Porto Seguro's Safe Driver Prediction

Manish Reddy Jannepally

December 12, 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.

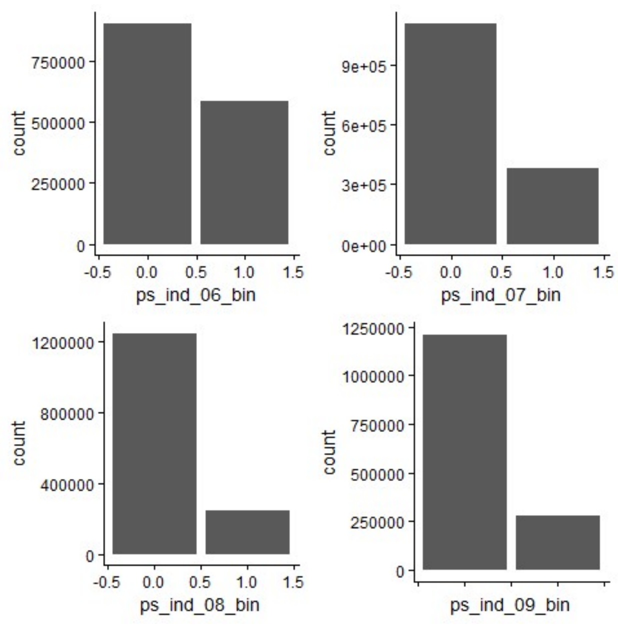
Loading Required Libraries

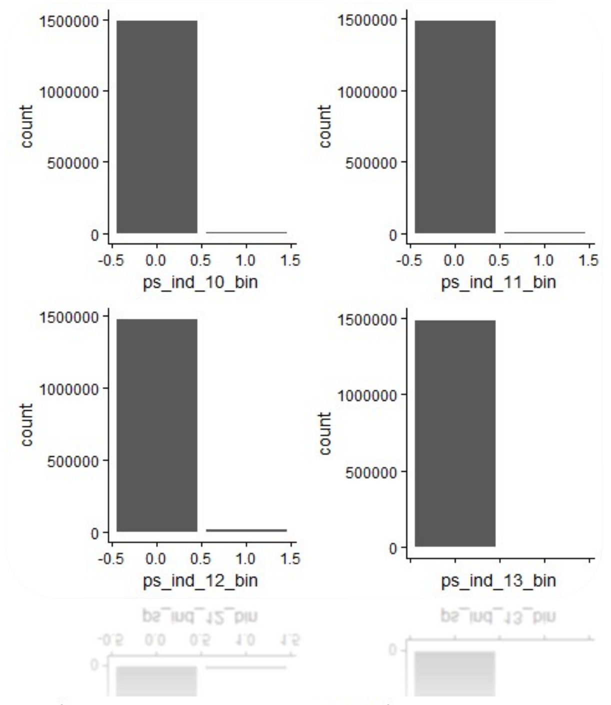
The following libraries are used in this project…

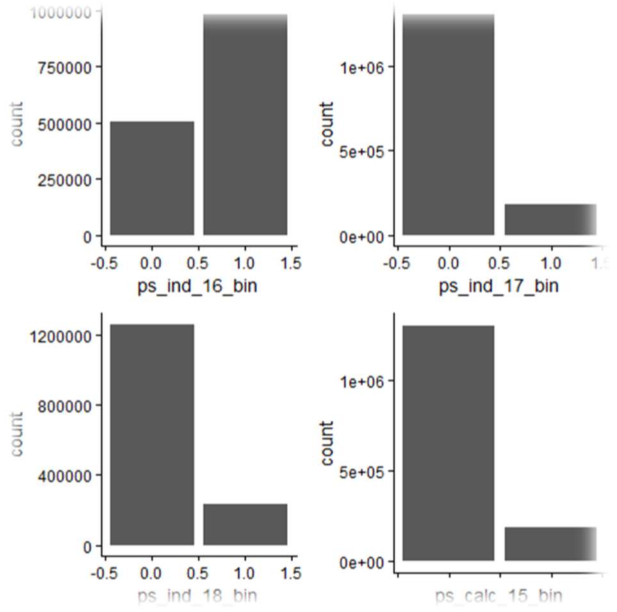
```r
library(dplyr) #data manipulation
library(readr) #input/output
library(tibble)#data wrangling
library(data.table) #data manipulation
