv("variable_description.csv")

knitr::kable(cbind(description[1:12,c(1,3)],
                   description[13:24,c(1,3)]),
             caption = "Data set features")
sum_charity <- charity %>%
                select(-part) %>%
                mutate(reg5 = ifelse(reg1 == 0 &
                                     reg2 == 0 &
                                     reg3 == 0 &
                                     reg4 == 0,1,0)) %>%
                basicStats() %>%
                t() %>%
                as.data.frame() %>%
                rownames_to_column(var = "Name") %>%
                arrange(Name)

sum_charity_desc <- sum_charity %>%
                      inner_join(description,
                                 by = "Name")


knitr::kable(cbind(sum_charity_desc[,c(1, 19)],
                   round(sum_charity_desc[,c(2,3,4,5, 8,9,15)],2)),
             caption = "Summary Statistic: Charity data")

by_part <- charity %>%
            group_by(part) %>%
            summarize(Total = n())


charity_2 <-  charity %>%
              mutate(reg5 = ifelse(reg1 == 0 &
                                   reg2 == 0 &
                                   reg3 == 0 &
                                   reg4 == 0,1,0)) %>%
              gather(key = "Region",
                     value = "Region_Value",
                     c("reg1", "reg2",
                       "reg3", "reg4",
                       "reg5"))
```

```r
by_region <-   charity_2 %>%
               dplyr::filter(part == c("train")) %>%
               group_by(Region) %>%
#percent of people from this region
               summarize(percnt_Reg_donr_yes =
                           mean(Region_Value == 1 &
                                  donr == 1),
# percent of people from that region is the donor
                          percnt_Reg_non_donr_yes =
                           mean(Region_Value == 1 &
                                  donr == 0)
                         ) %>%
               mutate(total_yes_no =
                        percnt_Reg_donr_yes +
                        percnt_Reg_non_donr_yes)



PlotSum <- ggplot(by_part, aes(x = part, y = Total, fill = part)) +
            geom_col() +
            geom_text(aes(label = Total), size = 4) +
            ggtitle("sample size") +
            theme_bw()

Plot1 <-  ggplot(by_region, aes(x = Region, y = percnt_Reg_donr_yes)) +
          geom_col() +
          scale_y_continuous(limits = c(0, 0.25)) +
          ggtitle("% of donors by region") +
          ylab("% of donors")+
          theme_bw()
Plot2 <- ggplot(by_region, aes(x = Region, y = percnt_Reg_non_donr_yes)) +
          geom_col() +
          scale_y_continuous(limits = c(0, 0.25)) +
          ggtitle("% of non-donors by region") +
          ylab("% of non-donors")+
          theme_bw()

gridExtra::grid.arrange(PlotSum, Plot1,
                        Plot2, layout_matrix =
                                  rbind(c(1,2),
                                    c(1,3)))

rm(by_part)
rm(charity_2)
rm(by_region)
rm(Plot1)
rm(Plot2)
rm(PlotSum)




train_df <-  charity %>%
```

```
                dplyr::filter(part == "train")

valid_df <-  charity %>%
                dplyr::filter(part == "valid")

test_df <-  charity %>%
                dplyr::filter(part == "test")



Plot1 <- ggplot(train_df, aes(x = home, y = (..density..))) +
          geom_histogram() +
        ylab("Percentage") +
        facet_wrap(~donr, labeller = label_both) +
        scale_x_continuous(breaks = seq(0,1,1)) +
        theme_bw()

Plot2 <- ggplot(train_df, aes(x = chld, y = (..density..))) +
          geom_histogram() +
          ylab("Percentage") +
        facet_wrap(~donr, labeller = label_both) +
        theme_bw()

Plot3 <- ggplot(train_df, aes(x = hinc, y = (..density..))) +
          geom_histogram() +
        ylab("Percentage") +
        facet_wrap(~donr, labeller = label_both) +
        scale_x_continuous(breaks = seq(1,7,1)) +
        theme_bw()

Plot4 <- ggplot(train_df, aes(x = genf, y = (..density..))) +
            geom_histogram() +
          ylab("Percentage") +
          facet_wrap(~donr, labeller = label_both) +
          scale_x_continuous(breaks = seq(0,1,1)) +
          theme_bw()

Plot5 <- ggplot(train_df, aes(x = wrat, y = (..density..))) +
          geom_histogram() +
        ylab("Percentage") +
        facet_wrap(~donr, labeller = label_both) +
        scale_x_continuous(breaks = seq(0,9,1)) +
        theme_bw()

gridExtra::grid.arrange(Plot1, Plot2,
                        Plot3, Plot4,
                        Plot5, nrow = 3,
                        ncol = 2)

rm(Plot1)
rm(Plot2)
rm(Plot3)
rm(Plot4)
```

```
rm(Plot5)

#glimpse(train_df)

PLOT1 <- ggplot(train_df, aes(y = damt, x = avhv )) +
        geom_point() +
        facet_wrap(~donr) +
        stat_smooth(method = loess) +
        theme_bw()

