**Contents:**

1. **Exploratory Data Analysis**

**\* Loading dataset and libraries**

**\* Data cleaning**

**\* Typecasting the attributes**

**\* Target classes count**

**\* Missing value analysis**

**2. Attributes Distributions and trends**

**\* Distribution of train attributes**

**\* Distribution of test attributes**

**\* Mean distribution of attributes**

**\* Standard deviation distribution of attributes**

**\* Skewness distribution of attributes**

**\* Kurtosis distribution of attributes**

**\* Outliers analysis**

**3. Correlation matrix**

**4. Split the dataset into train and test dataset**

**5. Modelling the training dataset**

**\* Logistic Regression Model**

**\* ROSE Model**

**\* LightGBM Model**

**6. Cross Validation Prediction**

**\* Logistic Regression CV Prediction**

**\* ROSE CV Prediction**

**\* LightGBM CV Prediction**

**7. Model performance on test dataset**

**\* Logistic Regression Prediction**

**\* ROSE Prediction**

**\* LightGBM Prediction**

**8. Model Evaluation Metrics**

**\* Confusion Matrix**

**\* ROC\_AUC score**

**9. Choosing best model for predicting customer transaction**

* Loading all the required libraries and cleaning up the work space.

rm(list = ls())

setwd("C:/Users/divij/Documents")

getwd()

library(tidyverse)

library(ggplot2)

library(moments)

library(DataExplorer)

library(caret)

library(Matrix)

library(pdp)

library(mlbench)

library(gridExtra)

library(caTools)

library(randomForest)

library(glmnet)

library(mlr)

library(vita)

library(rBayesianOptimization)

library(lightgbm)

library(pROC)

library(DMwR)

library(ROSE)

library(yardstick)

* Importing the train and test data

train\_df<- read.csv("train.csv", header = T, na.strings = c(" ", "", "NA"))

test\_df = read.csv("test.csv")

> head(train\_df)

ID\_code target var\_0 var\_1 var\_2 var\_3 var\_4 var\_5 var\_6 var\_7 var\_8 var\_9

1 train\_0 1 8.9255 -6.7863 11.9081 5.0930 11.4607 -9.2834 5.1187 18.6266 -4.9200 5.7470

2 train\_1 1 11.5006 -4.1473 13.8588 5.3890 12.3622 7.0433 5.6208 16.5338 3.1468 8.0851

3 train\_2 1 8.6093 -2.7457 12.0805 7.8928 10.5825 -9.0837 6.9427 14.6155 -4.9193 5.9525

4 train\_3 1 11.0604 -2.1518 8.9522 7.1957 12.5846 -1.8361 5.8428 14.9250 -5.8609 8.2450

var\_10 var\_11 var\_12 var\_13 var\_14 var\_15 var\_16 var\_17 var\_18 var\_19 var\_20

1 2.9252 3.1821 14.0137 0.5745 8.7989 14.5691 5.7487 -7.2393 4.2840 30.7133 10.5350

2 -0.4032 8.0585 14.0239 8.4135 5.4345 13.7003 13.8275 -15.5849 7.8000 28.5708 3.4287

3 -0.3249 -11.2648 14.1929 7.3124 7.5244 14.6472 7.6782 -1.7395 4.7011 20.4775 17.7559

4 2.3061 2.8102 13.8463 11.9704 6.4569 14.8372 10.7430 -0.4299 15.9426 13.7257 20.3010

var\_21 var\_22 var\_23 var\_24 var\_25 var\_26 var\_27 var\_28 var\_29 var\_30 var\_31 var\_32

1 16.2191 2.5791 2.4716 14.3831 13.4325 -5.1488 -0.4073 4.9306 5.9965 -0.3085 12.9041 -3.8766

2 2.7407 8.5524 3.3716 6.9779 13.8910 -11.7684 -2.5586 5.0464 0.5481 -9.2987 7.8755 1.2859

3 18.1377 1.2145 3.5137 5.6777 13.2177 -7.9940 -2.9029 5.8463 6.1439 -11.1025 12.4858 -2.2871