library(forcats) #factor manipulation
library(stringr) #string manipulation
library(caret) #training and evaluation model
library(randomForest) #Random Forest model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

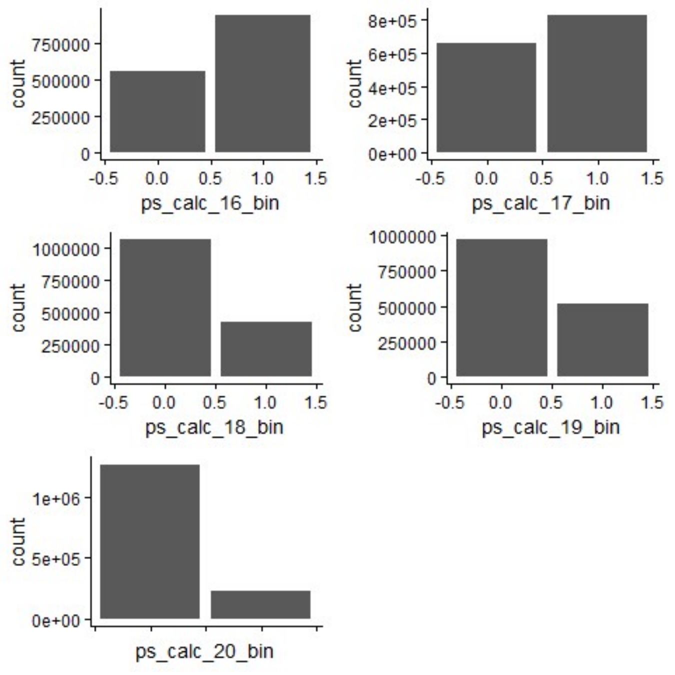
Exploratory Data Analysis

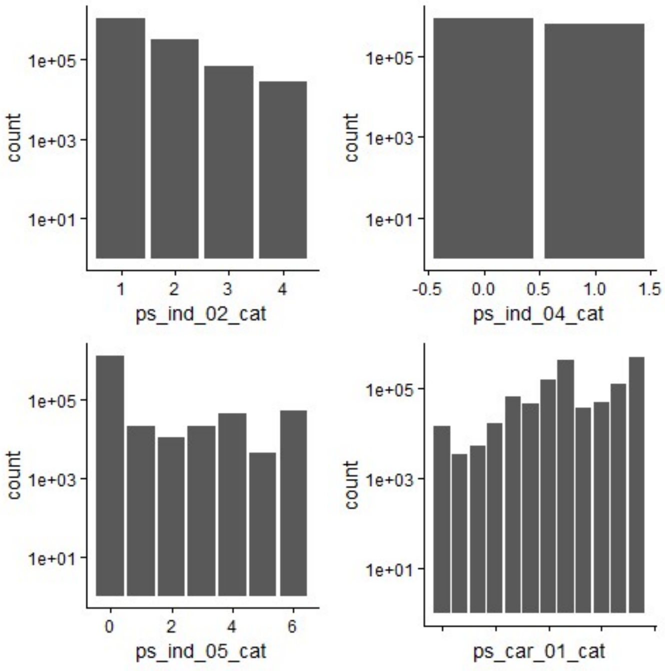
I have done an individual variable analysis using plots to ensure my assumptions regarding the variables' data type is right. Below are the plots of my EDA.