PLOT2 <- ggplot(train_df, aes(y = damt, x = incm )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT3 <- ggplot(train_df, aes(y = damt, x = inca )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT4 <- ggplot(train_df, aes(y = damt, x = plow )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT5 <- ggplot(train_df, aes(y = damt, x = npro )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT6 <- ggplot(train_df, aes(y = damt, x = tgif )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT7 <- ggplot(train_df, aes(y = damt, x = lgif )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT8 <- ggplot(train_df, aes(y = damt, x = rgif )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()
```

```
PLOT9 <- ggplot(train_df, aes(y = damt, x = tdon )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()

PLOT10 <- ggplot(train_df, aes(y = damt, x = agif )) +
  geom_point() +
  facet_wrap(~donr) +
  stat_smooth(method = loess) +
  theme_bw()



gridExtra::grid.arrange(PLOT1, PLOT2, PLOT3, PLOT4, PLOT5,
                        PLOT6,nrow = 3, ncol = 2)
gridExtra::grid.arrange(PLOT7, PLOT8, PLOT9, PLOT10,
                        nrow = 2, ncol = 2)

rm(PLOT1);rm(PLOT2);rm(PLOT3);rm(PLOT4);rm(PLOT5)
rm(PLOT6);rm(PLOT7);rm(PLOT8);rm(PLOT9);rm(PLOT10)

corrplot::corrplot(cor(train_df[,-24]),
                   method = "number",
                   number.cex = .5)

charity_stg <- charity %>%
                 select(-inca, -npro, -rgif)


finalVar <- names(charity_stg)
# names(finalVar) <- "Variable Names"
knitr::kable(cbind(finalVar[1:5],
                   finalVar[6:10],
                   finalVar[11:15],
                   finalVar[16:20]),
             col.names = NULL,
             caption = "List of variables retained")

train_df <-  charity_stg %>%
               dplyr::filter(part == "train")

valid_df <-  charity_stg %>%
               dplyr::filter(part == "valid")

test_df <-  charity_stg %>%
               dplyr::filter(part == "test")

train_df_std <- data.frame(scale(train_df[,!names(train_df) %in%
                                           c("part", "ID",
                                             "donr", "damt")]),
                           train_df[,names(train_df) %in%
                                      c("ID", "donr", "damt")])
```

```
valid_df_std <- data.frame(scale(valid_df[,!names(valid_df) %in%
                                          c("part", "ID",
                                            "donr", "damt")]),
                           valid_df[,names(valid_df) %in%
                                      c("ID", "donr", "damt")])

test_df_std <- data.frame(scale(test_df[,!names(test_df) %in%
                                        c("part", "ID",
                                          "donr", "damt")]))

train_df_std_reg <- train_df_std[train_df_std$donr == 1,-19]
valid_df_std_reg <- valid_df_std[valid_df_std$donr == 1,-19]

rm(train_df)
rm(valid_df)
rm(test_df)
rm(finalVar)



############### RandomForest Code - Start ########################################
set.seed(1)
model_rf <-  randomForest(as.factor(donr)~ .
                          ,train_df_std[,!names(train_df_std)
                                        %in% c("ID", "damt")]
                          ,importance = T
                         # ,proximity = T
                          ,ntree = 800
                         ,parallel = TRUE)

pred_rf_class <-  predict(model_rf,
                          newdata = valid_df_std[,!names(valid_df_std) %in%
                                                   c("ID", "damt")])

ROC_model_rf <- roc(as.numeric(as.character(valid_df_std$donr)),
                    as.numeric(as.character(pred_rf_class)))

CfM_rf <- caret::confusionMatrix(as.factor(pred_rf_class),
                                 as.factor(valid_df_std$donr))

#Error Rate of Random Forest
plot(model_rf, main = "Errors by trees")
plot_sv_rf <- recordPlot()

## Model tune to find mtry
model_tune_rf <-  tuneRF(train_df_std[,c( -18,-19,-20)]
                          ,as.factor(train_df_std[,20])
                          ,stepFactor = 0.5,
                          ,ntreeTry = 800, plot = F, trace = F,
                         parallel = TRUE)


plot(model_tune_rf, type = "b", main = "OOBError by mtry")
```

```
plot_sv_tune_rf <- recordPlot()


##Calculate Maximum Profit and No. of mails
class_rf <- as.numeric(as.character(valid_df_std[order(pred_rf_class, decreasing=T),19]))
profit_rf<- cumsum(14.5*(class_rf)-2)
no_mail_rf <-  which.max(profit_rf)
Max_Profit_rf <-  max(profit_rf)


##Calculate Maximum Profit and No. of mails using classification table
CfM_rf
check_mail_rf <-  145 + 935
check_profit_rf <- 14.5*935 - 2*check_mail_rf



