Porto Seguro’s Safe Driver Prediction

Manish Reddy Jannepally

December 3, 2017

# INTRODUCTION TO PORTO SEGURO’S SAFE DRIVER PREDICTION

## Problem Statement

Porto Seguro’s Insurance challenged Kagglers to build models that calculate the probability that a driver will file a claim in the next year. Hopefully, the models will help lower the cost for good drivers.

## Explanation of Case Study

Porto Seguro, one of Brazil’s largest auto and homeowner insurance companies wants to avoid the inaccuracies in car insurance company’s claim predictions which results in raise the cost of insurance for good drivers and reduce the price for bad ones.

The challenge is to build a model that predicts the probability that a driver will initiate an auto insurance claim in the next year.

## Data Description

In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc).

In addition, feature names include the postfix bin to indicate binary features and cat to indicate categorical features. Features without these designations are either continuous or ordinal.

Values of -1 indicate that the feature was missing from the observation. The target columns signifies whether or not a claim was filed for that policy holder.

## File Description

* train.csv contains the training data, where each row corresponds to a policy holder, and the target columns signifies that a claim was filed.
* test.csv contains the test data.
* sample\_submission.csv is submission file showing the correct format.

# Loading Required Libraries

The following libraries are used in this project…

library(dplyr) #data manipulation  
library(readr) #input/output  
library(tibble)#data wrangling  
library(data.table) #data manipulation  
library(forcats) #factor manipulation  
library(stringr) #string manipulation  
library(caret) #training and evaluation model  
library(MLmetrics) #Gini index  
library(ROSE) #over/under sampling

# PreProcessing the Data

According to the data description given, values “-1” indicate the features are missing from the observation. So, while importing the data I have considered “-1”,“-1.0” as NAs.

## [1] "C:/Users/janne/Desktop/Edwisor/Project 2"

##   
Read 0.0% of 595212 rows  
Read 20.2% of 595212 rows  
Read 40.3% of 595212 rows  
Read 60.5% of 595212 rows  
Read 80.6% of 595212 rows  
Read 595212 rows and 59 (of 59) columns from 0.108 GB file in 00:00:08

##   
Read 0.0% of 892816 rows  
Read 13.4% of 892816 rows  
Read 26.9% of 892816 rows  
Read 40.3% of 892816 rows  
Read 53.8% of 892816 rows  
Read 67.2% of 892816 rows  
Read 80.6% of 892816 rows  
Read 91.8% of 892816 rows  
Read 892816 rows and 58 (of 58) columns from 0.160 GB file in 00:00:12

## [1] 595212 59

## [1] 892816 58

The TRAIN dataset contains **595212 Observations and 59 Variables** (including target variable). The TEST dataset contains **892816 Observations and 58 Variables** (excluding target variable).

Let’s look at the structure and missing values of the datasets.