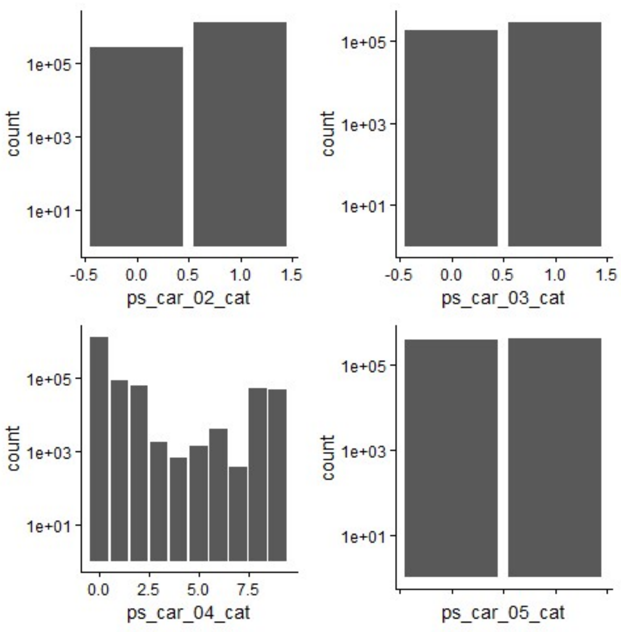


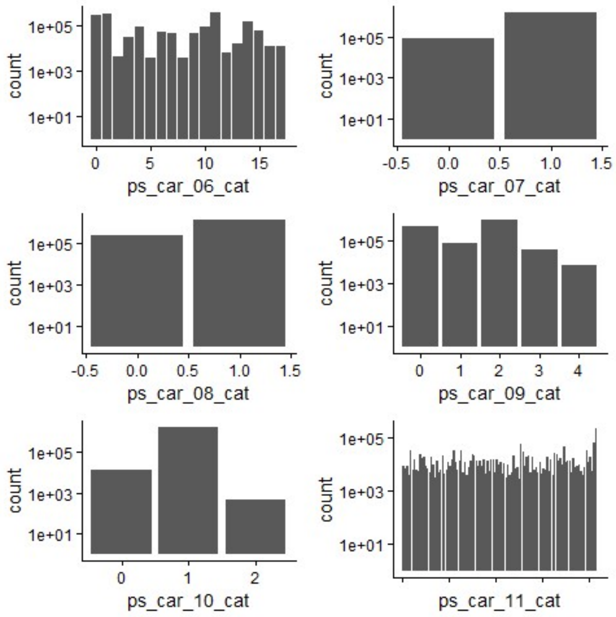


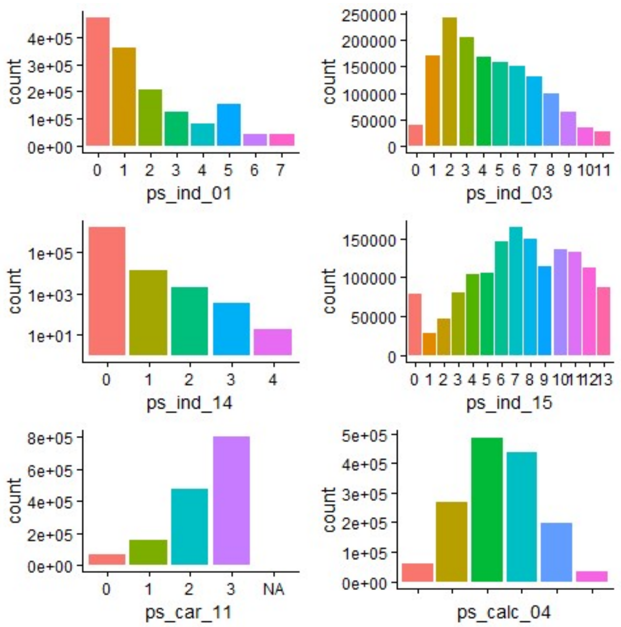


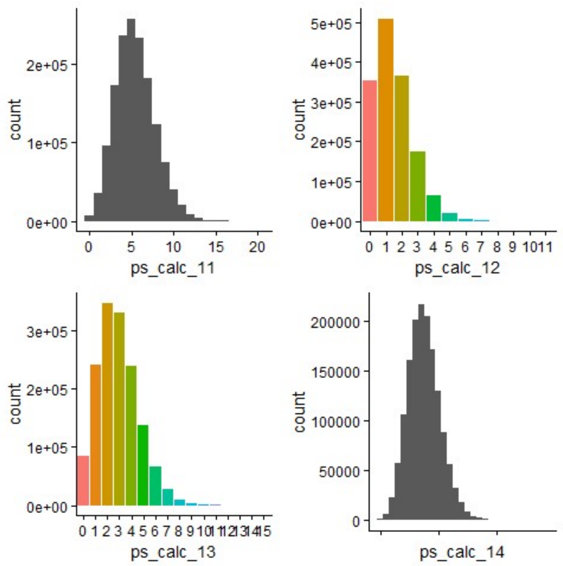


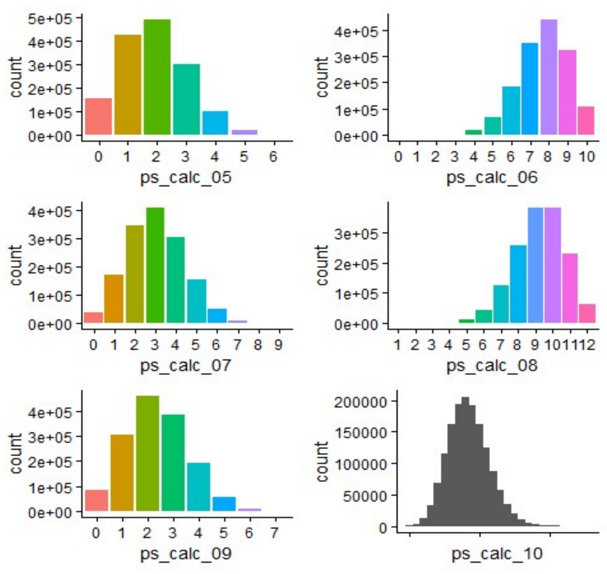

count
1e+05
1e+03
1e+01
1 2 3 4
ps_ind_02_cat
-0.5 0.0 0.5 1.0 1.5
ps_ind_04_cat
0 2 4 6
ps_ind_05_cat
ps_car_01_cat

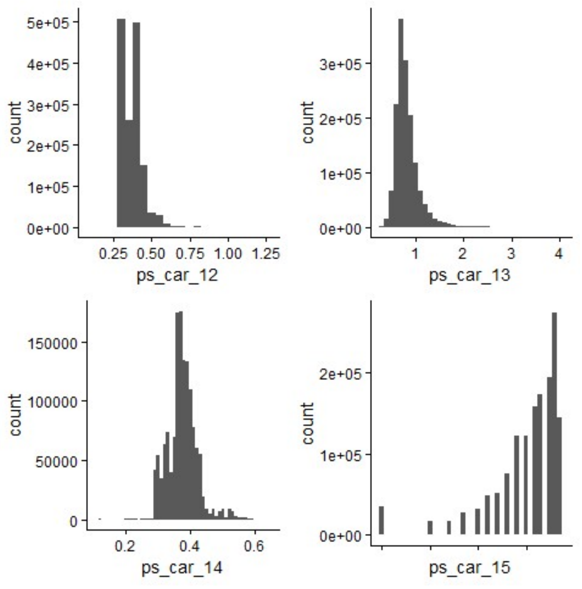

count
1e+05
1e+03
1e+01
-0.5 0.0 0.5 1.0 1.5
ps_car_02_cat
ps_car_03_cat
0.0 2.5 5.0 7.5
ps_car_04_cat
ps_car_05_cat

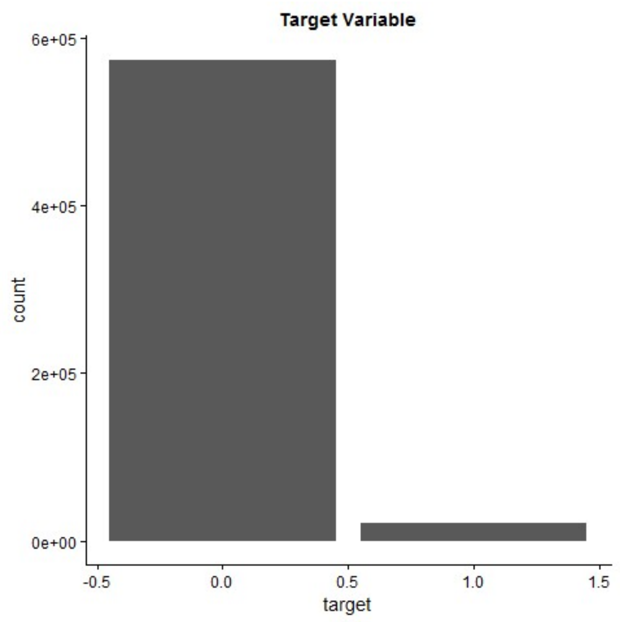

count
