PALLAV ANAND

A report on i-360's Data Challenge

TABLE OF CONTENTS

Contents

INTRODUCTION

Introduction

Promoting fiscal responsibility in government is a key part of i360's mission. Determining which voters support reducing the U.S. national deficit versus those who support higher government spending will help i360 and partner organizations optimize the delivery of messages related to government spending to voters.

We have over a hundred anonymized features to predict whether a voter is likely to support higher government spending versus reducing the national debt when given a choice.

The data contains survey responses from 20,000 randomly selected individuals across the country. The survey question asked was:

I'm going to read two statements, please tell me with which one you agree MOST:

1. When it comes to the Federal budget, the government should place a higher priority on spending to improve the economy.

2. When it comes to the Federal budget, the government should place higher priority on reducing the national debt and deficit.

In the data, ID column is the respondent's unique ID. The State column is the respondent's state of residence. The SPENDINGRESPONSE column is the variable to predict, where

• "Spend to Improve Economy" indicates the respondent agrees that "When it comes to the Federal budget, the government should place a higher priority on spending to improve the economy."

• "Reduce National Debt and Deficit" indicates the respondent agrees that "When it comes to the Federal budget, the government should place higher priority on reducing the national debt and deficit."

In addition, you are given a file containing 29,231 records to be scored. This file does not contain a SPENDINGRESPONSE column.

Objective:

The task at hand is a **classification problem** where the target is a binary variable with two levels: "Spend to Improve Economy" and "Reduce National Debt and Deficit".

We have been provided a training set of 20000 records and we have a test set of 29,231 records.

The Dataset: A First Glimpse

We'll be using R for analyzing the dataset and building predictive models.

First, let's download the required dataset from the website

file1=read.csv('/Users/04pallav/Documents/i360/ProjectFiles/File1.csv', header=TRUE)
head(file1)

```
> head(file1)
     ID State                SPENDINGRESPONSE
1 18138    AK         Spend to Improve Economy
2 25537    AK         Spend to Improve Economy
3 27167    AK Reduce National Debt and Deficit
4 31853    AK Reduce National Debt and Deficit
5 67467    AK Reduce National Debt and Deficit
6 76622    AK Reduce National Debt and Deficit
```

file1 has 3 columns: ID which corresponds to respondent's unique ID, State of residence and the target variable

file2=read.csv('/Users/04pallav/Documents/i360/ProjectFiles/File2.csv',header=TRUE,na.strings ="NULL")
head(file2)

```
> head(file2)
     ID State   f1 f2 f3 f4       f5 f6       f7    f8       f9   f10   f11 f12 f13        f14        f15       f16       f17        f18        f19        f20
1 18138    AK AK01 63  E 61 55.33333 66 52.33333 49.25 48.29167 0.592 0.503   A   B 0.01145038 0.16412210 0.2137405 0.5076336 0.00000000 0.00000000 0.00000000
2 25537    AK AK01 55  E 61 55.33333 66 52.33333 49.25 48.29167 0.270 0.138   A   B 0.05135135 0.06216216 0.3513514 0.4486486 0.00000000 0.03783784 0.00000000
3 27167    AK AK01 49  D 61 55.33333 66 52.33333 49.25 48.29167 0.221 0.138   A   A 0.00000000 0.06866953 0.4549356 0.4163090 0.00000000 0.00000000 0.00000000
4 31853    AK AK01 67  D 61 55.33333 66 52.33333 49.25 48.29167 0.361 0.318   E   A 0.02227723 0.02722772 0.3514851 0.4603961 0.00000000 0.00000000 0.00000000
5 67467    AK AK01 58  M 61 55.33333 66 52.33333 49.25 48.29167 0.773 0.688   A   B 0.29253110 0.17427390 0.3257262 0.1639004 0.04564315 0.01867220 0.00000000
6 76622    AK AK01 70  E 61 55.33333 66 52.33333 49.25 48.29167 0.825    NA   A   A 0.13247860 0.14102560 0.3461539 0.3418804 0.09829060 0.00000000 0.01068376
         f21        f22        f23        f24        f25         f26        f27        f28        f29       f30        f31        f32        f33       f34
1 0.00000000 0.01145038 0.00000000 0.08778626 0.01908397 0.057251910 0.06870229 0.01145038 0.13358780 0.1374046 0.32442750 0.04580153 0.10305340 0.9010989
2 0.00000000 0.01351351 0.03243243 0.02972973 0.00000000 0.000000000 0.15135130 0.11621620 0.08378378 0.1135135 0.17837840 0.15675680 0.08648649 0.7248485
3 0.00000000 0.00000000 0.00000000 0.06866953 0.00000000 0.000000000 0.07725322 0.18454940 0.19313300 0.1030043 0.17167380 0.14163090 0.06008584 0.8306189
4 0.01237624 0.00990099 0.00990099 0.00990099 0.00000000 0.007425743 0.06435644 0.10148510 0.18564360 0.1905941 0.14851490 0.12128710 0.13861390 0.8379630
5 0.02282158 0.20539420 0.02074689 0.07468880 0.04149378 0.037344400 0.21576760 0.08091287 0.02904564 0.0560166 0.08298755 0.02489627 0.04356847 0.8253769
6 0.02350427 0.00000000 0.01709402 0.04914530 0.04059829 0.034188040 0.10042740 0.05555556 0.19017090 0.1944444 0.05555556 0.09188034 0.03846154 0.8384401
```