## Classes 'data.table' and 'data.frame': 595212 obs. of 59 variables:  
## $ id : int 7 9 13 16 17 19 20 22 26 28 ...  
## $ target : int 0 0 0 0 0 0 0 0 0 1 ...  
## $ ps\_ind\_01 : int 2 1 5 0 0 5 2 5 5 1 ...  
## $ ps\_ind\_02\_cat : int 2 1 4 1 2 1 1 1 1 1 ...  
## $ ps\_ind\_03 : int 5 7 9 2 0 4 3 4 3 2 ...  
## $ ps\_ind\_04\_cat : int 1 0 1 0 1 0 1 0 1 0 ...  
## $ ps\_ind\_05\_cat : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_06\_bin : int 0 0 0 1 1 0 0 1 0 0 ...  
## $ ps\_ind\_07\_bin : int 1 0 0 0 0 0 1 0 0 1 ...  
## $ ps\_ind\_08\_bin : int 0 1 1 0 0 0 0 0 1 0 ...  
## $ ps\_ind\_09\_bin : int 0 0 0 0 0 1 0 0 0 0 ...  
## $ ps\_ind\_10\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_11\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_12\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_13\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_14 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_15 : int 11 3 12 8 9 6 8 13 6 4 ...  
## $ ps\_ind\_16\_bin : int 0 0 1 1 1 1 1 1 1 0 ...  
## $ ps\_ind\_17\_bin : int 1 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_18\_bin : int 0 1 0 0 0 0 0 0 0 1 ...  
## $ ps\_reg\_01 : num 0.7 0.8 0 0.9 0.7 0.9 0.6 0.7 0.9 0.9 ...  
## $ ps\_reg\_02 : num 0.2 0.4 0 0.2 0.6 1.8 0.1 0.4 0.7 1.4 ...  
## $ ps\_reg\_03 : num 0.718 0.766 NA 0.581 0.841 ...  
## $ ps\_car\_01\_cat : int 10 11 7 7 11 10 6 11 10 11 ...  
## $ ps\_car\_02\_cat : int 1 1 1 1 1 0 1 1 1 0 ...  
## $ ps\_car\_03\_cat : int NA NA NA 0 NA NA NA 0 NA 0 ...  
## $ ps\_car\_04\_cat : int 0 0 0 0 0 0 0 0 0 1 ...  
## $ ps\_car\_05\_cat : int 1 NA NA 1 NA 0 1 0 1 0 ...  
## $ ps\_car\_06\_cat : int 4 11 14 11 14 14 11 11 14 14 ...  
## $ ps\_car\_07\_cat : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_car\_08\_cat : int 0 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_car\_09\_cat : int 0 2 2 3 2 0 0 2 0 2 ...  
## $ ps\_car\_10\_cat : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_car\_11\_cat : int 12 19 60 104 82 104 99 30 68 104 ...  
## $ ps\_car\_11 : int 2 3 1 1 3 2 2 3 3 2 ...  
## $ ps\_car\_12 : num 0.4 0.316 0.316 0.374 0.316 ...  
## $ ps\_car\_13 : num 0.884 0.619 0.642 0.543 0.566 ...  
## $ ps\_car\_14 : num 0.371 0.389 0.347 0.295 0.365 ...  
## $ ps\_car\_15 : num 3.61 2.45 3.32 2 2 ...  
## $ ps\_calc\_01 : num 0.6 0.3 0.5 0.6 0.4 0.7 0.2 0.1 0.9 0.7 ...  
## $ ps\_calc\_02 : num 0.5 0.1 0.7 0.9 0.6 0.8 0.6 0.5 0.8 0.8 ...  
## $ ps\_calc\_03 : num 0.2 0.3 0.1 0.1 0 0.4 0.5 0.1 0.6 0.8 ...  
## $ ps\_calc\_04 : int 3 2 2 2 2 3 2 1 3 2 ...  
## $ ps\_calc\_05 : int 1 1 2 4 2 1 2 2 1 2 ...  
## $ ps\_calc\_06 : int 10 9 9 7 6 8 8 7 7 8 ...  
## $ ps\_calc\_07 : int 1 5 1 1 3 2 1 1 3 2 ...  
## $ ps\_calc\_08 : int 10 8 8 8 10 11 8 6 9 9 ...  
## $ ps\_calc\_09 : int 1 1 2 4 2 3 3 1 4 1 ...  
## $ ps\_calc\_10 : int 5 7 7 2 12 8 10 13 11 11 ...  
## $ ps\_calc\_11 : int 9 3 4 2 3 4 3 7 4 3 ...  
## $ ps\_calc\_12 : int 1 1 2 2 1 2 0 1 2 5 ...  
## $ ps\_calc\_13 : int 5 1 7 4 1 0 0 3 1 0 ...  
## $ ps\_calc\_14 : int 8 9 7 9 3 9 10 6 5 6 ...  
## $ ps\_calc\_15\_bin: int 0 0 0 0 0 0 0 1 0 0 ...  
## $ ps\_calc\_16\_bin: int 1 1 1 0 0 1 1 0 1 1 ...  
## $ ps\_calc\_17\_bin: int 1 1 1 0 0 0 0 1 0 0 ...  
## $ ps\_calc\_18\_bin: int 0 0 0 0 1 1 0 0 0 0 ...  
## $ ps\_calc\_19\_bin: int 0 1 1 0 1 1 1 1 0 1 ...  
## $ ps\_calc\_20\_bin: int 1 0 0 0 0 1 0 0 1 0 ...  
## - attr(\*, ".internal.selfref")=<externalptr>