1e+05
1e+03
1e+01
0 5 10 15
ps_car_06_cat
-0.5 0.0 0.5 1.0 1.5
ps_car_07_cat
ps_car_08_cat
0 1 2 3 4
ps_car_09_cat
0 1 2
ps_car_10_cat
ps_car_11_cat

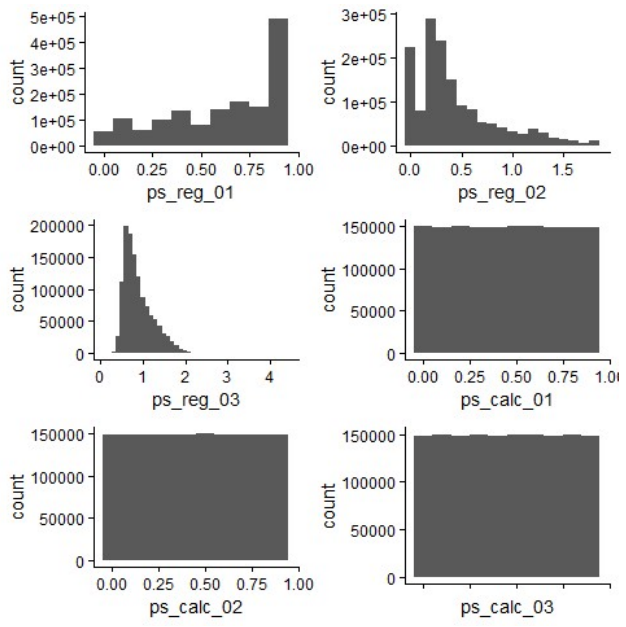

count
4e+05
3e+05
2e+05
1e+05
0e+00
0 1 2 3 4 5 6 7
ps_ind_01
250000
200000
150000
100000
50000
0
0 1 2 3 4 5 6 7 8 9 10 11
ps_ind_03
1e+05
1e+03
1e+01
0 1 2 3 4
ps_ind_14
150000
100000
50000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13
ps_ind_15
8e+05
6e+05
4e+05
2e+05
0e+00
0 1 2 3 NA
ps_car_11
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
ps_calc_04


count
2e+05
1e+05
0e+00
0 5 10 15 20
ps_calc_11
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
0 1 2 3 4 5 6 7 8 9 10 11
ps_calc_12
3e+05
2e+05
1e+05
0e+00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
ps_calc_13
200000
150000
100000
50000
0
ps_calc_14









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] 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 #missing values in train data

## [1] 1270295 #missing values in test data
```

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.