4 12.5579 6.8202 2.7229 12.1354 13.7367 0.8135 -0.9059 5.9070 2.8407 -15.2398 10.4407 -2.5731

var\_33 var\_34 var\_35 var\_36 var\_37 var\_38 var\_39 var\_40 var\_41 var\_42 var\_43 var\_44

1 16.8911 11.1920 10.5785 0.6764 7.8871 4.6667 3.8743 -5.2387 7.3746 11.5767 12.0446 11.6418

2 19.3710 11.3702 0.7399 2.7995 5.8434 10.8160 3.6783 -11.1147 1.8730 9.8775 11.7842 1.2444

3 19.0422 11.0449 4.1087 4.6974 6.9346 10.8917 0.9003 -13.5174 2.2439 11.5283 12.0406 4.1006

4 6.1796 10.6093 -5.9158 8.1723 2.8521 9.1738 0.6665 -3.8294 -1.0370 11.7770 11.2834 8.0485

var\_45 var\_46 var\_47 var\_48 var\_49 var\_50 var\_51 var\_52 var\_53 var\_54 var\_55

1 -7.0170 5.9226 -14.2136 16.0283 5.3253 12.9194 29.0460 -0.6940 5.1736 -0.7474 14.8322

2 -47.3797 7.3718 0.1948 34.4014 25.7037 11.8343 13.2256 -4.1083 6.6885 -8.0946 18.5995

3 -7.9078 11.1405 -5.7864 20.7477 6.8874 12.9143 19.5856 0.7268 6.4059 9.3124 6.2846

4 -24.6840 12.7404 -35.1659 0.7613 8.3838 12.6832 9.5503 1.7895 5.2091 8.0913 12.3972

var\_56 var\_57 var\_58 var\_59 var\_60 var\_61 var\_62 var\_63 var\_64 var\_65 var\_66 var\_67

1 11.2668 5.3822 2.0183 10.1166 16.1828 4.9590 2.0771 -0.2154 8.6748 9.5319 5.8056 22.4321

2 19.3219 7.0118 1.9210 8.8682 8.0109 -7.2417 1.7944 -1.3147 8.1042 1.5365 5.4007 7.9344

3 15.6372 5.8200 1.1000 9.1854 12.5963 -10.3734 0.8748 5.8042 3.7163 -1.1016 7.3667 9.8565

4 14.4698 6.5850 3.3164 9.4638 15.7820 -25.0222 3.4418 -4.3923 8.6464 6.3072 5.6221 23.6143

var\_68 var\_69 var\_70 var\_71 var\_72 var\_73 var\_74 var\_75 var\_76 var\_77 var\_78 var\_79

1 5.0109 -4.7010 21.6374 0.5663 5.1999 8.8600 43.1127 18.3816 -2.3440 23.4104 6.5199 12.1983

2 5.0220 2.2302 40.5632 0.5134 3.1701 20.1068 7.7841 7.0529 3.2709 23.4822 5.5075 13.7814

3 5.0228 -5.7828 2.3612 0.8520 6.3577 12.1719 19.7312 19.4465 4.5048 23.2378 6.3191 12.8046

4 5.0220 -3.9989 4.0462 0.2500 1.2516 24.4187 4.5290 15.4235 11.6875 23.6273 4.0806 15.2733

var\_80 var\_81 var\_82 var\_83 var\_84 var\_85 var\_86 var\_87 var\_88 var\_89 var\_90

1 13.6468 13.8372 1.3675 2.9423 -4.5213 21.4669 9.3225 16.4597 7.9984 -1.7069 -21.4494

2 2.5462 18.1782 0.3683 -4.8210 -5.4850 13.7867 -13.5901 11.0993 7.9022 12.2301 0.4768

3 7.4729 15.7811 13.3529 10.1852 5.4604 19.0773 -4.4577 9.5413 11.9052 2.1447 -22.4038

4 0.7839 10.5404 1.6212 -5.2896 1.6027 17.9762 -2.3174 15.6298 4.5474 7.5509 -7.5866

var\_91 var\_92 var\_93 var\_94 var\_95 var\_96 var\_97 var\_98 var\_99 var\_100 var\_101 var\_102