##Plots
#variable importance plot
 plot_sv_rf_varImp <-  importance(model_rf) %>%
                         data.frame() %>%
                         rownames_to_column(var = "rowname") %>%
                         arrange(MeanDecreaseAccuracy, rowname) %>%
                         ggplot(aes(x = reorder(rowname,-MeanDecreaseAccuracy),
                                    y = MeanDecreaseAccuracy, fill = "red")) +
                         geom_col () + #(stat = "identity") +
                         coord_flip() +
                         scale_fill_discrete(guide = FALSE) +
                         ggtitle("Variable Importance") +
                         xlab("") +
                         theme_bw()

##profit plot
 plot_sv_rf_profit <- ggplot(data.frame(profit_rf), aes(y = profit_rf, x = 1:length(profit_rf))) +
                        geom_line() +
                        ggtitle("Profit by Mails") +
                        xlab("No. of Mails") +
                        ylab("Profit") +
                        annotate("text", x = no_mail_rf -100, y = Max_Profit_rf + 300,
                                 label = paste0("Max Profit: ", Max_Profit_rf )) +
                        annotate("text", x = no_mail_rf + 150, y = Max_Profit_rf - 500,
                                 label = paste0("Mails: ", no_mail_rf )) +
                        theme_bw()


print(model_rf)

  knitr::kable(list(round(CfM_rf$overall,5),
                            CfM_rf$table),
           # col.names = c("Values",NULL),
               caption = "Confusion Matric Result"
  )
```

```
par(mfrow = c(1,2))
replayPlot(plot_sv_rf)
replayPlot(plot_sv_tune_rf)

par(mfrow = c(1,2))
plot.roc(ROC_model_rf, print.auc = T,
         col = "blue", main = "Model 1: Validation ROC Curve")
varImpPlot(model_rf, type = 2, main = "Variable Importance")
gridExtra::grid.arrange(plot_sv_rf_varImp,
                        plot_sv_rf_profit, nrow = 1)
# #CfM_rf
# knitr::kable(cbind("Mails" = no_mail_rf,
#                    "Max Profit" = Max_Profit_rf),
#                caption = "Maximum Profit and No. of mails")


knitr::kable(cbind("Mails" = check_mail_rf,
                   "Max Profit" = check_profit_rf),
             caption = "Profit and No. of mails by Classification Table")

############### RandomForest Code - END ##########################################

############### Boosting Code - START ############################################
set.seed(1)
model_gbm <-  gbm(as.character(donr)~ .
                          ,data = train_df_std[,!names(train_df_std) %in% c("ID", "damt")]
                          ,shrinkage=0.01
                          ,distribution = 'bernoulli'
                          ,cv.folds = 10
                          ,n.trees = 5000
                          ,interaction.depth = 4
                          ,verbose = F)

summary_gbm <- summary(model_gbm)
Plot_sv_gbm_sum <- recordPlot()

#finding the best iteration
best.iter <- gbm.perf(model_gbm, method = "cv")
Plot_sv_gbm <- recordPlot()
print(best.iter)

#get the predicted values
pred_gbm_class <- predict(model_gbm,
                          newdata = valid_df_std[,!names(valid_df_std) %in%
                                                   c("ID", "damt")],
                          n.trees = best.iter
                          ,type = "response")
#creating roc model
ROC_model_gbm <- roc(as.numeric(as.character(valid_df_std$donr)),
                     as.numeric(as.character(pred_gbm_class)))

#Calculate the profit
class_gbm <- as.numeric(as.character(valid_df_std[order(pred_gbm_class, decreasing=T),19]))
```

```r
profit_gbm<- cumsum(14.5*(class_gbm)-2)
no_mail_gbm <-  which.max(profit_gbm)
Max_Profit_gbm <-  max(profit_gbm)

#calcuate cutoff value
cutoff_gbm <- sort(pred_gbm_class, decreasing=T)[no_mail_gbm+1]
#calcuate c-hat value
chat_gbm <- ifelse(pred_gbm_class > cutoff_gbm, 1, 0)
#Creating confusion Matrix and profit using classificatin table
CfM_gbm <- caret::confusionMatrix(as.factor(chat_gbm), as.factor(valid_df_std$donr))
CfM_gbm
check_mail_gbm <-  224 + 990
check_profit_gbm <- 14.5*990 - 2*check_mail_gbm


#plots
##profit plot
 plot_sv_gbm_profit <- ggplot(data.frame(profit_gbm),
                              aes(y = profit_gbm, x = 1:length(profit_gbm))) +
                       geom_line() +
                       ggtitle("Profit by Mails") +
                       xlab("No. of Mails") +
                       ylab("Profit") +
                       annotate("text", x = no_mail_gbm -100, y = Max_Profit_gbm + 300,
                                label = paste0("Max Profit: ", Max_Profit_gbm )) +
                       annotate("text", x = no_mail_gbm + 150, y = Max_Profit_gbm - 500,
                                label = paste0("Mails: ", no_mail_gbm )) +
                       theme_bw()