colnames(file2)

```
> colnames(file2)
  [1] "ID"    "State" "f1"    "f2"    "f3"    "f4"    "f5"    "f6"    "f7"    "f8"    "f9"    "f10"   "f11"   "f12"   "f13"   "f14"   "f15"   "f16"   "f17"
 [20] "f18"   "f19"   "f20"   "f21"   "f22"   "f23"   "f24"   "f25"   "f26"   "f27"   "f28"   "f29"   "f30"   "f31"   "f32"   "f33"   "f34"   "f35"   "f36"
 [39] "f37"   "f38"   "f39"   "f40"   "f41"   "f42"   "f43"   "f44"   "f45"   "f46"   "f47"   "f48"   "f49"   "f50"   "f51"   "f52"   "f53"   "f54"   "f55"
 [58] "f56"   "f57"   "f58"   "f59"   "f60"   "f61"   "f62"   "f63"   "f64"   "f65"   "f66"   "f67"   "f68"   "f69"   "f70"   "f71"   "f72"   "f73"   "f74"
 [77] "f75"   "f76"   "f77"   "f78"   "f79"   "f80"   "f81"   "f82"   "f83"   "f84"   "f85"   "f86"   "f87"   "f88"   "f89"   "f90"   "f91"   "f92"   "f93"
 [96] "f94"   "f95"   "f96"   "f97"   "f98"   "f99"   "f100"  "f101"  "f102"  "f103"  "f104"  "f105"  "f106"  "f107"  "f108"  "f109"  "f110"  "f111"  "f112"
[115] "f113"  "f114"  "f115"  "f116"  "f117"  "f118"  "f119"  "f120"  "f121"  "f122"  "f123"  "f124"  "f125"  "f126"  "f127"  "f128"  "f129"  "f130"  "f131"
[134] "f132"  "f133"  "f134"  "f135"  "f136"  "f137"  "f138"  "f139"  "f140"  "f141"  "f142"  "f143"  "f144"  "f145"  "f146"  "f147"
```

THE DATASET: A FIRST GLIMPSE

File2 has 20,000 rows and 147 features with ID and State columns. Let's merge file1 and file2 together on respondent's id and State.

merged=merge(x=file1, y = file2, by = c("ID","State"), all = TRUE)

```
> dim(merged)
[1] 20000   150
```

The resulting fie has 20,000 rows and 150 columns.

There are 147 anonymized features, one State column, one respondent ID and one target column.

Cleaning the Data

THE CLUES ARE IN THE SUMMARIES

After looking at the Summary statistics of the data, we can tell that there are many missing values. I read "NULL" values in the data as NA and these are visible in the summary.

We also realize that many of the anonymized features are numerical and many are categorical.

```
> summary(merged)
       ID                 State                             SPENDINGRESPONSE      f1                 f2                  f3              f4               f5
 Min.   :1.814e+04   CA     : 1090   Reduce National Debt and Deficit:13831   DE01   :  127   Min.   : 18.00   I        :7837   Min.   : 5.167   Min.   : 6.50
 1st Qu.:1.371e+08   PA     :  863   Spend to Improve Economy        : 6169   UT03   :  113   1st Qu.: 35.00   E        :4692   1st Qu.:29.083   1st Qu.:26.92
 Median :3.388e+08   NY     :  837                                            MT01   :  111   Median : 51.00   M        :3789   Median :56.500   Median :57.25
 Mean   :7.145e+08   TX     :  774                                            SD01   :  109   Mean   : 49.97   P        :2713   Mean   :52.446   Mean   :52.41
 3rd Qu.:7.426e+08   VA     :  747                                            MO02   :   95   3rd Qu.: 63.00   J        : 328   3rd Qu.:73.600   3rd Qu.:74.25
 Max.   :6.196e+09   OH     :  726                                            UT04   :   95   Max.   :104.00   (Other): 618    Max.   :95.833   Max.   :94.17
       f6                f7                f8                f9                f10                f11              f12          f13             f14                f15
 Min.   : 3.75   Min.   :10.04   Min.   :15.50   Min.   :18.00   Min.   :0.0010   Min.   :0.0000   A   :13034   A:10866   Min.   :0.00000   Min.   :0.0000
 1st Qu.:28.92   1st Qu.:23.75   1st Qu.:25.54   1st Qu.:28.42   1st Qu.:0.3610   1st Qu.:0.3490   B   : 2144   B: 9134   1st Qu.:0.06618   1st Qu.:0.0916
 Median :57.92   Median :47.04   Median :48.08   Median :48.71   Median :0.4900   Median :0.5070   C   :  394             Median :0.14631   Median :0.1630
 Mean   :52.08   Mean   :48.55   Mean   :48.36   Mean   :48.48   Mean   :0.4754   Mean   :0.4812   D   : 1063             Mean   :0.18709   Mean   :0.1756
 3rd Qu.:73.92   3rd Qu.:67.75   3rd Qu.:69.91   3rd Qu.:69.75   3rd Qu.:0.6140   3rd Qu.:0.6350   E   :  833             3rd Qu.:0.26635   3rd Qu.:0.2424
 Max.   :94.00   Max.   :90.88   Max.   :89.67   Max.   :90.79   Max.   :1.5430   Max.   :1.0000   NA's: 2532             Max.   :1.00000   Max.   :1.0000
```

Let's see the categorical variables.