1 6.7806 11.0924 9.9913 14.8421 0.1812 8.9642 16.2572 2.1743 -3.4132 9.4763 13.3102 26.5376

2 6.8852 8.0905 10.9631 11.7569 -1.2722 24.7876 26.6881 1.8944 0.6939 -13.6950 8.4068 35.4734

3 7.0883 14.1613 10.5080 14.2621 0.2647 20.4031 17.0360 1.6981 -0.0269 -0.3939 12.6317 14.8863

4 7.0364 14.4027 10.7795 7.2887 -1.0930 11.3596 18.1486 2.8344 1.9480 -19.8592 22.5316 18.6129

var\_103 var\_104 var\_105 var\_106 var\_107 var\_108 var\_109 var\_110 var\_111 var\_112 var\_113

1 1.4403 14.7100 6.0454 9.5426 17.1554 14.1104 24.3627 2.0323 6.7602 3.9141 -0.4851

2 1.7093 15.1866 2.6227 7.3412 32.0888 13.9550 13.0858 6.6203 7.1051 5.3523 8.5426

3 1.3854 15.0284 3.9995 5.3683 8.6273 14.1963 20.3882 3.2304 5.7033 4.5255 2.1929

4 1.3512 9.3291 4.2835 10.3907 7.0874 14.3256 14.4135 4.2827 6.9750 1.6480 11.6896

var\_114 var\_115 var\_116 var\_117 var\_118 var\_119 var\_120 var\_121 var\_122 var\_123 var\_124

1 2.5240 1.5093 2.5516 15.5752 -13.4221 7.2739 16.0094 9.7268 0.8897 0.7754 4.2218

2 3.6159 4.1569 3.0454 7.8522 -11.5100 7.5109 31.5899 9.5018 8.2736 10.1633 0.1225

3 3.1290 2.9044 1.1696 28.7632 -17.2738 2.1056 21.1613 8.9573 2.7768 -2.1746 3.6932

4 2.5762 -2.5459 5.3446 38.1015 3.5732 5.0988 30.5644 11.3025 3.9618 -8.2464 2.7038

var\_125 var\_126 var\_127 var\_128 var\_129 var\_130 var\_131 var\_132 var\_133 var\_134 var\_135

1 12.0039 13.8571 -0.7338 -1.9245 15.4462 12.8287 0.3587 9.6508 6.5674 5.1726 3.1345

2 12.5942 14.5697 2.4354 0.8194 16.5346 12.4205 -0.1780 5.7582 7.0513 1.9568 -8.9921

3 12.4653 14.1978 -2.5511 -0.9479 17.1092 11.5419 0.0975 8.8186 6.6231 3.9358 -11.7218

4 12.3441 12.5431 -1.3683 3.5974 13.9761 14.3003 1.0486 8.9500 7.1954 -1.1984 1.9586

var\_136 var\_137 var\_138 var\_139 var\_140 var\_141 var\_142 var\_143 var\_144 var\_145 var\_146

1 29.4547 31.4045 2.8279 15.6599 8.3307 -5.6011 19.0614 11.2663 8.6989 8.3694 11.5659

2 9.7797 18.1577 -1.9721 16.1622 3.6937 6.6803 -0.3243 12.2806 8.6086 11.0738 8.9231

3 24.5437 15.5827 3.8212 8.6674 7.3834 -2.4438 10.2158 7.4844 9.1104 4.3649 11.4934

4 27.5609 24.6065 -2.8233 8.9821 3.8873 15.9638 10.0142 7.8388 9.9718 2.9253 10.4994

var\_147 var\_148 var\_149 var\_150 var\_151 var\_152 var\_153 var\_154 var\_155 var\_156 var\_157

1 -16.4727 4.0288 17.9244 18.5177 10.7800 9.0056 16.6964 10.4838 1.6573 12.1749 -13.1324

2 11.7700 4.2578 -4.4223 20.6294 14.8743 9.4317 16.7242 -0.5687 0.1898 12.2419 -9.6953

3 1.7624 4.0714 -1.2681 14.3330 8.0088 4.4015 14.1479 -5.1747 0.5778 14.5362 -1.7624