 plot.roc(ROC_model_gbm, print.auc = T,
          col = "blue", main = "Model 2: Validation ROC Curve")
 plot_sv_gbm_roc <- recordPlot()


knitr::kable(summary_gbm,
             caption = "Relative influence statistics")

par(mfrow = c(1,2))
replayPlot(Plot_sv_gbm_sum)
replayPlot(Plot_sv_gbm)

par(mfrow = c(1,2))
replayPlot(plot_sv_gbm_roc)
plot_sv_gbm_profit

# knitr::kable(cbind("Mails" = no_mail_gbm,
#                    "Max Profit" = Max_Profit_gbm),
#              caption = "Maximum Profit and No. of mails")

knitr::kable(cbind("Mails" = check_mail_gbm,
                   "Max Profit" = check_profit_gbm),
             caption = "Profit and No. of mails by Classification Table")
```

```
############### Boosting Code - END ########################################

############### Support Vector Classifier Code - START ###############################
set.seed(1)
model_tune_svm_c <-  tune(svm, as.factor(donr) ~.
                       ,data = train_df_std[,!names(train_df_std) %in% c("ID", "damt")]
                      ,scale = FALSE
                      ,kernel = "linear"
                     ,ranges = list(cost =c(0.001, 0.01, 0.1, 1, 5, 10))
                     ,parallel = TRUE)

model_best_svm_c <-  model_tune_svm_c$best.model

## calculate fitted values
pred_svm_class <-  predict(model_best_svm_c,
                           newdata = valid_df_std[,!names(valid_df_std) %in% c("ID", "damt")]
                           ,type = "response")

#creating roc model
ROC_model_svm_c <- roc(as.numeric(as.character(valid_df_std$donr)),
                    as.numeric(as.character(pred_svm_class)))

##confusion Matrix
CfM_svm_c <- caret::confusionMatrix(as.factor(pred_svm_class),
                                    as.factor(valid_df_std$donr) )


## Calculate profit
class_svm_c <- as.numeric(as.character(valid_df_std[order(pred_svm_class, decreasing=T),19]))
profit_svm_c <- cumsum((14.5*class_svm_c)-2)

no_mail_svm_c <-  which.max(profit_svm_c)
Max_Profit_svm_c <-  max(profit_svm_c)

##Calculate profit using classification table
CfM_svm_c
 check_mail_svm_c <-  191 + 873
check_profit_svm_c <- 14.5*873 - 2*check_mail_svm_c

##profit plot
 plot_sv_svm_c_profit <- ggplot(data.frame(profit_svm_c), aes(y = profit_svm_c,
                                                              x = 1:length(profit_svm_c))) +
                         geom_line() +
                         ggtitle("Profit by Mails") +
                         xlab("No. of Mails") +
                         ylab("Profit") +
                         annotate("text", x = no_mail_svm_c, y = Max_Profit_svm_c + 300,
                                  label = Max_Profit_svm_c ) +
                         annotate("text", x = no_mail_svm_c , y = Max_Profit_svm_c - 500,
                                  label = no_mail_svm_c ) +
                         theme_bw()

plot.roc(ROC_model_svm_c, print.auc = T,
```

```
          col = "blue", main = "Model 3: Validation ROC Curve")
plot_sv_svm_c_roc <- recordPlot()

 #summary(model_tune_sum_c)
summary(model_best_svm_c)
par(mfrow = c(1,2))
replayPlot(plot_sv_svm_c_roc)
plot_sv_svm_c_profit

# knitr::kable(cbind("Mails" = no_mail_sum_c,
#                      "Max Profit" = Max_Profit_sum_c),
#              caption = "Profit and No. of mails")
knitr::kable(cbind("Mails" = check_mail_svm_c,
                   "Max Profit" = check_profit_svm_c),
             caption = "Profit and No. of mails by Classification Table")


################ Support Vector Classifier Code - END ###############################

################ Support Vector Machine Code - START ################################

set.seed(1)
model_tune_svm_m <-  tune(svm, as.factor(donr) ~.
                       ,data = train_df_std[,!names(train_df_std) %in% c("ID", "damt")]
                       ,scale = FALSE
                       ,kernel = "radial"
                       ,ranges = list(cost =c(0.001, 0.01, 0.1, 1, 5)
                                      ,gamma = c(0.1,0.5, 1, 2))
                       ,parallel = TRUE)

model_best_svm_m <-  model_tune_svm_m$best.model

##calculate predicted values
pred_svm_machine <-  predict(model_best_svm_m,
                             newdata = valid_df_std[,!names(valid_df_std) %in% c("ID", "damt")]
                           ,type = "response")

#creating roc model
ROC_model_svm_m <- roc(as.numeric(as.character(valid_df_std$donr)),
                    as.numeric(as.character(pred_svm_machine)))