Categorical Variables

```
> colnames(merged[,sapply(merged, is.factor)])
 [1] "State"            "SPENDINGRESPONSE" "f1"               "f3"               "f12"              "f13"              "f95"
 [9] "f97"              "f98"              "f99"              "f100"             "f101"             "f102"             "f103"
[17] "f110"             "f114"             "f115"             "f118"             "f119"             "f120"             "f121"
[25] "f126"
```

Next we want to see the percentage of missing values in the dataset.

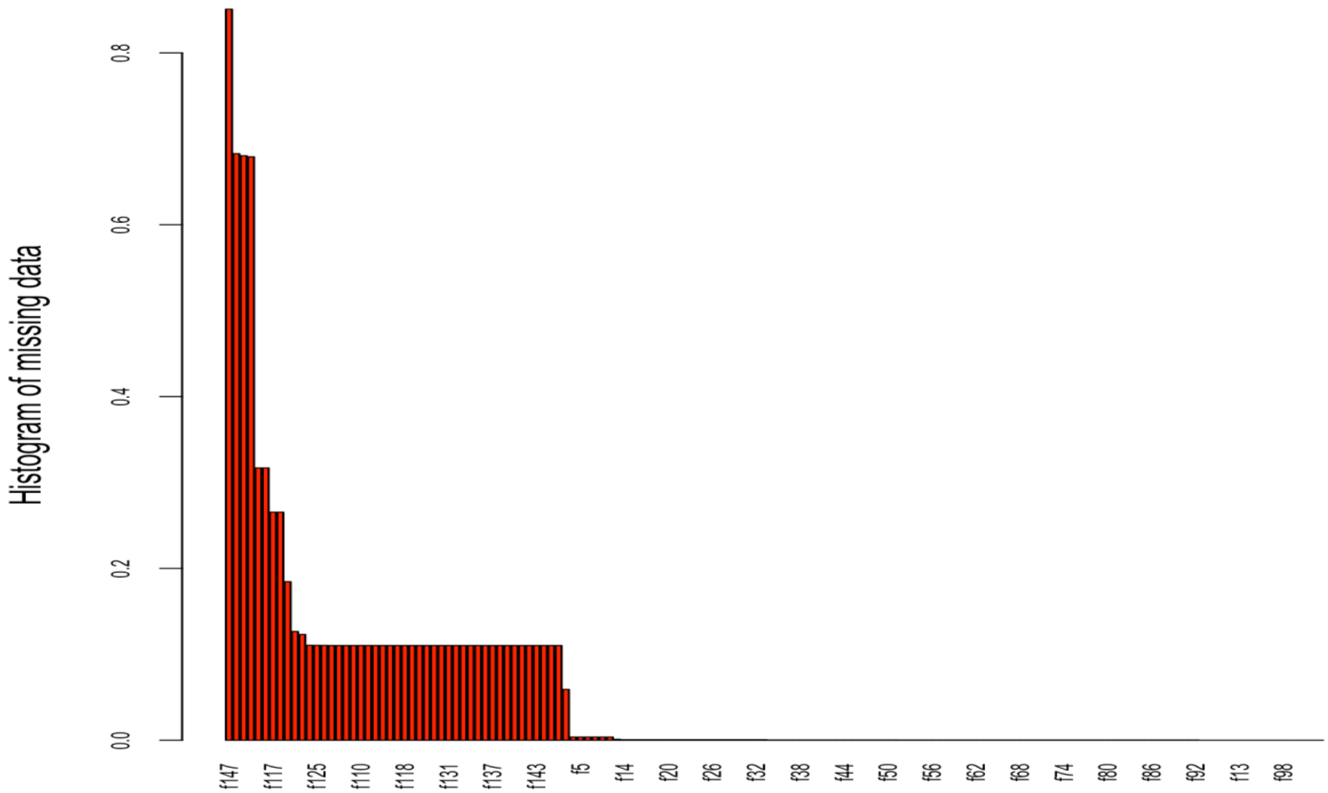
```
> mean(is.na(merged))*100
[1] 5.643967
```

We see that **around 6% of the data** has missing values. This can cause a problem while running our classification algorithms and will have to be dealt with.