## Classes 'data.table' and 'data.frame': 892816 obs. of 58 variables:  
## $ id : int 0 1 2 3 4 5 6 8 10 11 ...  
## $ ps\_ind\_01 : int 0 4 5 0 5 0 0 0 0 1 ...  
## $ ps\_ind\_02\_cat : int 1 2 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_03 : int 8 5 3 6 7 6 3 0 7 6 ...  
## $ ps\_ind\_04\_cat : int 1 1 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_05\_cat : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_06\_bin : int 0 0 0 1 0 1 0 1 0 0 ...  
## $ ps\_ind\_07\_bin : int 1 0 0 0 0 0 1 0 1 0 ...  
## $ ps\_ind\_08\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_09\_bin : int 0 1 1 0 1 0 0 0 0 1 ...  
## $ ps\_ind\_10\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_11\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_12\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_13\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_14 : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_ind\_15 : int 12 5 10 4 4 10 11 7 6 7 ...  
## $ ps\_ind\_16\_bin : int 1 1 0 1 1 1 0 1 1 0 ...  
## $ ps\_ind\_17\_bin : int 0 0 0 0 0 0 1 0 0 1 ...  
## $ ps\_ind\_18\_bin : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ ps\_reg\_01 : num 0.5 0.9 0.4 0.1 0.9 0.9 0.1 0.9 0.4 0.9 ...  
## $ ps\_reg\_02 : num 0.3 0.5 0 0.2 0.4 0.5 0.1 1.1 0 1 ...  
## $ ps\_reg\_03 : num 0.61 0.771 0.916 NA 0.818 ...  
## $ ps\_car\_01\_cat : int 7 4 11 7 11 9 6 7 11 11 ...  
## $ ps\_car\_02\_cat : int 1 1 1 1 1 1 1 1 0 0 ...  
## $ ps\_car\_03\_cat : int NA NA NA NA NA NA NA NA 1 NA ...  
## $ ps\_car\_04\_cat : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ ps\_car\_05\_cat : int NA 0 NA NA NA NA 0 NA 0 NA ...  
## $ ps\_car\_06\_cat : int 1 11 14 1 11 11 1 11 2 4 ...  
## $ ps\_car\_07\_cat : int 1 1 1 1 1 0 1 1 NA 1 ...  
## $ ps\_car\_08\_cat : int 1 1 1 1 1 0 1 1 0 1 ...  
## $ ps\_car\_09\_cat : int 2 0 2 2 2 2 0 2 0 2 ...  
## $ ps\_car\_10\_cat : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_car\_11\_cat : int 65 103 29 40 101 11 10 103 104 104 ...  
## $ ps\_car\_11 : int 1 1 3 2 3 2 2 3 2 2 ...  
## $ ps\_car\_12 : num 0.316 0.316 0.4 0.374 0.374 ...  
## $ ps\_car\_13 : num 0.67 0.606 0.896 0.652 0.813 ...  
## $ ps\_car\_14 : num 0.352 0.358 0.398 0.381 0.385 ...  
## $ ps\_car\_15 : num 3.46 2.83 3.32 2.45 3.32 ...  
## $ ps\_calc\_01 : num 0.1 0.4 0.6 0.1 0.9 0.7 0.9 0.8 0.9 0 ...  
## $ ps\_calc\_02 : num 0.8 0.5 0.6 0.5 0.6 0.9 0.8 0.9 0.3 0.9 ...  
## $ ps\_calc\_03 : num 0.6 0.4 0.6 0.5 0.8 0.4 0.8 0.5 0 0.7 ...  
## $ ps\_calc\_04 : int 1 3 2 2 3 2 1 2 2 2 ...  
## $ ps\_calc\_05 : int 1 3 3 1 4 1 1 2 2 1 ...  
## $ ps\_calc\_06 : int 6 8 7 7 7 9 7 8 9 7 ...  
## $ ps\_calc\_07 : int 3 4 4 3 1 5 3 4 7 1 ...  
## $ ps\_calc\_08 : int 6 10 6 12 10 9 9 11 9 9 ...  
## $ ps\_calc\_09 : int 2 2 3 1 4 4 5 2 0 1 ...  
## $ ps\_calc\_10 : int 9 7 12 13 12 12 6 8 10 11 ...  
## $ ps\_calc\_11 : int 1 2 4 5 4 8 2 3 5 6 ...  
## $ ps\_calc\_12 : int 1 0 0 1 0 1 0 1 1 1 ...  
## $ ps\_calc\_13 : int 1 3 2 0 0 4 4 4 4 6 ...  
## $ ps\_calc\_14 : int 12 10 4 5 4 9 6 9 6 10 ...  
## $ ps\_calc\_15\_bin: int 0 0 0 1 0 1 1 0 0 0 ...  
## $ ps\_calc\_16\_bin: int 1 0 0 0 1 0 1 1 0 1 ...  
## $ ps\_calc\_17\_bin: int 1 1 0 1 1 1 0 0 1 1 ...  
## $ ps\_calc\_18\_bin: int 0 1 0 0 0 0 0 0 0 0 ...  
## $ ps\_calc\_19\_bin: int 0 0 0 0 0 1 0 0 0 0 ...  
## $ ps\_calc\_20\_bin: int 1 1 0 0 1 0 0 0 0 0 ...  
## - attr(\*, ".internal.selfref")=<externalptr>

## [1] 846458

## [1] 1270295

The structure of the datasets says the data types of the variables are either numerical or integer.As per the data description,features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc) and also the binary and categorical variables are postfixed as \_bin and \_cat respectively. Remaining features which are not tagged are either continuous or ordinal.

So, we have to convert the data types according to the given data description. I have done this step by row binding TEST and TRAIN sets together as we can do the pre-processing to the whole data at once.

## [1] 1488028 60

I have added a ‘target’ column to the TEST set (to make TEST set have same number of variables) and a ‘data’ column to the TEST and TRAIN sets which can be used to identify the test and train observations.

The combined dataset has **1488028 Observations and 60 Variables**. I have formed a csv file named var\_groups.csv with the names of variables as one column and another column with type of them according to the data description. We use this file to convert the data types of the variables as per the data discription.

I have used the below code to convert the data types.

var\_groups = fread("var\_groups.csv")  
names = intersect(colnames(combined\_data), var\_groups[["names"]])  
for(var\_name in names){  
   
 var\_type = subset(var\_groups, names %in% var\_name, select=type)  
 if(var\_type == "numeric")  
 combined\_data[,var\_name] = as.numeric(combined\_data[,var\_name])  
 else if(var\_type == "binary" || var\_type == "categorical")  
 combined\_data[,var\_name] <- as.factor(combined\_data[,var\_name])  
 else if(var\_type == "ordinal")  
 combined\_data[,var\_name] <- as.ordered(combined\_data[,var\_name])  
}

Now, Let’s see the structure of the data.