4 4.1622 3.7613 2.3701 18.0984 17.1765 7.6508 18.2452 17.0336 -10.9370 12.0500 -1.2155

var\_158 var\_159 var\_160 var\_161 var\_162 var\_163 var\_164 var\_165 var\_166 var\_167 var\_168

1 17.6054 11.5423 15.4576 5.3133 3.6159 5.0384 6.6760 12.6644 2.7004 -0.6975 9.5981

2 22.3949 10.6261 29.4846 5.8683 3.8208 15.8348 -5.0121 15.1345 3.2003 9.3192 3.8821

3 33.8820 11.6041 13.2070 5.8442 4.7086 5.7141 -1.0410 20.5092 3.2790 -5.5952 7.3176

4 19.9750 12.3892 31.8833 5.9684 7.2084 3.8899 -11.0882 17.2502 2.5881 -2.7018 0.5641

var\_169 var\_170 var\_171 var\_172 var\_173 var\_174 var\_175 var\_176 var\_177 var\_178 var\_179

1 5.4879 -4.7645 -8.4254 20.8773 3.1531 18.5618 7.7423 -10.1245 13.7241 -3.5189 1.7202

2 5.7999 5.5378 5.0988 22.0330 5.5134 30.2645 10.4968 -7.2352 16.5721 -7.3477 11.0752

3 5.7690 -7.0927 -3.9116 7.2569 -5.8234 25.6820 10.9202 -0.3104 8.8438 -9.7009 2.4013

4 5.3430 -7.1541 -6.1920 18.2366 11.7134 14.7483 8.1013 11.8771 13.9552 -10.4701 5.6961

var\_180 var\_181 var\_182 var\_183 var\_184 var\_185 var\_186 var\_187 var\_188 var\_189 var\_190

1 -8.4051 9.0164 3.0657 14.3691 25.8398 5.8764 11.8411 -19.7159 17.5743 0.5857 4.4354

2 -5.5937 9.4878 -14.9100 9.4245 22.5441 -4.8622 7.6543 -15.9319 13.3175 -0.3566 7.6421

3 -4.2935 9.3908 -13.2648 3.1545 23.0866 -5.3000 5.3745 -6.2660 10.1934 -0.8417 2.9057

4 -3.7546 8.4117 1.8986 7.2601 -0.4639 -0.0498 7.9336 -12.8279 12.4124 1.8489 4.4666

var\_191 var\_192 var\_193 var\_194 var\_195 var\_196 var\_197 var\_198 var\_199

1 3.9642 3.1364 1.6910 18.5227 -2.3978 7.8784 8.5635 12.7803 -1.0914

2 7.7214 2.5837 10.9516 15.4305 2.0339 8.1267 8.7889 18.3560 1.9518

3 9.7905 1.6704 1.6858 21.6042 3.1417 -6.5213 8.2675 14.7222 0.3965

4 4.7433 0.7178 1.4214 23.0347 -1.2706 -2.9275 10.2922 17.9697 -8.9996

[ reached getOption("max.print") -- omitted 2 rows ]

* dimension of train data

dim(train\_df)

[1] 200000 202

* Summary of the train dataset

> str(train\_df)

'data.frame': 200000 obs. of 202 variables:

$ ID\_code: Factor w/ 200000 levels "train\_0","train\_1",..: 1 2 111113 122224 133335 144446 155557 166668 177779 188890 ...

$ target : num 1 1 1 1 1 1 1 1 1 1 ...

$ var\_0 : num 8.93 11.5 8.61 11.06 9.84 ...

$ var\_1 : num -6.79 -4.15 -2.75 -2.15 -1.48 ...

$ var\_2 : num 11.91 13.86 12.08 8.95 12.87 ...

$ var\_3 : num 5.09 5.39 7.89 7.2 6.64 ...

$ var\_4 : num 11.5 12.4 10.6 12.6 12.3 ...

$ var\_5 : num -9.28 7.04 -9.08 -1.84 2.45 ...

$ var\_6 : num 5.12 5.62 6.94 5.84 5.94 ...

$ var\_7 : num 18.6 16.5 14.6 14.9 19.3 ...

$ var\_8 : num -4.92 3.15 -4.92 -5.86 6.27 ...