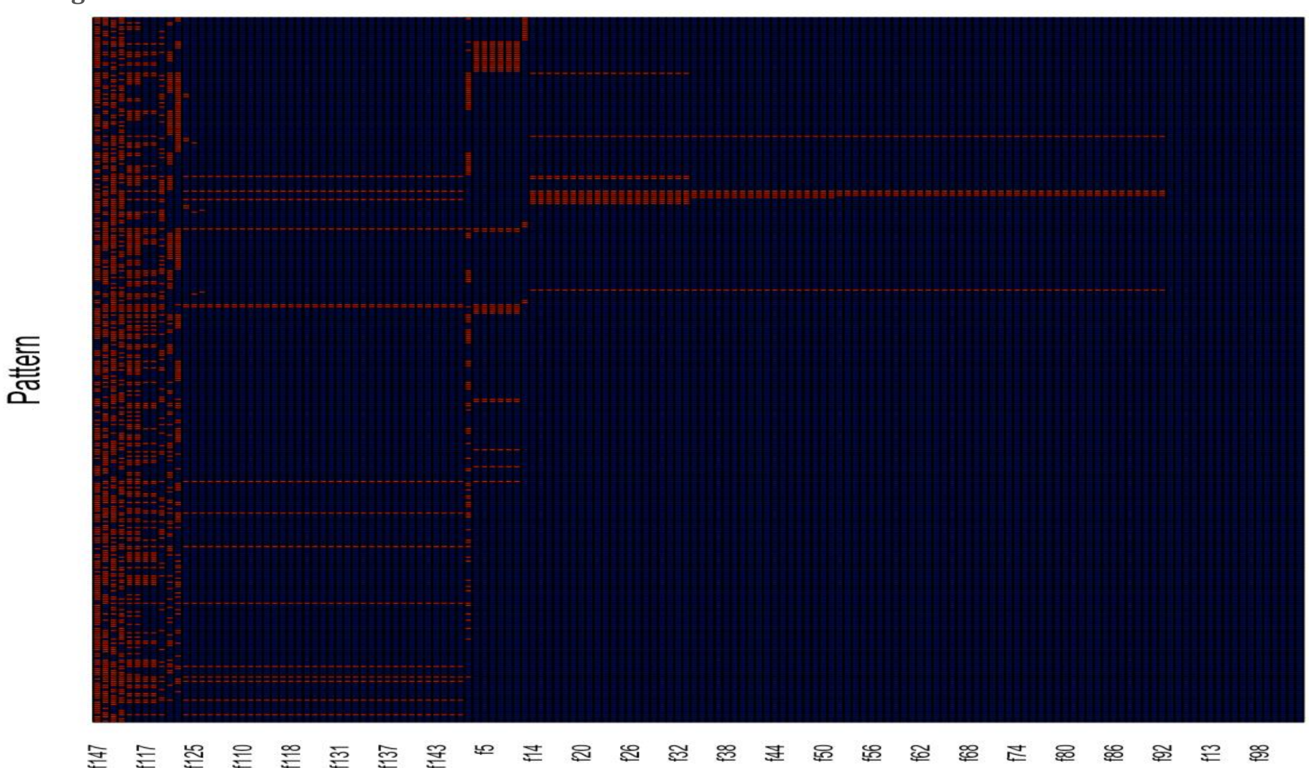
We choose the **mice package** for analysis of missing values and first try to identify if the data is **Missing Completely at Random or if there is a pattern to the missing values.**

md.pattern(merged)
aggr_plot <- aggr(merged, col=c('navyblue','red'),numbers=TRUE, sortVars=TRUE, labels=names(merged), cex.axis=.7, gap=3, ylab=c("Histogram of missing data","Pattern"))

CLEANING THE DATA



We see the missing values by different features. f147, f117 these are the columns with most missing values.



CLEANING THE DATA

We can definitely see some patterns in the missing data. So we will **try to dig deeper into the find what is missing before imputation.**

Let us see what percentage of data is **missing by each column**

pMiss <- function(x){sum(is.na(x))/length(x)*100}
apply(merged,2, pMiss) # by column
apply(merged,1,pMiss)   # by row

```
> sort(apply(merged,2,pMiss),decreasing=T)
      f147      f119      f122      f127      f115
    85.085    68.275    68.025    67.910    31.695
      f121      f117      f120      f126       f12
    31.695    26.545    26.545    18.455    12.660
       f10      f105      f125      f107      f104
    12.315    11.055    11.045    11.040    11.025
      f106      f108      f109      f110      f111
    11.025    11.025    11.025    11.025    11.025
      f112      f113      f114      f116      f118
    11.025    11.025    11.025    11.025    11.025
      f123      f124      f128      f129      f130
    11.025    11.025    11.025    11.025    11.025
      f131      f132      f133      f134      f135
    11.025    11.025    11.025    11.025    11.025
      f136      f137      f138      f139      f140
    11.025    11.025    11.025    11.025    11.025
      f141      f142      f143      f144      f145
    11.025    11.025    11.025    11.025    11.025
```

We see that features **f147, f119, f122, f127** has large number of missing values (>50%) We can choose to remove these columns from our analysis.

drops <- c("f147","f119","f127","f122")
cleanDF=merged[, !(names(merged) %in% drops)]

Let's have a look at the **missing values by rows.**

```
> sort(apply(merged,1,pMiss),decreasing=T)
  [1] 81.33333 58.66667 56.00000 54.66667 52.66667 52.66667 42.66667 42.00000 34.00000 34.00000 33.33333 33.33333
 [13] 33.33333 33.33333 33.33333 33.33333 32.66667 32.66667 32.66667 32.66667 32.66667 32.66667 32.66667 32.66667
 [25] 32.66667 32.66667 32.66667 32.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667
 [37] 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.66667 30.00000 30.00000 30.00000 30.00000
 [49] 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000
 [61] 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000
 [73] 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000 30.00000
```

We see that a lot of rows have more than 50% missing values. We can choose to remove these rows from our analysis.

I **chose a threshold of 25% for removing the rows** which means I removed the rows which have more than 25% missing values. I chose this threshold by iteration and sought to balance the data removed and the missing values we have in our data.

CLEANING THE DATA

cleanDF=cleanDF[-which(rowMeans(is.na(cleanDF)) > 0.25),]
*mean(is.na(cleanDF))*100*

```
> mean(is.na(cleanDF))*100
[1] 0.8208089
```

After removing some rows and columns the percentage of missing values in the data has dropped to **0.8% from 5.6%**

Data Imputation using MICE

We will use this cleaned dataset and further impute these missing values using the 'MICE' package.