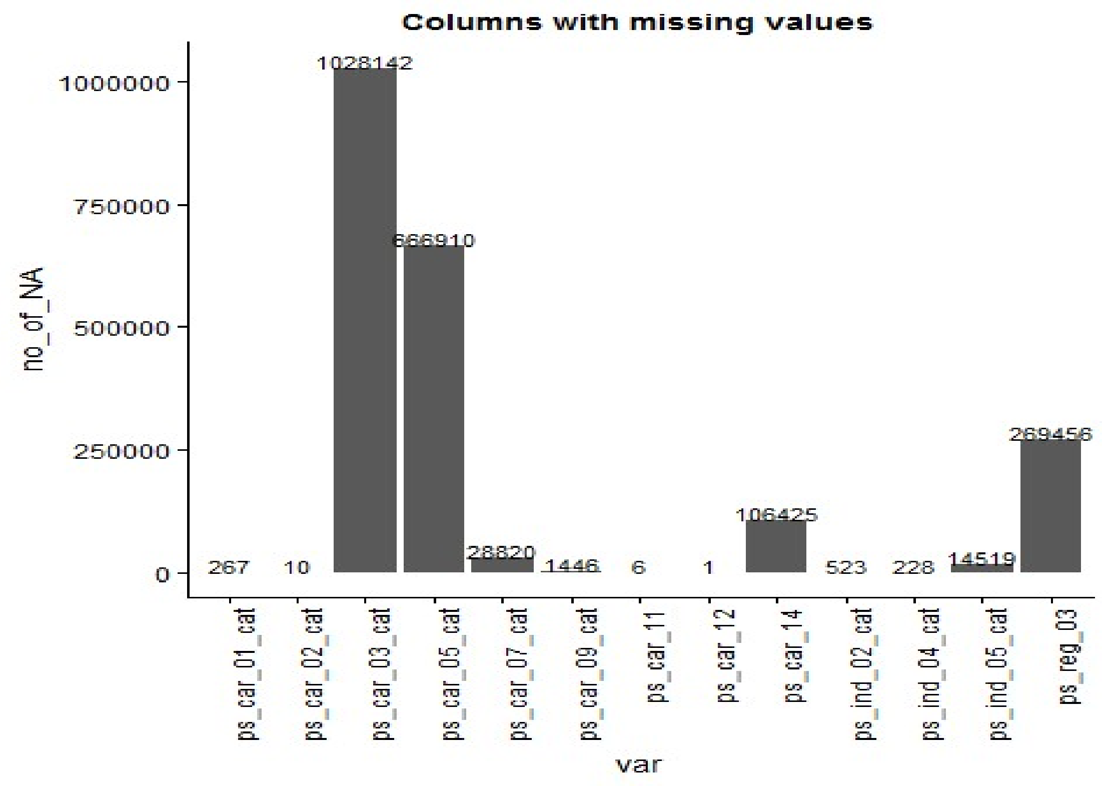
```
var_groups = fread("var_groups.csv")
names = intersect(colnames(combined_data), var_groups[["names"]])
for(var_name in names){

        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.

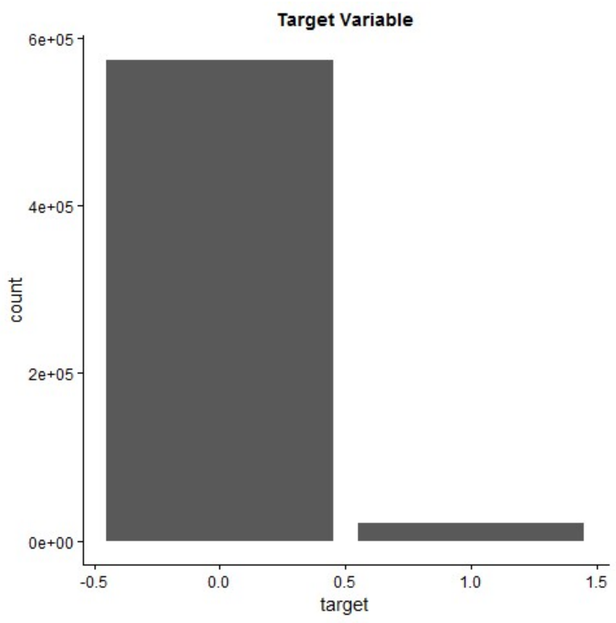
```
## [1] 0
```

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

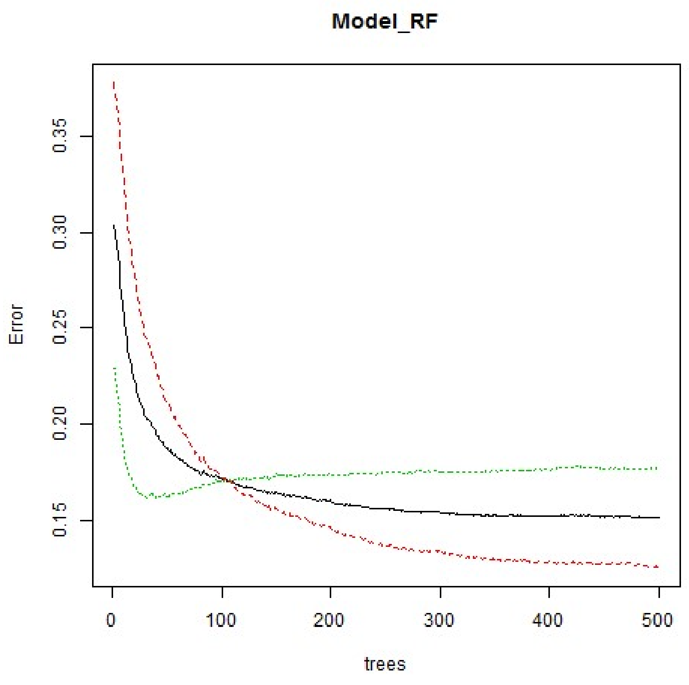
Random forest can take only maximum 53 levels in a variable. "ps_car_11_cat" has 104 levels in total. So, it has to be removed to build a random forest model.

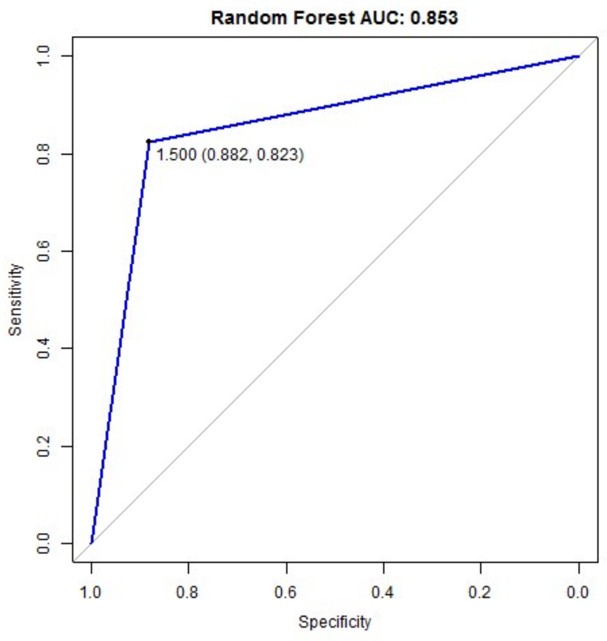
```
## [1] 63000    54

## [1] 27000    54
```

Random Forest is giving the below results.

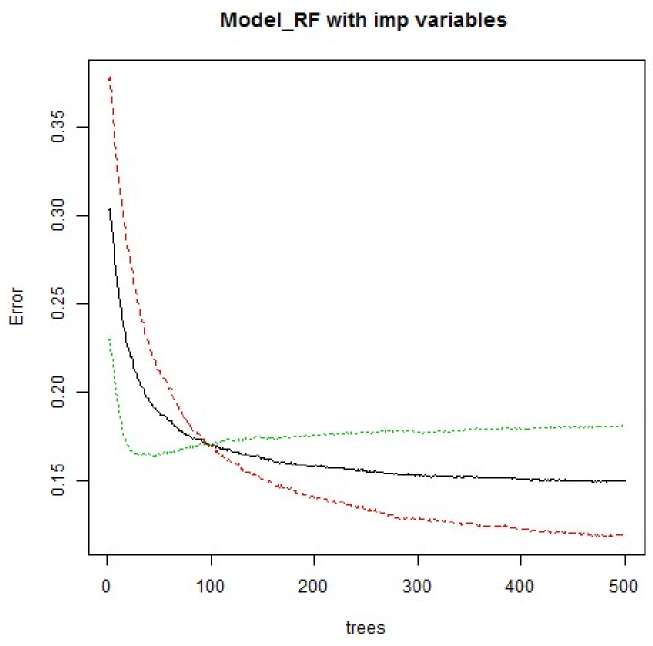