## 'data.frame': 1488028 obs. of 60 variables:  
## $ id : num 7 9 13 16 17 19 20 22 26 28 ...  
## $ data : chr "train" "train" "train" "train" ...  
## $ target : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ ps\_ind\_01 : num 2 1 5 0 0 5 2 5 5 1 ...  
## $ ps\_ind\_02\_cat : Factor w/ 4 levels "1","2","3","4": 2 1 4 1 2 1 1 1 1 1 ...  
## $ ps\_ind\_03 : Ord.factor w/ 12 levels "0"<"1"<"2"<"3"<..: 6 8 10 3 1 5 4 5 4 3 ...  
## $ ps\_ind\_04\_cat : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 1 ...  
## $ ps\_ind\_05\_cat : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_06\_bin : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 1 2 1 1 ...  
## $ ps\_ind\_07\_bin : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 1 1 2 ...  
## $ ps\_ind\_08\_bin : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 1 2 1 ...  
## $ ps\_ind\_09\_bin : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...  
## $ ps\_ind\_10\_bin : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_11\_bin : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_12\_bin : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_13\_bin : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_14 : Ord.factor w/ 5 levels "0"<"1"<"2"<"3"<..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_15 : Ord.factor w/ 14 levels "0"<"1"<"2"<"3"<..: 12 4 13 9 10 7 9 14 7 5 ...  
## $ ps\_ind\_16\_bin : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 2 2 1 ...  
## $ ps\_ind\_17\_bin : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...  
## $ ps\_ind\_18\_bin : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...  
## $ ps\_reg\_01 : num 0.7 0.8 0 0.9 0.7 0.9 0.6 0.7 0.9 0.9 ...  
## $ ps\_reg\_02 : num 0.2 0.4 0 0.2 0.6 1.8 0.1 0.4 0.7 1.4 ...  
## $ ps\_reg\_03 : num 0.718 0.766 NA 0.581 0.841 ...  
## $ ps\_car\_01\_cat : Factor w/ 12 levels "0","1","2","3",..: 11 12 8 8 12 11 7 12 11 12 ...  
## $ ps\_car\_02\_cat : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 2 2 1 ...  
## $ ps\_car\_03\_cat : Factor w/ 2 levels "0","1": NA NA NA 1 NA NA NA 1 NA 1 ...  
## $ ps\_car\_04\_cat : Factor w/ 10 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 2 ...  
## $ ps\_car\_05\_cat : Factor w/ 2 levels "0","1": 2 NA NA 2 NA 1 2 1 2 1 ...  
## $ ps\_car\_06\_cat : Factor w/ 18 levels "0","1","2","3",..: 5 12 15 12 15 15 12 12 15 15 ...  
## $ ps\_car\_07\_cat : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ps\_car\_08\_cat : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...  
## $ ps\_car\_09\_cat : Factor w/ 5 levels "0","1","2","3",..: 1 3 3 4 3 1 1 3 1 3 ...  
## $ ps\_car\_10\_cat : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...  
## $ ps\_car\_11\_cat : Factor w/ 104 levels "1","2","3","4",..: 12 19 60 104 82 104 99 30 68 104 ...  
## $ ps\_car\_11 : Ord.factor w/ 4 levels "0"<"1"<"2"<"3": 3 4 2 2 4 3 3 4 4 3 ...  
## $ ps\_car\_12 : num 0.4 0.316 0.316 0.374 0.316 ...  
## $ ps\_car\_13 : num 0.884 0.619 0.642 0.543 0.566 ...  
## $ ps\_car\_14 : num 0.371 0.389 0.347 0.295 0.365 ...  
## $ ps\_car\_15 : num 3.61 2.45 3.32 2 2 ...  
## $ ps\_calc\_01 : num 0.6 0.3 0.5 0.6 0.4 0.7 0.2 0.1 0.9 0.7 ...  
## $ ps\_calc\_02 : num 0.5 0.1 0.7 0.9 0.6 0.8 0.6 0.5 0.8 0.8 ...  
## $ ps\_calc\_03 : num 0.2 0.3 0.1 0.1 0 0.4 0.5 0.1 0.6 0.8 ...  
## $ ps\_calc\_04 : Ord.factor w/ 6 levels "0"<"1"<"2"<"3"<..: 4 3 3 3 3 4 3 2 4 3 ...  
## $ ps\_calc\_05 : Ord.factor w/ 7 levels "0"<"1"<"2"<"3"<..: 2 2 3 5 3 2 3 3 2 3 ...  
## $ ps\_calc\_06 : Ord.factor w/ 11 levels "0"<"1"<"2"<"3"<..: 11 10 10 8 7 9 9 8 8 9 ...  
## $ ps\_calc\_07 : Ord.factor w/ 10 levels "0"<"1"<"2"<"3"<..: 2 6 2 2 4 3 2 2 4 3 ...  
## $ ps\_calc\_08 : Ord.factor w/ 12 levels "1"<"2"<"3"<"4"<..: 10 8 8 8 10 11 8 6 9 9 ...  
## $ ps\_calc\_09 : Ord.factor w/ 8 levels "0"<"1"<"2"<"3"<..: 2 2 3 5 3 4 4 2 5 2 ...  
## $ ps\_calc\_10 : Ord.factor w/ 26 levels "0"<"1"<"2"<"3"<..: 6 8 8 3 13 9 11 14 12 12 ...  
## $ ps\_calc\_11 : Ord.factor w/ 21 levels "0"<"1"<"2"<"3"<..: 10 4 5 3 4 5 4 8 5 4 ...  
## $ ps\_calc\_12 : Ord.factor w/ 12 levels "0"<"1"<"2"<"3"<..: 2 2 3 3 2 3 1 2 3 6 ...  
## $ ps\_calc\_13 : Ord.factor w/ 16 levels "0"<"1"<"2"<"3"<..: 6 2 8 5 2 1 1 4 2 1 ...  
## $ ps\_calc\_14 : Ord.factor w/ 25 levels "0"<"1"<"2"<"3"<..: 9 10 8 10 4 10 11 7 6 7 ...  
## $ ps\_calc\_15\_bin: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...  
## $ ps\_calc\_16\_bin: Factor w/ 2 levels "0","1": 2 2 2 1 1 2 2 1 2 2 ...  
## $ ps\_calc\_17\_bin: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 2 1 1 ...  
## $ ps\_calc\_18\_bin: Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 1 1 ...  
## $ ps\_calc\_19\_bin: Factor w/ 2 levels "0","1": 1 2 2 1 2 2 2 2 1 2 ...  
## $ ps\_calc\_20\_bin: Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 1 2 1 ...