I divided the features into numeric and categorical variables. I used "mean imputation" for the numeric features.

Mean imputation **has some disadvantages** that it standard deviations and the variance estimates tend to be underestimated. I would have preferred to use some other imputation methods from mice package like pmm (predictive mean matching) or tree based imputation methods but I chose to go with mean imputation because of the time and computational constraints.

```
numericsC$ID=NULL
NumImpute<- mice(numericsC, m=5,maxit=10,method='mean',seed=500,printFlag=FALSE)
numericsCImp=complete(NumImpute)
```

Further I chose to **impute categorical variables using 'cart' imputation** in MICE.

```
FacImpute<- mice(cleanDFNumImp, m=1,maxit=10,method='cart', seed=500,printFlag=T)
cleanDFNFImp=complete(FacImpute)
```

CLASSIFICATION MODELS

After cleaning and imputation our data set is ready for modelling. Let us try some classification models. I am using the **caret package** for tuning machine learning algorithms.

Logistic Regression

library(caret)   # Loading the caret package

I will drop some additional categorical variables which have too many levels. Having too many levels in categorical variables can cause problems in fitting models. We can **choose to include them later in our model by creating dummy variables.**

dropsl32 <- c("State","f1","f115","f121")

mergedl32=cleanDFNFImp[, !(names(cleanDFNFImp) %in% dropsl32)]

Creating a test and train set

I am using caret's createDataPartition function to split my data into 80/20 train and validation sets.

trainIndex <- createDataPartition(mydf$SPENDINGRESPONSE, p = .8,list = FALSE,times = 1)

train <- mydf[trainIndex,]

test <- mydf[-trainIndex,]

Cross Validation

I am using caret's trainControl method to perform 5-fold cross-validation in my training set.
I am repeating my analysis once.

ctrl <- trainControl(method = "repeatedcv",number=5,repeats = 1,savePredictions = TRUE, classProbs = TRUE,summaryFunction = twoClassSummary)

Training the model

For logistic regression I use the method "glm" and binomial family. I am using **ROC metric** to choose the best model. This will maximize the area under the curve while choosing the model.

mod_fit <- train(SPENDINGRESPONSE~., data=train,method="glm",metric="ROC",family="binomial",trControl = ctrl)

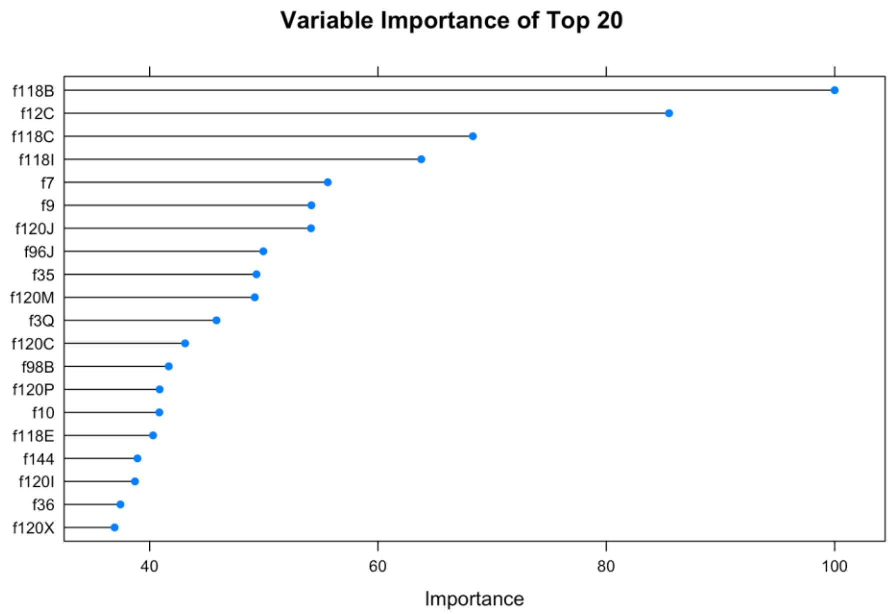
Results

```
> mod_fit
Generalized Linear Model

14232 samples
  140 predictor
    2 classes: 'Reduce', 'Spend'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 1 times)
Summary of sample sizes: 11385, 11386, 11385, 11386, 11386
Resampling results:

  ROC        Sens       Spec
  0.5803212  0.8949644  0.1543578
```



summary(mod_fit)

Test Set Results

pred = predict(mod_fit, newdata=test,type='raw')
head(pred)
confusionMatrix(pred, test$SPENDINGRESPONSE)

```
> confusionMatrix(pred, test$SPENDINGRESPONSE)
Confusion Matrix and Statistics

          Reference
Prediction Reduce Spend
    Reduce   2305   958
    Spend     162   132

               Accuracy : 0.6851
                 95% CI : (0.6696, 0.7004)
    No Information Rate : 0.6936
    P-Value [Acc > NIR] : 0.8662

                  Kappa : 0.0696
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9343
            Specificity : 0.1211
         Pos Pred Value : 0.7064
         Neg Pred Value : 0.4490
             Prevalence : 0.6936
         Detection Rate : 0.6480
   Detection Prevalence : 0.9173
      Balanced Accuracy : 0.5277

       'Positive' Class : Reduce
```