```
##     0     1   #Summary of Predicted target variable
## 14214 12786

## 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
## 
```

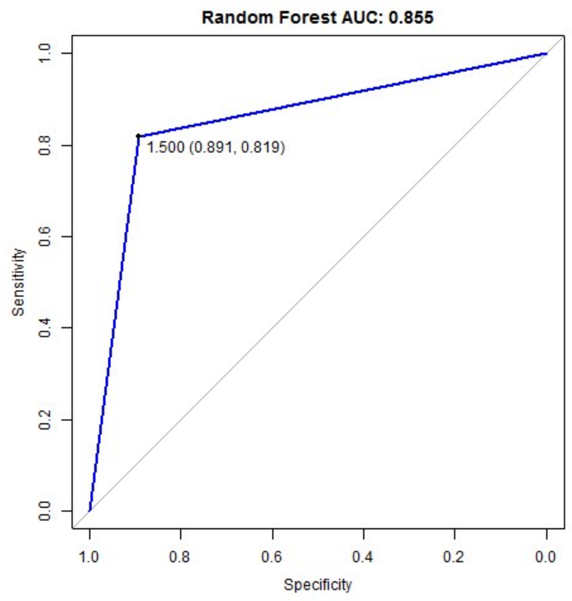
We got an accuracy of 85.3% with the above model. Using the variable importance, I included only the important variable and build models. But the accuracy seems to be





fluctuvating just 0.2 - 0.3%.





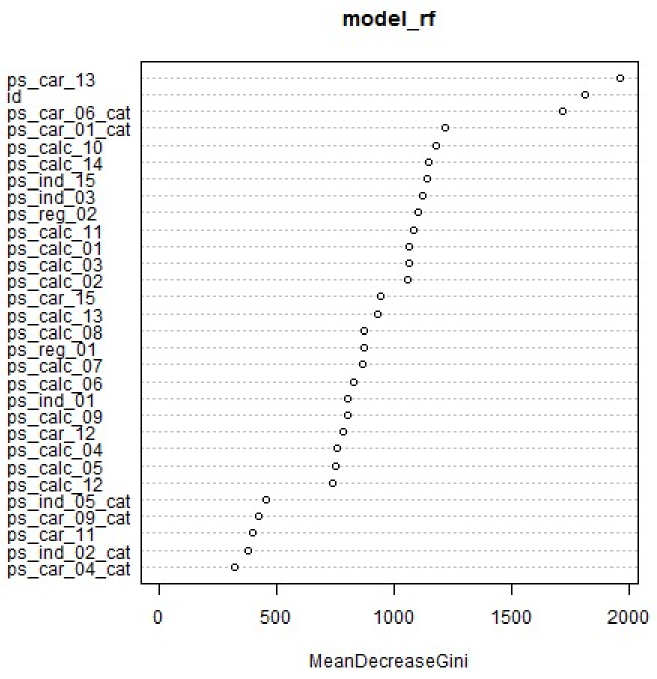
Let’s see the variable importance of model_rf.

Including only the most important variables in the model building, below are the improvements.

Results of tuned Random Forests are below,

```
##     0     1  #summary of predicted target variable
## 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
##
```

As experiments, I have build model with Naive Bayes, Gradient Boosting algorithms. Naive Bayes gave an accuracy of 58% where as the GBM gave 59% of accuracy. Random Forests gave us 85% percent of accuracy. So, Random Forest is our base model.

Now, I will find the normalized gini coefficient of the TEST data.

The Gini Coefficient ranges from approximately 0 for random guessing, to approximately 0.5 for a perfect score. The theoretical maximum for the discrete calculation is (1 - frac_pos) / 2.

The head of the final submission file is as shown below.

id	target
0	0.336
1	0.372
2	0.35
3	0.228
4	0.39
5	0.412
6	0.244

The results for the test data given is submitted as Final_submission.csv file.

THANK YOU…