Let’s explore the missing values of the data. We have already seen there are **846458** and **1270295** missing values in TRAIN and TEST data respectively. Below are the column wise missing values in combined data (both TEST and TRAIN)

## colSums(is.na(combined\_data))  
## id 0  
## data 0  
## target 0  
## ps\_ind\_01 0  
## ps\_ind\_02\_cat 523  
## ps\_ind\_03 0  
## ps\_ind\_04\_cat 228  
## ps\_ind\_05\_cat 14519  
## ps\_ind\_06\_bin 0  
## ps\_ind\_07\_bin 0  
## ps\_ind\_08\_bin 0  
## ps\_ind\_09\_bin 0  
## ps\_ind\_10\_bin 0  
## ps\_ind\_11\_bin 0  
## ps\_ind\_12\_bin 0  
## ps\_ind\_13\_bin 0  
## ps\_ind\_14 0  
## ps\_ind\_15 0  
## ps\_ind\_16\_bin 0  
## ps\_ind\_17\_bin 0  
## ps\_ind\_18\_bin 0  
## ps\_reg\_01 0  
## ps\_reg\_02 0  
## ps\_reg\_03 269456  
## ps\_car\_01\_cat 267  
## ps\_car\_02\_cat 10  
## ps\_car\_03\_cat 1028142  
## ps\_car\_04\_cat 0  
## ps\_car\_05\_cat 666910  
## ps\_car\_06\_cat 0  
## ps\_car\_07\_cat 28820  
## ps\_car\_08\_cat 0  
## ps\_car\_09\_cat 1446  
## ps\_car\_10\_cat 0  
## ps\_car\_11\_cat 0  
## ps\_car\_11 6  
## ps\_car\_12 1  
## ps\_car\_13 0  
## ps\_car\_14 106425  
## ps\_car\_15 0  
## ps\_calc\_01 0  
## ps\_calc\_02 0  
## ps\_calc\_03 0  
## ps\_calc\_04 0  
## ps\_calc\_05 0  
## ps\_calc\_06 0  
## ps\_calc\_07 0  
## ps\_calc\_08 0  
## ps\_calc\_09 0  
## ps\_calc\_10 0  
## ps\_calc\_11 0  
## ps\_calc\_12 0  
## ps\_calc\_13 0  
## ps\_calc\_14 0  
## ps\_calc\_15\_bin 0  
## ps\_calc\_16\_bin 0  
## ps\_calc\_17\_bin 0  
## ps\_calc\_18\_bin 0  
## ps\_calc\_19\_bin 0  
## ps\_calc\_20\_bin 0

## [1] "Columns with missing values are:"

## [1] "ps\_ind\_02\_cat" "ps\_ind\_04\_cat" "ps\_ind\_05\_cat" "ps\_reg\_03"   
## [5] "ps\_car\_01\_cat" "ps\_car\_02\_cat" "ps\_car\_03\_cat" "ps\_car\_05\_cat"  
## [9] "ps\_car\_07\_cat" "ps\_car\_09\_cat" "ps\_car\_11" "ps\_car\_12"   
## [13] "ps\_car\_14"

There are a lot of concentration of mising values in few columns. Let’s drop them with a threshold percentage. I am dropping variable with >=5% of missing values in them.

## [1] "Columns with >=5% of missing values are:"

## [1] "ps\_reg\_03" "ps\_car\_03\_cat" "ps\_car\_05\_cat" "ps\_car\_14"

## [1] "Dimensions after droping the variables with >=5% of missing values:"

## [1] 1488028 56

There are missing values to be imputed in these remaining variables. I am imputing NAs with ‘mode’ in categorical/factor variable and with ‘mean’ in numerical variables. Let’s see if any other missing values are there.