# create confusionMatrix table
CfM_svm_m <- caret::confusionMatrix(as.factor(pred_svm_machine),
                                    as.factor(valid_df_std$donr))
#calculate profit
class_svm_m <- as.numeric(as.character(valid_df_std[order(pred_svm_machine, decreasing=T),19]))
profit_svm_m <- cumsum((14.5*class_svm_m)-2)
no_mail_svm_m <- which.max(profit_svm_m)
Max_Profit_svm_m <-  max(profit_svm_m)

#Calculate profit by classification table
CfM_svm_m
```

```r
check_mail_svm_m <-  160 + 904
check_profit_svm_m <- 14.5*904 - 2*check_mail_svm_m


##profit plot
 plot_sv_svm_m_profit <- ggplot(data.frame(profit_svm_m), aes(y = profit_svm_m,
                                                              x = 1:length(profit_svm_m))) +
                         geom_line() +
                         ggtitle("Profit by Mails") +
                         xlab("No. of Mails") +
                         ylab("Profit") +
                         annotate("text", x = no_mail_svm_m, y = Max_Profit_svm_m + 300,
                                  label = Max_Profit_svm_m ) +
                         annotate("text", x = no_mail_svm_m , y = Max_Profit_svm_m - 500,
                                  label = no_mail_svm_m ) +
                         theme_bw()

plot.roc(ROC_model_svm_m, print.auc = T,
          col = "blue", main = "Model 4: Validation ROC Curve")
plot_sv_svm_m_roc <- recordPlot()
# summary(model_tune_svm_m)
summary(model_best_svm_m)
par(mfrow = c(1,2))
replayPlot(plot_sv_svm_m_roc)
plot_sv_svm_m_profit

knitr::kable(cbind("Mails" = check_mail_svm_m,
                   "Max Profit" = check_profit_svm_m),
             caption = "Profit and No. of mails by Classification Table")


################ Support Vector Machine Code - END ################################

################ LASSO Code - END ################################

grid = 10^seq(10,-2,length = 100)
set.seed(1)
model_lasso <- cv.glmnet(x = as.matrix(train_df_std[,!names(train_df_std) %in%
                                                      c("ID","donr","damt")]) ,
                         y= as.factor(as.character(train_df_std$donr)),
                         family = "binomial",
                         parallel = TRUE,
                         alpha = 1,
                         lambda = grid,
                         standardize = FALSE)

model_best_lamb <- model_lasso$lambda.min

# calculate predicted values
pred_lasso_class <- predict(model_lasso,
                            newx = as.matrix(valid_df_std[,!names(train_df_std) %in%
                                                            c("ID","donr","damt")]),
                            s = model_best_lamb, type = "class")
```

```
#creating roc model
ROC_model_lasso <- roc(as.numeric(as.character(valid_df_std$donr)),
                       as.numeric(as.character(pred_lasso_class)))

#Confusion Matrix
CfM_lasso <- caret::confusionMatrix(as.factor(pred_lasso_class),
                                    as.factor(valid_df_std$donr))

#calculate profit
class_lasso_c <- as.numeric(as.character(valid_df_std[order(pred_lasso_class, decreasing=T),19]))
profit_lasso_c <- cumsum((14.5*class_lasso_c)-2)
no_mail_lasso_c <-  which.max(profit_lasso_c)
Max_Profit_lasso_c <-  max(profit_lasso_c)

#Calculate profit using classification tree
CfM_lasso
check_mail_lasso_c <-  201 + 865
check_profit_lasso_c <- 14.5*865 - 2*check_mail_lasso_c

##profit plot
 plot_sv_lasso_profit <- ggplot(data.frame(profit_lasso_c), aes(y = profit_lasso_c,
                                                                x = 1:length(profit_lasso_c))) +
                         geom_line() +
                         ggtitle("Profit by Mails") +
                         xlab("No. of Mails") +
                         ylab("Profit") +
                         annotate("text", x = no_mail_lasso_c, y = Max_Profit_lasso_c + 300,
                                  label = Max_Profit_lasso_c ) +
                         annotate("text", x = no_mail_lasso_c , y = Max_Profit_lasso_c - 500,
                                  label = no_mail_lasso_c ) +
                         theme_bw()

plot.roc(ROC_model_lasso, print.auc = T,
          col = "blue", main = "Model 5: Validation ROC Curve")
plot_sv_lasso_roc <- recordPlot()

par(mfrow = c(1,2))
replayPlot(plot_sv_lasso_roc)
plot_sv_lasso_profit

knitr::kable(cbind("Mails" = check_mail_lasso_c,
                   "Max Profit" = check_profit_lasso_c),
             caption = "Profit and No. of mails by Classification Table")


################ LASSO Code - END ################################
plot.roc(ROC_model_rf, col = "blue",
        # print.auc = TRUE,
         main = "ROC Curves for classification models")
lines.roc(ROC_model_gbm, col = "red")
lines.roc(ROC_model_svm_c, col = "green")
lines.roc(ROC_model_svm_m, col = "gold4")
lines.roc(ROC_model_lasso, col = "black")
```