$ var\_9 : num 5.75 8.09 5.95 8.24 7.68 ...

$ var\_10 : num 2.925 -0.403 -0.325 2.306 -9.446 ...

$ var\_11 : num 3.18 8.06 -11.26 2.81 -12.14 ...

$ var\_12 : num 14 14 14.2 13.8 13.8 ...

$ var\_13 : num 0.575 8.414 7.312 11.97 7.889 ...

$ var\_14 : num 8.8 5.43 7.52 6.46 7.79 ...

$ var\_15 : num 14.6 13.7 14.6 14.8 15.1 ...

$ var\_16 : num 5.75 13.83 7.68 10.74 8.49 ...

$ var\_17 : num -7.24 -15.58 -1.74 -0.43 -3.07 ...

$ var\_18 : num 4.28 7.8 4.7 15.94 6.53 ...

$ var\_19 : num 30.7 28.6 20.5 13.7 11.3 ...

$ var\_20 : num 10.54 3.43 17.76 20.3 21.42 ...

$ var\_21 : num 16.22 2.74 18.14 12.56 18.96 ...

$ var\_22 : num 2.58 8.55 1.21 6.82 10.11 ...

$ var\_23 : num 2.47 3.37 3.51 2.72 2.71 ...

$ var\_24 : num 14.38 6.98 5.68 12.14 14.21 ...

$ var\_25 : num 13.4 13.9 13.2 13.7 13.5 ...

$ var\_26 : num -5.149 -11.768 -7.994 0.814 3.174 ...

$ var\_27 : num -0.407 -2.559 -2.903 -0.906 -3.342 ...

$ var\_28 : num 4.93 5.05 5.85 5.91 5.9 ...

$ var\_29 : num 5.997 0.548 6.144 2.841 7.935 ...

$ var\_30 : num -0.308 -9.299 -11.102 -15.24 -3.158 ...

$ var\_31 : num 12.9 7.88 12.49 10.44 9.47 ...

$ var\_32 : num -3.8766 1.2859 -2.2871 -2.5731 -0.0083 ...

$ var\_33 : num 16.89 19.37 19.04 6.18 19.32 ...

$ var\_34 : num 11.2 11.4 11 10.6 12.4 ...

$ var\_35 : num 10.579 0.74 4.109 -5.916 0.633 ...

$ var\_36 : num 0.676 2.8 4.697 8.172 2.792 ...

$ var\_37 : num 7.89 5.84 6.93 2.85 5.82 ...

$ var\_38 : num 4.67 10.82 10.89 9.17 19.3 ...

$ var\_39 : num 3.874 3.678 0.9 0.666 1.445 ...

$ var\_40 : num -5.24 -11.11 -13.52 -3.83 -5.6 ...

$ var\_41 : num 7.37 1.87 2.24 -1.04 14.07 ...

$ var\_42 : num 11.58 9.88 11.53 11.78 11.92 ...

$ var\_43 : num 12 11.8 12 11.3 11.5 ...

$ var\_44 : num 11.64 1.24 4.1 8.05 6.91 ...

$ var\_45 : num -7.02 -47.38 -7.91 -24.68 -65.49 ...

$ var\_46 : num 5.92 7.37 11.14 12.74 13.87 ...

$ var\_47 : num -14.2136 0.1948 -5.7864 -35.1659 0.0444 ...

$ var\_48 : num 16.028 34.401 20.748 0.761 -0.135 ...

$ var\_49 : num 5.33 25.7 6.89 8.38 14.43 ...

$ var\_50 : num 12.9 11.8 12.9 12.7 13.3 ...

$ var\_51 : num 29.05 13.23 19.59 9.55 10.49 ...

$ var\_52 : num -0.694 -4.108 0.727 1.79 -1.437 ...

$ var\_53 : num 5.17 6.69 6.41 5.21 5.76 ...

$ var\_54 : num -0.747 -8.095 9.312 8.091 -8.541 ...

$ var\_55 : num 14.83 18.6 6.28 12.4 14.15 ...

$ var\_56 : num 11.3 19.3 15.6 14.5 17 ...