## [1] 0

We have converted the data types as required and imputed the missing values. Now, split the combined data back to TRAIN and TEST sets.

## [1] "Dimensions of TRAIN:"

## [1] 595212 55

## [1] "Dimensions of TEST"

## [1] 892816 54

# Feature Engineering

Let’s analyze the target variable in TRAIN.

##   
## 0 1   
## 573518 21694

##   
## 0 1   
## 0.96355248 0.03644752

We can see that the target variable is imbalanced. Class 0 have 0.963% observations & 1 have 0.037% observations. Let’s balance the data set using over/under sampling.

## [1] "Dimensions of data after balancing the target variable"

## [1] 90000 55

## [1] "No of missing values:"

## [1] 0

## [1] "Balance of the target variable:"

##   
## 0 1   
## 44959 45041

##   
## 0 1   
## 0.4995444 0.5004556

Now the target variable is balanced in sampled out data frame with 90000 observations.

We can take whole data set but it requires more computational power. And more over using over and under sampling technique, if we draw a large data set, the algorithm may drop contributing observations. We can use ‘ROSE’ function but it works only on factor and numerical data not on ordinal data.

# Model Building

I am buidling a Random Forest model to predict the target variable. To check the efficiency of the model, I am diving the TRAIN set to train and test sets with a ratio of 7:3(train:test).

## [1] 63000 55

## [1] 27000 55

The ‘train’ set’s dimensions are 63000 X 55 and ‘test’ set’s dimensions are 27000 X 55.

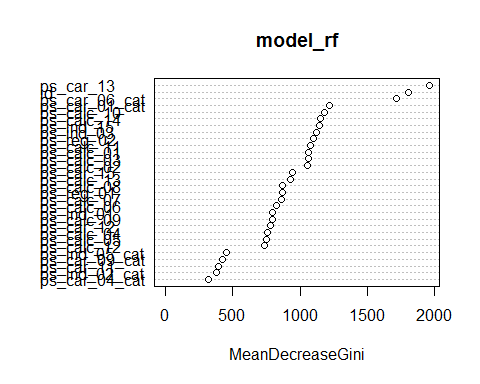
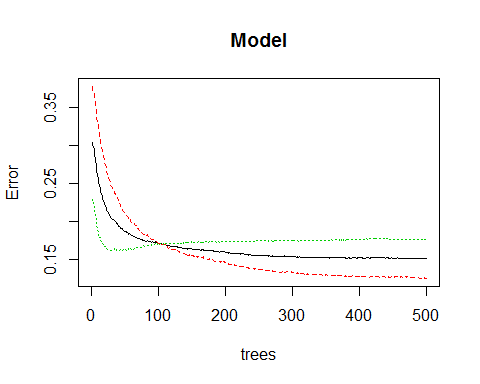
# Random Forest

Random forest can take only maximum 53 levels in a variable. “ps\_car\_11\_cat” has 104 levels in total. So, it has to be removed to build a random forest model.

## [1] 63000 54

## [1] 27000 54

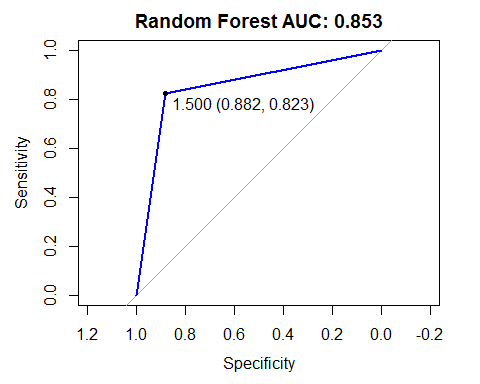
Let’s build the model.