```
legend("bottomright",
       legend = c(paste0("Random Forest:",
                         c(round(pROC::auc(ROC_model_rf),2))),
                  paste0("Boosting:",
                         c(round(pROC::auc(ROC_model_gbm),2))),
                  paste0("SV-Classifier:",
                         c(round(pROC::auc(ROC_model_svm_c),2))),
                  paste0("SV-Machine:",
                         c(round(pROC::auc(ROC_model_svm_m),2))),
                  paste0("Lasso:",
                         c(round(pROC::auc(ROC_model_lasso),2)))),
       col = c("blue", "red","green","gold4" ,"black"), lty = 1,cex = 1)


knitr::kable(rbind("Random Forest" = CfM_rf$overall[1],
                   "Boosting" = CfM_gbm$overall[1],
                   "SV-Classifier" = CfM_svm_c$overall[1],
                   "SV-Machine" = CfM_svm_m$overall[1],
                   "Lasso" = CfM_lasso$overall[1]),
             caption = "Accuracy values for each model")


############### Classification Model Selection - START ################################
knitr::kable(rbind(cbind("Random Forest",
                         "Mails" = check_mail_rf,
                         "Profit" = check_profit_rf),
                   cbind("Boosting",
                         "Mails" = check_mail_gbm,
                         "Profit" = check_profit_gbm),
                   cbind("Support Vector Classifier",
                         "Mails" = check_mail_svm_c,
                         "Profit" = check_profit_svm_c),
                   cbind("Support Vector Machine",
                         "Mails" = check_mail_svm_m,
                         "Profit" = check_profit_svm_m),
                   cbind("Lasso for Classification",
                         "Mails" = check_mail_lasso_c,
                         "Profit" = check_profit_lasso_c)),
             caption = "Number of mails and profit by each model")
############### Classification Model Selection - END ##################################

#get the predicted values using test data
test_donr <- predict(model_gbm,
                          newdata = test_df_std
                          ,n.trees = best.iter
                          ,type = "response")

# Oversampling adjustment for calculating number of mailings for test set
    #no_mail_gbm
rate_tr <- .1 # typical response rate is .1
rate_vr <- .5 # whereas validation response rate is .5
n_valid_c <- nrow(valid_df_std)
```

```
adj_test_1 <- (no_mail_gbm/n_valid_c)/(rate_vr/rate_tr) # adjustment for mail yes
adj_test_0 <- ((n_valid_c-no_mail_gbm)/n_valid_c)/((1-rate_vr)/(1-rate_tr)) # adjustment for mail no
adj_test <- adj_test_1/(adj_test_1+adj_test_0) # scale into a proportion

n_test <- nrow(test_df_std)
test_n_mail <-  round(n_test*adj_test, 0) # calculate number of mailings for test set
cutoff_test <- sort(test_donr, decreasing=T)[test_n_mail+1] # set cutoff based on n.mail.test
test_c_hat <- ifelse(test_donr>cutoff_test, 1,0) # mail to everyone above the cutoff

test_values <- table(test_c_hat)

knitr::kable(rbind("Mail = No" = test_values[[1]],
                                "Mail = Yes" = test_values[[2]]),
             caption = "Prediction on the mails",
             col.names = "# of Mails" )
################ RandomForest Code Regression - Start ###################################
set.seed(1)
reg_model_rf_damt <-  randomForest(damt ~ .
                          ,train_df_std_reg[,!names(train_df_std_reg) %in% c("ID")]
                          ,importance = T
                          ,ntree = 800)

#Error Rate of Random Forest
plot(reg_model_rf_damt, main = "Errors by trees")
plot_reg_rf_damt <- recordPlot()

      # ## Model tune to find mtry
      # reg_model_tune_rf_damt <-  tuneRF(train_df_std_reg[,c( -21,-22)]
      #                           ,as.factor(train_df_std_reg[,22])
      #                           ,stepFactor = 0.5,
      #                           ,ntreeTry = 800, plot = F, trace = F)
      #
      #
      # plot(reg_model_tune_rf_damt, type = "b") #, main = "OOBError by mtry")
      # plot_reg_tune_rf_damt <- recordPlot()


reg_pred_rf_damt <-  predict(reg_model_rf_damt,
                           newdata = valid_df_std_reg[,!names(valid_df_std_reg) %in%
                                                        c("ID", "donr")])

reg_mse_rf_damt <- mean((valid_df_std_reg$damt - reg_pred_rf_damt)^2)
reg_R_Squared_rf_damt <- cor(reg_pred_rf_damt, valid_df_std_reg$damt)^2


##Plots
#variable importance plot
 plot_reg_rf_damt_varImp <-  importance(reg_model_rf_damt) %>%
                          data.frame() %>%
                          rownames_to_column(var = "rowname") %>%
                          arrange(X.IncMSE, rowname) %>%
                          ggplot(aes(x = reorder(rowname,-X.IncMSE),
                                     y = X.IncMSE, fill = "red")) +
```