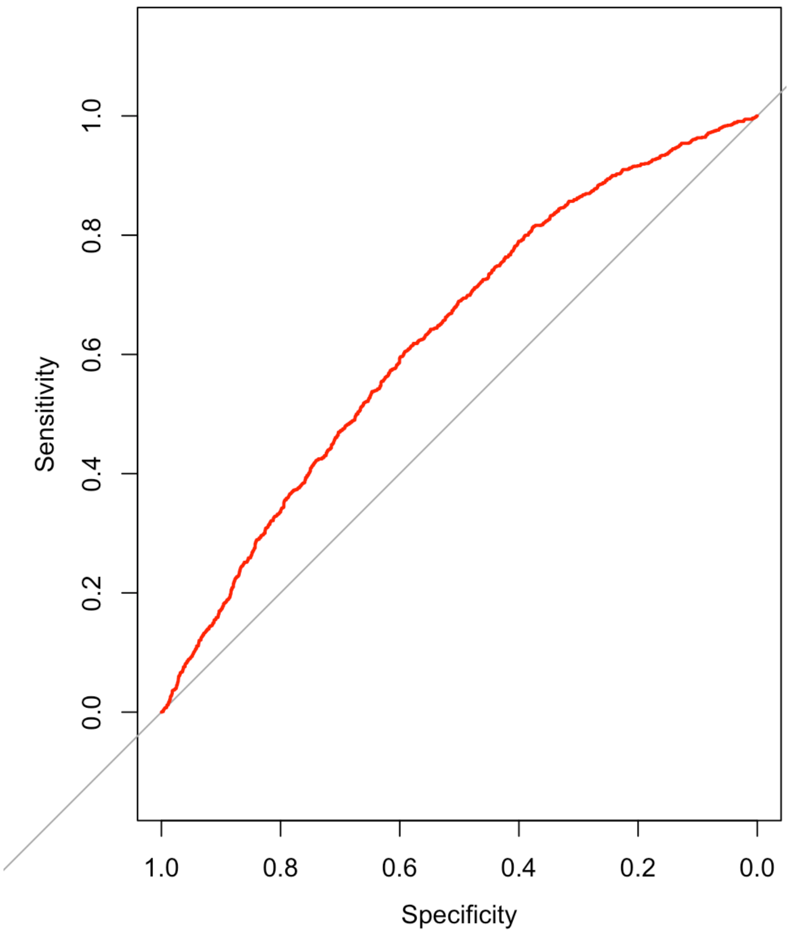
CLASSIFICATION MODELS

We have an accuracy of 68.5 % on the test set.

ROC Curve

##roc

```
predprob = predict(mod_fit, newdata=test,type='prob')
roc1=roc(test$SPENDINGRESPONSE,predprob[[2]])
plot(roc1, col = "red")
auc1=auc(roc1)
auc1
```




```
> auc1
Area under the curve: 0.6297
```

We have an AUC of 0.63 with this classifier.

CLASSIFICATION MODELS

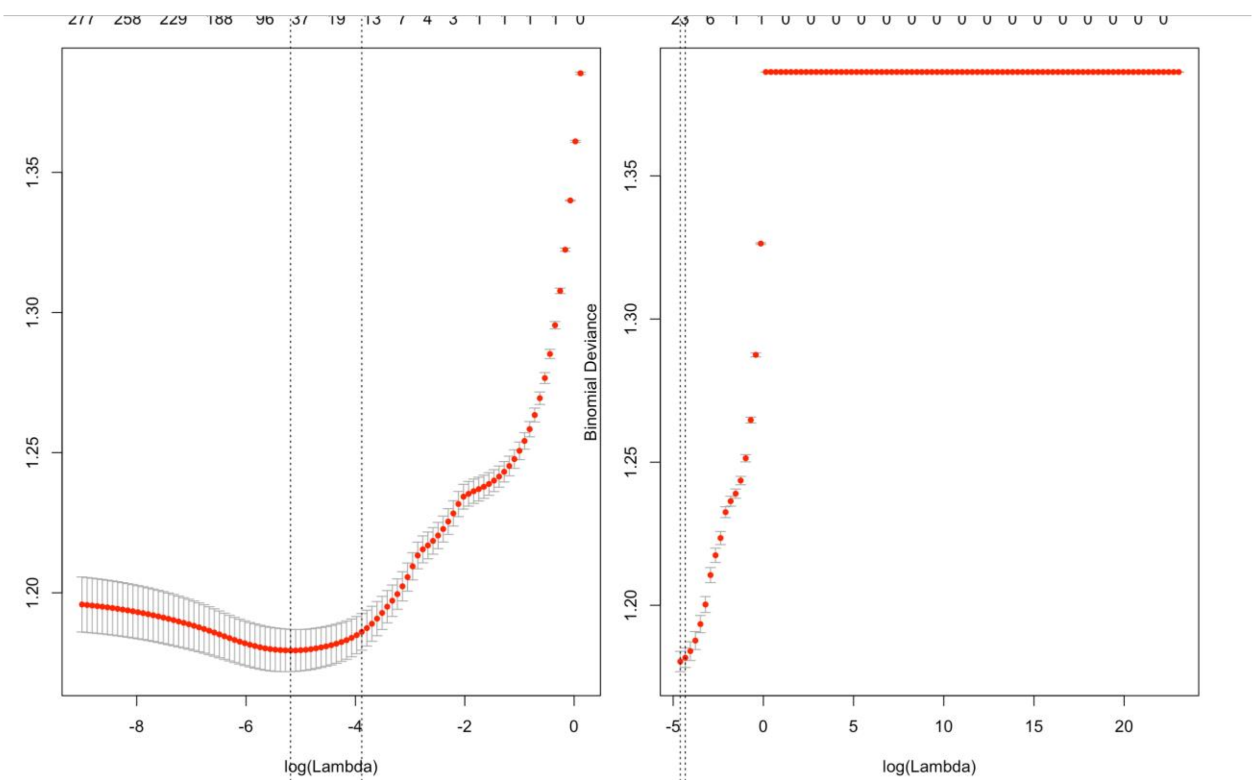
As we have a number of predictors is very high it might help us to fit a logistic model with regularization.
Ridge regression keeps all the predictors in the model whereas **lasso regression** has the effect of feature selection.
They are often useful in high dimensional scenarios where there are many variables.
I used both the models which had comparable results