## MeanDecreaseGini Variables  
## ps\_car\_13 1959.401662 ps\_car\_13  
## id 1809.780345 id  
## ps\_car\_06\_cat 1715.060395 ps\_car\_06\_cat  
## ps\_car\_01\_cat 1217.526845 ps\_car\_01\_cat  
## ps\_calc\_10 1178.749389 ps\_calc\_10  
## ps\_calc\_14 1148.456327 ps\_calc\_14  
## ps\_ind\_15 1141.539511 ps\_ind\_15  
## ps\_ind\_03 1119.998441 ps\_ind\_03  
## ps\_reg\_02 1098.079986 ps\_reg\_02  
## ps\_calc\_11 1079.363616 ps\_calc\_11  
## ps\_calc\_01 1062.214147 ps\_calc\_01  
## ps\_calc\_03 1059.361054 ps\_calc\_03  
## ps\_calc\_02 1056.982397 ps\_calc\_02  
## ps\_car\_15 943.227106 ps\_car\_15  
## ps\_calc\_13 930.268941 ps\_calc\_13  
## ps\_calc\_08 872.438665 ps\_calc\_08  
## ps\_reg\_01 869.544035 ps\_reg\_01  
## ps\_calc\_07 864.056858 ps\_calc\_07  
## ps\_calc\_06 822.771903 ps\_calc\_06  
## ps\_ind\_01 798.907850 ps\_ind\_01  
## ps\_calc\_09 797.975401 ps\_calc\_09  
## ps\_car\_12 783.503308 ps\_car\_12  
## ps\_calc\_04 756.423926 ps\_calc\_04  
## ps\_calc\_05 749.027062 ps\_calc\_05  
## ps\_calc\_12 737.597692 ps\_calc\_12  
## ps\_ind\_05\_cat 455.100268 ps\_ind\_05\_cat  
## ps\_car\_09\_cat 422.446973 ps\_car\_09\_cat  
## ps\_car\_11 395.300326 ps\_car\_11  
## ps\_ind\_02\_cat 377.493887 ps\_ind\_02\_cat  
## ps\_car\_04\_cat 320.669210 ps\_car\_04\_cat  
## ps\_calc\_17\_bin 224.731844 ps\_calc\_17\_bin  
## ps\_calc\_16\_bin 219.046439 ps\_calc\_16\_bin  
## ps\_calc\_19\_bin 216.435430 ps\_calc\_19\_bin  
## ps\_calc\_18\_bin 212.104535 ps\_calc\_18\_bin  
## ps\_ind\_04\_cat 207.392311 ps\_ind\_04\_cat  
## ps\_ind\_06\_bin 189.689647 ps\_ind\_06\_bin  
## ps\_ind\_17\_bin 185.102812 ps\_ind\_17\_bin  
## ps\_ind\_16\_bin 173.715714 ps\_ind\_16\_bin  
## ps\_calc\_20\_bin 171.230355 ps\_calc\_20\_bin  
## ps\_ind\_07\_bin 160.916608 ps\_ind\_07\_bin  
## ps\_calc\_15\_bin 160.455851 ps\_calc\_15\_bin  
## ps\_ind\_09\_bin 127.407999 ps\_ind\_09\_bin  
## ps\_car\_02\_cat 125.014593 ps\_car\_02\_cat  
## ps\_ind\_08\_bin 124.106908 ps\_ind\_08\_bin  
## ps\_car\_08\_cat 122.996815 ps\_car\_08\_cat  
## ps\_ind\_18\_bin 117.440132 ps\_ind\_18\_bin  
## ps\_car\_07\_cat 110.008255 ps\_car\_07\_cat  
## ps\_ind\_14 34.603606 ps\_ind\_14  
## ps\_car\_10\_cat 25.340777 ps\_car\_10\_cat  
## ps\_ind\_12\_bin 21.334100 ps\_ind\_12\_bin  
## ps\_ind\_11\_bin 4.507077 ps\_ind\_11\_bin  
## ps\_ind\_13\_bin 2.002674 ps\_ind\_13\_bin  
## ps\_ind\_10\_bin 1.398680 ps\_ind\_10\_bin

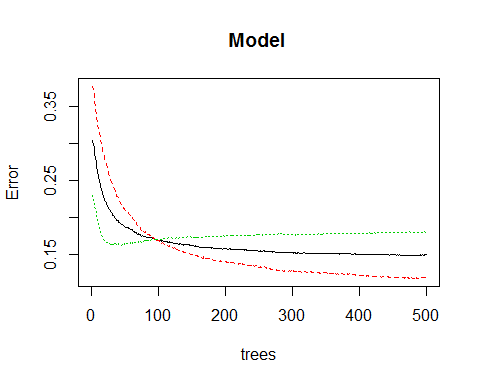
## 0 1   
## 14214 12786

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 11808 2406  
## 1 1574 11212  
##   
## Accuracy : 0.8526   
## 95% CI : (0.8483, 0.8568)  
## No Information Rate : 0.5044   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7053   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8824   
## Specificity : 0.8233   
## Pos Pred Value : 0.8307   
## Neg Pred Value : 0.8769   
## Prevalence : 0.4956   
## Detection Rate : 0.4373   
## Detection Prevalence : 0.5264   
## Balanced Accuracy : 0.8529   
##   
## 'Positive' Class : 0   
##

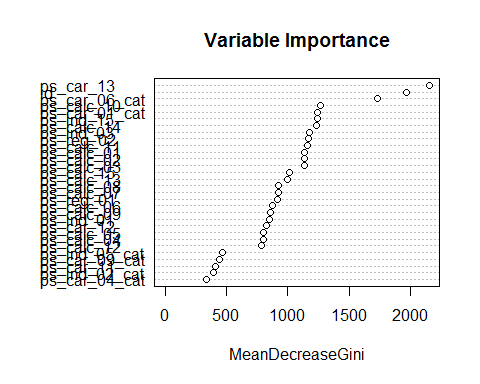
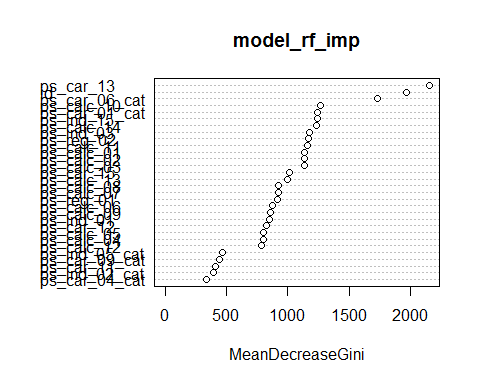