```
                          geom_col () + #(stat = "identity") +
                          coord_flip() +
                          scale_fill_discrete(guide = FALSE) +
                          ggtitle("Variable Importance") +
                          xlab("") +
                          theme_bw()
varImpPlot(reg_model_rf_damt, type = 2)
plot_reg_rf_damt_varImp2 <- recordPlot()

print(reg_model_rf_damt)
par(mfrow = c(1,2))
plot_reg_rf_damt_varImp2
plot_reg_rf_damt_varImp

knitr::kable(rbind("MSE" = reg_mse_rf_damt,
                   "Valid R-Squared" = reg_R_Squared_rf_damt),
             col.names = "Values",
             caption = "Random Forest: Metrics")
################ RandomForest Code Regression - END ###################################


################ Boosting Code Regression - Start #####################################
set.seed(1)
reg_model_gbm_damt <-  gbm(damt~ .
                          ,data = train_df_std_reg[,!names(train_df_std_reg) %in% c("ID")]
                          ,shrinkage=0.01
                          ,distribution = 'gaussian'
                          ,cv.folds = 10
                          ,n.trees = 5000
                          ,interaction.depth = 4
                          ,verbose = "CV")

reg_summary_gbm_damt <- summary(reg_model_gbm_damt)
Plot_reg_gbm_damt_sum <- recordPlot()


#finding the best iteration
reg_best_iter_damt <- gbm.perf(reg_model_gbm_damt, method = "cv")
Plot_reg_gbm_damt_iter <- recordPlot()
print(reg_best_iter_damt)

#get the predicted values
reg_pred_gbm_damt <- predict(reg_model_gbm_damt,
                          newdata = valid_df_std_reg[,!names(valid_df_std_reg) %in%
                                                       c("ID")],
                          n.trees = reg_best_iter_damt)

reg_mse_gbm_damt <- mean((valid_df_std_reg$damt - reg_pred_gbm_damt)^2)
reg_R_Squared_gbm_damt <- cor(reg_pred_gbm_damt, valid_df_std_reg$damt)^2

print(reg_model_gbm_damt)
reg_summary_gbm_damt
par(mfrow = c(1,2))
```

```
replayPlot(Plot_reg_gbm_damt_sum)
replayPlot(Plot_reg_gbm_damt_iter)
knitr::kable(rbind("MSE" = reg_mse_gbm_damt,
                   "Valid R-Squared" = reg_R_Squared_gbm_damt),
             col.names = "Values",
             caption = "Boosting: Metrics")

############### Boosting Code Regression - END ###################################


############### Ridge & Lasso Code Regression - START ##############################

grid = 10^seq(10,-2,length = 100)
## Ridge Regression
set.seed(1)
reg_model_ridge_damt <- cv.glmnet(x = as.matrix(train_df_std_reg[,!names(train_df_std_reg) %in%
                                                    c("ID","damt")]) ,
                          y= train_df_std_reg$damt,
                          family = "gaussian",
                          parallel = TRUE,
                          alpha = 0,
                          lambda = grid,
                          standardize = FALSE)

reg_best_lamb_ridge_damt <- reg_model_ridge_damt$lambda.min

# calculate predicted values
reg_pred_ridge_damt <- predict(reg_model_ridge_damt,
                           newx = as.matrix(valid_df_std_reg[,!names(valid_df_std_reg)
                                                         %in% c("ID","damt")]),
                           s = reg_best_lamb_ridge_damt, exact = T)

reg_mse_ridge_damt <- mean((valid_df_std_reg$damt - reg_pred_ridge_damt)^2)
reg_R_Squared_ridge_damt <- cor(reg_pred_ridge_damt, valid_df_std_reg$damt)^2

##Lasso Regression
set.seed(1)
reg_model_lasso_damt <- cv.glmnet(x = as.matrix(train_df_std_reg[,!names(train_df_std_reg) %in%
                                                    c("ID","damt")]) ,
                          y= train_df_std_reg$damt,
                          family = "gaussian",
                          parallel = TRUE,
                          alpha = 1,
                          lambda = grid,
                          standardize = FALSE)

reg_best_lamb_lasso_damt <- reg_model_lasso_damt$lambda.min

# calculate predicted values
reg_pred_lasso_damt <- predict(reg_model_lasso_damt,
                           newx = as.matrix(valid_df_std_reg[,!names(valid_df_std_reg)
                                                         %in% c("ID","damt")]),
                           s = reg_best_lamb_lasso_damt, exact = T)
```

```
reg_mse_lasso_damt <- mean((valid_df_std_reg$damt - reg_pred_lasso_damt)^2)
reg_R_Squared_lasso_damt <- cor(reg_pred_lasso_damt, valid_df_std_reg$damt)^2


par(mfrow = c(1,2))
plot(reg_model_ridge_damt)
plot(reg_model_lasso_damt)
knitr::kable(cbind(rbind("MSE" = reg_mse_ridge_damt,
                   "Valid R-Squared" = reg_R_Squared_ridge_damt[[1]]),
                   rbind("MSE" = reg_mse_lasso_damt,
                   "Valid R-Squared" = reg_R_Squared_lasso_damt[[1]])),
             col.names = c("Ridge", "Lasso"),
             caption = "Ridge & Lasso: Metrics")