Logistic Regression with Ridge and LASSO Penalty

```
x_train <- model.matrix( ~ .-1, train[,-1])
y_train_std=as.factor(train[,1])
grid=10^seq(10,-2,length=100)
lm = cv.glmnet(x=x_train,y = as.factor(train[,1]), intercept=FALSE ,family ="binomial",
          alpha=0, nfolds=7,lambda=grid)
best_lambda <- lm$lambda[which.min(lm$cvm)]
best_lambda
```

```
> best_lambda
[1] 0.1629751
```

```
plot(lm)
```



```
> std_ridge_logit <- glmnet(x_train, as.factor(train[,1]), family="binomial", alpha=0)
> SRL_pred_train <- predict(std_ridge_logit, x_train, type="class", s=best_lambda)
> confusionMatrix( SRL_pred_train,y_train_std)      ###training error
Confusion Matrix and Statistics

          Reference
Prediction Reduce Spend
    Reduce   9651  4103
    Spend     221   257

               Accuracy : 0.6962
```

```
> x_test <- model.matrix( ~ .-1, test[,-1])
> y_test_std=as.factor(test[,1])
> SRL_pred_test <- predict(std_ridge_logit, x_test, type="class", s=best_lambda)
> confusionMatrix( SRL_pred_test,y_test_std)
Confusion Matrix and Statistics

          Reference
Prediction Reduce Spend
    Reduce   2388  1041
    Spend      79    49

               Accuracy : 0.6851
```

With ridge regression the accuracy of the model remains the same.

LASSO

I tried the above for LASSO regression changing the alpha value to 1.

This gives a slightly improved accuracy of 69%

```
> confusionMatrix( SRL_pred_test,y_test_std)
Confusion Matrix and Statistics

          Reference
Prediction Reduce Spend
    Reduce   2417  1052
    Spend      50    38

               Accuracy : 0.6902
                 95% CI : (0.6747, 0.7054)
    No Information Rate : 0.6936
    P-Value [Acc > NIR] : 0.676
```

Collinearity in the features and Principal Least Squares

```
myDf=mergedl32[,sapply(mergedl32, is.numeric)]
myDf$ID=NULL
correlationMatrix <- cor(myDf)
# summarize the correlation matrix
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.95)
print(highlyCorrelated)
> highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.95,T,T)

 Combination row 2 and column 3 is above the cut-off, value = 0.979
         Flagging column 3
 Combination row 2 and column 4 is above the cut-off, value = 0.956
         Flagging column 2
 Combination row 3 and column 4 is above the cut-off, value = 0.957
         Flagging column 3
 Combination row 5 and column 6 is above the cut-off, value = 0.977
         Flagging column 6
 Combination row 5 and column 7 is above the cut-off, value = 0.973
         Flagging column 5
 Combination row 6 and column 7 is above the cut-off, value = 0.976
         Flagging column 6
 Combination row 31 and column 32 is above the cut-off, value = 0.955
         Flagging column 32
 Combination row 39 and column 40 is above the cut-off, value = 0.97
         Flagging column 39
 Combination row 95 and column 97 is above the cut-off, value = 0.98
         Flagging column 95
> print(highlyCorrelated)
[1] "f5"   "f8"   "f36"  "f4"   "f7"   "f43"  "f109"
```

The findCorrelation function from caret find the features which are highest correlated in the model. As we have multicollinearity in the model and a very high number of features, we can try dimension reduction techniques such as **Principal Least Squares or Principal Component Analysis.**

Here I have implemented the **Principal Least Squares approach** using the caret package**.**

```
grid <- expand.grid(ncomp=seq(1,120,10))
mod_fit <- train(SPENDINGRESPONSE~., data=train,method="pls",metric ="ROC",trControl =
ctrl,tuneGrid=grid)
```

I am testing the performance of the model based on the number of principal components.

CLASSIFICATION MODELS

```
> mod_fit
Partial Least Squares

14232 samples
  145 predictor
    2 classes: 'Reduce', 'Spend'

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 1 times)
Summary of sample sizes: 9488, 9489, 9487
Resampling results across tuning parameters:

  ncomp  ROC        Sens       Spec
    1    0.5365619  1.0000000  0.00000000
   11    0.5910556  0.9937194  0.01284320
   21    0.6325159  0.9583669  0.08921919
   31    0.6103100  0.9399311  0.10963356
   41    0.6061243  0.9225080  0.13944887
   51    0.6009043  0.9065032  0.16720091
   61    0.5948049  0.8941441  0.18325664
   71    0.5937818  0.8939416  0.18440243
   81    0.5932998  0.8932325  0.18187939
   91    0.5929826  0.8934350  0.18371483
  101    0.5929151  0.8927262  0.18165013
  111    0.5930448  0.8931314  0.18394392

ROC was used to select the optimal model using  the largest value.
The final value used for the model was ncomp = 21.
```

It seems that the ROC increases till 21 principal components and then starts decreases again. So the optimal complexity in this model is 21 principal components.