## MeanDecreaseGini  
## id 1809.780345  
## ps\_ind\_01 798.907850  
## ps\_ind\_02\_cat 377.493887  
## ps\_ind\_03 1119.998441  
## ps\_ind\_04\_cat 207.392311  
## ps\_ind\_05\_cat 455.100268  
## ps\_ind\_06\_bin 189.689647  
## ps\_ind\_07\_bin 160.916608  
## ps\_ind\_08\_bin 124.106908  
## ps\_ind\_09\_bin 127.407999  
## ps\_ind\_10\_bin 1.398680  
## ps\_ind\_11\_bin 4.507077  
## ps\_ind\_12\_bin 21.334100  
## ps\_ind\_13\_bin 2.002674  
## ps\_ind\_14 34.603606  
## ps\_ind\_15 1141.539511  
## ps\_ind\_16\_bin 173.715714  
## ps\_ind\_17\_bin 185.102812  
## ps\_ind\_18\_bin 117.440132  
## ps\_reg\_01 869.544035  
## ps\_reg\_02 1098.079986  
## ps\_car\_01\_cat 1217.526845  
## ps\_car\_02\_cat 125.014593  
## ps\_car\_04\_cat 320.669210  
## ps\_car\_06\_cat 1715.060395  
## ps\_car\_07\_cat 110.008255  
## ps\_car\_08\_cat 122.996815  
## ps\_car\_09\_cat 422.446973  
## ps\_car\_10\_cat 25.340777  
## ps\_car\_11 395.300326  
## ps\_car\_12 783.503308  
## ps\_car\_13 1959.401662  
## ps\_car\_15 943.227106  
## ps\_calc\_01 1062.214147  
## ps\_calc\_02 1056.982397  
## ps\_calc\_03 1059.361054  
## ps\_calc\_04 756.423926  
## ps\_calc\_05 749.027062  
## ps\_calc\_06 822.771903  
## ps\_calc\_07 864.056858  
## ps\_calc\_08 872.438665  
## ps\_calc\_09 797.975401  
## ps\_calc\_10 1178.749389  
## ps\_calc\_11 1079.363616  
## ps\_calc\_12 737.597692  
## ps\_calc\_13 930.268941  
## ps\_calc\_14 1148.456327  
## ps\_calc\_15\_bin 160.455851  
## ps\_calc\_16\_bin 219.046439  
## ps\_calc\_17\_bin 224.731844  
## ps\_calc\_18\_bin 212.104535  
## ps\_calc\_19\_bin 216.435430  
## ps\_calc\_20\_bin 171.230355

We got an accuracy of 85% with the above model. Using the variable importance, lets just include only the important variable and see how our model performes



## MeanDecreaseGini  
## id 1965.5798  
## ps\_ind\_01 851.0282  
## ps\_ind\_02\_cat 392.6165  
## ps\_ind\_03 1177.9201  
## ps\_ind\_04\_cat 218.7093  
## ps\_ind\_05\_cat 461.5491  
## ps\_ind\_15 1236.9724  
## ps\_reg\_01 918.3863  
## ps\_reg\_02 1166.6729  
## ps\_car\_01\_cat 1243.5574  
## ps\_car\_04\_cat 336.7746  
## ps\_car\_06\_cat 1730.5016  
## ps\_car\_09\_cat 442.3402  
## ps\_car\_11 406.5555  
## ps\_car\_12 826.0099  
## ps\_car\_13 2152.0441  
## ps\_car\_15 1012.1005  
## ps\_calc\_01 1136.8202  
## ps\_calc\_02 1134.2293  
## ps\_calc\_03 1133.2601  
## ps\_calc\_04 800.2136  
## ps\_calc\_05 800.3744  
## ps\_calc\_06 874.6576  
## ps\_calc\_07 924.6854  
## ps\_calc\_08 925.8367  
## ps\_calc\_09 855.4589  
## ps\_calc\_10 1263.3665  
## ps\_calc\_11 1160.3448  
## ps\_calc\_12 785.1832  
## ps\_calc\_13 998.3513  
## ps\_calc\_14 1233.6412  
## ps\_calc\_16\_bin 233.8106  
## ps\_calc\_17\_bin 238.6797  
## ps\_calc\_18\_bin 224.2464  
## ps\_calc\_19\_bin 229.2240



## 0 1   
## 14392 12608

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 11929 2463  
## 1 1453 11155  
##   
## Accuracy : 0.855   
## 95% CI : (0.8507, 0.8591)  
## No Information Rate : 0.5044   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7101   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.8914   
## Specificity : 0.8191   
## Pos Pred Value : 0.8289   
## Neg Pred Value : 0.8848   
## Prevalence : 0.4956   
## Detection Rate : 0.4418   
## Detection Prevalence : 0.5330   
## Balanced Accuracy : 0.8553   
##   
## 'Positive' Class : 0   
##