$ var\_57 : num 5.38 7.01 5.82 6.58 6.18 ...

$ var\_58 : num 2.02 1.92 1.1 3.32 1.95 ...

$ var\_59 : num 10.12 8.87 9.19 9.46 9.2 ...

$ var\_60 : num 16.18 8.01 12.6 15.78 8.66 ...

$ var\_61 : num 4.96 -7.24 -10.37 -25.02 -27.74 ...

$ var\_62 : num 2.077 1.794 0.875 3.442 -0.495 ...

$ var\_63 : num -0.215 -1.315 5.804 -4.392 -1.784 ...

$ var\_64 : num 8.67 8.1 3.72 8.65 5.27 ...

$ var\_65 : num 9.53 1.54 -1.1 6.31 -4.32 ...

$ var\_66 : num 5.81 5.4 7.37 5.62 6.99 ...

$ var\_67 : num 22.43 7.93 9.86 23.61 1.62 ...

$ var\_68 : num 5.01 5.02 5.02 5.02 5.03 ...

$ var\_69 : num -4.7 2.23 -5.78 -4 -3.24 ...

$ var\_70 : num 21.64 40.56 2.36 4.05 40.12 ...

$ var\_71 : num 0.566 0.513 0.852 0.25 0.774 ...

$ var\_72 : num 5.2 3.17 6.358 1.252 -0.726 ...

$ var\_73 : num 8.86 20.11 12.17 24.42 4.59 ...

$ var\_74 : num 43.11 7.78 19.73 4.53 -4.53 ...

$ var\_75 : num 18.38 7.05 19.45 15.42 23.35 ...

$ var\_76 : num -2.34 3.27 4.5 11.69 1.03 ...

$ var\_77 : num 23.4 23.5 23.2 23.6 19.2 ...

$ var\_78 : num 6.52 5.51 6.32 4.08 7.17 ...

$ var\_79 : num 12.2 13.8 12.8 15.3 14.4 ...

$ var\_80 : num 13.647 2.546 7.473 0.784 2.96 ...

$ var\_81 : num 13.8 18.2 15.8 10.5 13.3 ...

$ var\_82 : num 1.367 0.368 13.353 1.621 -9.259 ...

$ var\_83 : num 2.94 -4.82 10.19 -5.29 -6.71 ...

$ var\_84 : num -4.52 -5.49 5.46 1.6 7.9 ...

$ var\_85 : num 21.5 13.8 19.1 18 14.5 ...

$ var\_86 : num 9.32 -13.59 -4.46 -2.32 7.08 ...

$ var\_87 : num 16.46 11.1 9.54 15.63 20.17 ...

$ var\_88 : num 8 7.9 11.91 4.55 8.01 ...

$ var\_89 : num -1.71 12.23 2.14 7.55 3.8 ...

$ var\_90 : num -21.449 0.477 -22.404 -7.587 -39.8 ...

$ var\_91 : num 6.78 6.89 7.09 7.04 7.01 ...

$ var\_92 : num 11.09 8.09 14.16 14.4 9.36 ...

$ var\_93 : num 9.99 10.96 10.51 10.78 10.43 ...

$ var\_94 : num 14.84 11.76 14.26 7.29 14.06 ...

$ var\_95 : num 0.1812 -1.2722 0.2647 -1.093 0.0213 ...

$ var\_96 : num 8.96 24.79 20.4 11.36 14.72 ...

[list output truncated]

* Typecasting the target variable

train\_df$target=as.factor(train\_df$target)