############### Ridge & Lasso Code Regression - END ###################################

## PCR Regression
set.seed(1)
reg_model_pcr_damt <- pcr( damt ~. ,
                           data= train_df_std_reg[,!names(train_df_std_reg) %in%
                                                    c("ID")],
                           scale = FALSE,
                           parallel = TRUE,
                           validation = "CV")

reg_pred_pcr_damt <- predict(reg_model_pcr_damt,
                             newdata =  valid_df_std_reg[,!names(valid_df_std_reg) %in%
                                                   c("ID")],
                             ncomp = 14)

reg_mse_pcr_damt <- mean((valid_df_std_reg$damt - reg_pred_pcr_damt)^2)
reg_R_Squared_pcr_damt <- cor(reg_pred_pcr_damt, valid_df_std_reg$damt)^2
summary(reg_model_pcr_damt)
validationplot(reg_model_pcr_damt, val.type = "MSEP")
knitr::kable(rbind("MSE" = reg_mse_pcr_damt,
                   "Valid R-Squared" = reg_R_Squared_pcr_damt),
             col.names = "Values",
             caption = "PCR: Metrics")
reg_model_gam_damt_stg <- gam(damt ~
                                reg4
                                + lgif
                                + chld
                                + agif
                                + hinc
                                + tgif
                                + wrat
                                + incm
                              ,data =train_df_std_reg[,!names(train_df_std_reg)
                                                      %in%  c("ID")] )


reg_model_gamSTEP_damt <- step.Gam(reg_model_gam_damt_stg
                                   ,scope=list("reg4"=~1+reg4,
```

```
                                          "lgif"=~1+lgif+s(lgif,4)
                                         +s(lgif,6)+s(lgif,12),

                                         "chld"=~1+chld,

                                         "agif"=~1+agif+s(agif,4)
                                         +s(agif,6)+s(agif,12),

                                         "hinc"=~1+hinc,

                                         "tgif"=~1+tgif+s(tgif,4)
                                         +s(tgif,6)+s(tgif,12),

                                         "wrat"=~1+wrat,

                                         "incm"=~1+incm+s(incm,4)
                                         +s(incm,6)+s(incm,12)
                                         )
                                  ,parallel = TRUE
                                  ,trace = FALSE
                                  )

reg_pred_gamSTEP_damt <- predict(reg_model_gamSTEP_damt,
                                 newdata =  valid_df_std_reg[,!names(valid_df_std_reg) %in%
                                                c("ID")])

reg_mse_gamSTEP_damt <- mean((valid_df_std_reg$damt - reg_pred_gamSTEP_damt)^2)
reg_R_Squared_gamSTEP_damt <- cor(reg_pred_gamSTEP_damt, valid_df_std_reg$damt)^2

summary(reg_model_gamSTEP_damt)
knitr::kable(rbind("MSE" = reg_mse_gamSTEP_damt,
                   "Valid R-Squared" = reg_R_Squared_gamSTEP_damt),
             col.names = "Values",
             caption = "Stepwise GAM: Metrics")
################ Prediction Model Selection - START ################################
knitr::kable(rbind(cbind("Random Forest",
                         "MSE" = round(reg_mse_rf_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_rf_damt,3)),
                   cbind("Boosting",
                         "MSE" = round(reg_mse_gbm_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_gbm_damt,3)),
                   cbind("Ridge",
                         "MSE" = round(reg_mse_ridge_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_ridge_damt[[1]],3)),
                   cbind("Lasso",
                         "MSE" = round(reg_mse_lasso_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_lasso_damt[[1]],3)),
                   cbind("PCR",
                         "MSE" = round(reg_mse_pcr_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_pcr_damt,3)),
                   cbind("GAM",
                         "MSE" = round(reg_mse_gamSTEP_damt,3),
                         "Valid R-Squared" = round(reg_R_Squared_gamSTEP_damt,3))),
```

```
             caption = "MSE and R-Squared (validation set) for each model")
################ Prediction Model Selection - END ################################

test_y_hat <- predict(reg_model_gbm_damt,
                          newdata = test_df_std,
                          n.trees = reg_best_iter_damt)
test_y_hat_mean <- mean(test_y_hat)


final_submission <- data.frame(chat = test_c_hat,
                               yhat = test_y_hat)

write.csv(final_submission,
          file = "Singh_Gurjeet_Final_Project_Submission.csv",
          row.names = FALSE)

test_y_hat_mean_donor <- mean(final_submission[final_submission$chat ==1, ]$yhat)
knitr::kable(rbind(cbind("All yhat values",
                          round(test_y_hat_mean,2)),
                   cbind("Donors only (chat = 1)",
                          round(test_y_hat_mean_donor,2))),
             col.names = c("","Average donation Amount" ))
```