CLASSIFICATION MODELS

Random Forests

Next we try tree based methods for classification.
Random Forests are ensemble methods where many decorrelated trees are used to come up with the final class prediction.

library(randomForest)
ctrl <- trainControl(method = "repeatedcv",number=5,repeats =1,savePredictions = TRUE,classProbs = TRUE,summaryFunction = twoClassSummary)
mod_fit <- train(SPENDINGRESPONSE~., data=train,method="rf",metric ="ROC",trControl = ctrl,prox=TRUE,allowParallel=TRUE,verbose = TRUE)

Random Forest also gave me accuracy of around 0.7

Support Vector Machines

I implemented support vector machines as they are known to perform well in high dimensional spaces.
I used a radial basis kernel using the default parameters.

library(kernlab)
ctrl <- trainControl(method = "repeatedcv",number=5,repeats =1,savePredictions = TRUE,classProbs = TRUE,summaryFunction = twoClassSummary)
mod_fit <- train(SPENDINGRESPONSE~., data=train,method="svmRadial",metric ="ROC",trControl = ctrl,verbose = TRUE)#,tuneGrid=grid)

```
> mod_fit
Support Vector Machines with Radial Basis Function Kernel

14232 samples
  141 predictor
    2 classes: 'Reduce', 'Spend'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 1 times)
Summary of sample sizes: 11385, 11386, 11386, 11386, 11385
Resampling results across tuning parameters:

  C     ROC        Sens       Spec
  0.25  0.5556973  0.9892636  0.01972477
  0.50  0.5555200  0.9894656  0.01743119
  1.00  0.5517735  0.9894656  0.02018349

Tuning parameter 'sigma' was held constant at a value of 0.001987932
ROC was used to select the optimal model using  the largest value.
The final values used for the model were sigma = 0.001987932 and C = 0.25.
>
```

Better parameter tuning is needed to fit a better model

PREDICTION ON NEW DATA

The new data has to preprocessed for prediction using the same imputation methods used for the training data.

Mean imputation for Numerical variables
CART imputation for Categorical variables

I will use the LASSO model for prediction which I implemented earlier. I am now using the full data set for training the model and prediction on the test set.

My LASSO model has an effect of variable selection and **an accuracy of 69.3%**
We have tested the model for overfitting.

```
> std_lasso_logit <- glmnet(x_train, as.factor(train[,1]), family="binomial", alpha=1)
> SRL_pred_train <- predict(std_lasso_logit, x_train, type="class", s=best_lambda)
```

```
> confusionMatrix( SRL_pred_train,y_train_std)      ###training error
Confusion Matrix and Statistics

                                  Reference
Prediction                          Reduce National Debt and Deficit Spend to Improve Economy
  Reduce National Debt and Deficit                            12168                     5297
  Spend to Improve Economy                                      171                      153

               Accuracy : 0.6926
                 95% CI : (0.6858, 0.6994)
    No Information Rate : 0.6936
    P-Value [Acc > NIR] : 0.6186

                  Kappa : 0.0193
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.98614
            Specificity : 0.02807
         Pos Pred Value : 0.69671
         Neg Pred Value : 0.47222
             Prevalence : 0.69363
         Detection Rate : 0.68402
   Detection Prevalence : 0.98179
      Balanced Accuracy : 0.50711

       'Positive' Class : Reduce National Debt and Deficit
```

Code for prediction is similar and is provided in the Rscript

I saved the new predictions in file4.csv

```
f145 f146 f147 Probability                       Prediction
   1    1   NA   0.3042076 Reduce National Debt and Deficit
   4    4   NA   0.2892492 Reduce National Debt and Deficit
   1    1   NA   0.2805466 Reduce National Debt and Deficit
   3    2   NA   0.3370825 Reduce National Debt and Deficit
   0    0   NA   0.3305775 Reduce National Debt and Deficit
   4    6   NA   0.3703673 Reduce National Debt and Deficit
```

Looking Ahead

Though we tried many different classification models there are several areas where the models by which the model can be improved.

1. I have used **mean imputation** for numeric variables. This has several disadvantages such underestimation of standard deviance and variance. More **computationally expensive imputation** can be performed using several other imputation techniques present in MICE or HMISC packages in R.
2. **Extensive Hyper-parameter search**: A lot of models like SVM, LASSO depend on tuning the model correctly. It involves building a lot of models and comparing which models which is taking a lot of time on my computer given the large number of variables With more time and more powerful machines better model fitting can be performed.
3. **Forward or Backward Stepwise Selection**: Forward and Backward stepwise selection can be done This again takes a lot of time and is computationally expensive.
4. Variable Transformation: Several transformation of the predictor variables can be done like log transformation, square root transformation.
5. Advanced methods like **Gradient Boosting Machine or Neural Networks** can be implemented.
6. **Separate model for different states** is a possibility to be explored.