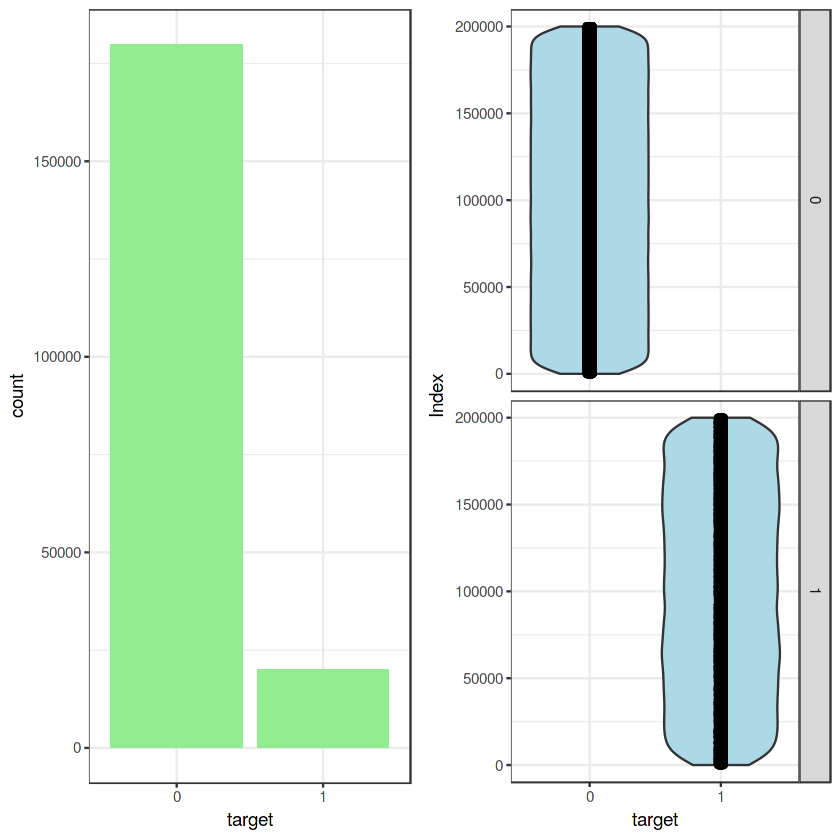
* Target classes count in train data

0 1

179902 20098

0 1

89.951 10.049



From the above plots we have a unbalanced data, where 90% of the data is the data of number of customers those will not make a transaction and 10% of the data is those who will make a transaction.

* **Let us see distribution of train attributes from 3 to 102**

**for (var in names(train\_df)[c(3:102)]){**

**target<-train\_df$target**

**plot<-ggplot(train\_df, aes(x=train\_df[[var]],fill=target)) +**

**geom\_density(kernel='gaussian') + ggtitle(var)+theme\_classic()**

**print(plot)**

**}**

****

* **Let us see distribution of train attributes from 103 to 202**

**for (var in names(train\_df)[c(103:202)]){**

**target<-train\_df$target**

**plot<-ggplot(train\_df, aes(x=train\_df[[var]], fill=target)) +**

**geom\_density(kernel='gaussian') + ggtitle(var)+theme\_classic()**

**print(plot)**

**}**

****

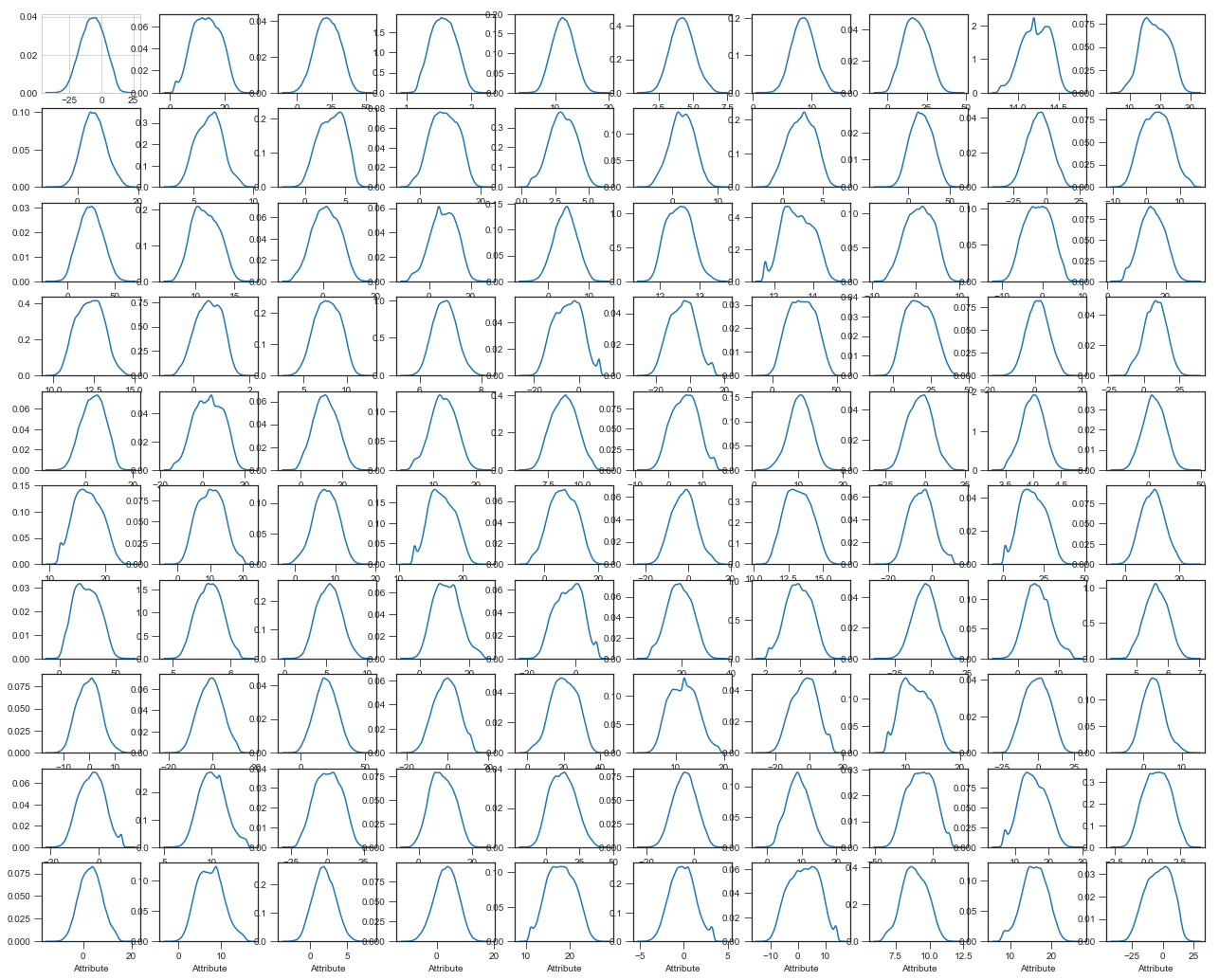
* From above two distributionplots we can observe that there is a considerable number of features which are significantly have different distributions for two target variables. For example like var\_0,var\_1,var\_9,var\_198 var\_180 etc.
* We can observe that there is a considerable number of features which are significantly have same distributions for two target variables. For example like var\_3,var\_7,var\_10,var\_171,var\_185 etc.
* **Let us see distribution of test attributes from 2 to 101**

**plot\_density(test\_df[,c(2:101)], ggtheme = theme\_classic(),geom\_density\_args = list(color='blue'))**

****

* **Let us see distribution of test attributes from 102 to 201**

**plot\_density(test\_df[,c(102:201)], ggtheme = theme\_classic(),geom\_density\_args = list(color='blue'))**

****

* From the above plots we can observe that there is a considerable number of features which are significantly have different distributions. For example like var\_0,var\_1,var\_9,var\_180 var\_198 etc.
* We can observe that there is a considerable number of features which are significantly have same distributions. For example like var\_3,var\_7,var\_10,var\_171,var\_185,var\_192 etc.
* **Let us see distribution of mean values per row and column in train and test dataset**

train\_mean<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=mean)

test\_mean<-apply(test\_df[,-c(1)],MARGIN=1,FUN=mean)

ggplot()+

#Distribution of mean values per row in train data

geom\_density(data=train\_df[,-c(1,2)],aes(x=train\_mean),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of mean values per row in test data

geom\_density(data=test\_df[,-c(1)],aes(x=test\_mean),kernel='gaussian',show.legend=TRUE,color='green')+

labs(x='mean values per row',title="Distribution of mean values per row in train and test dataset")

#Applying the function to find mean values per column in train and test data.

train\_mean<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=mean)

test\_mean<-apply(test\_df[,-c(1)],MARGIN=2,FUN=mean)

ggplot()+

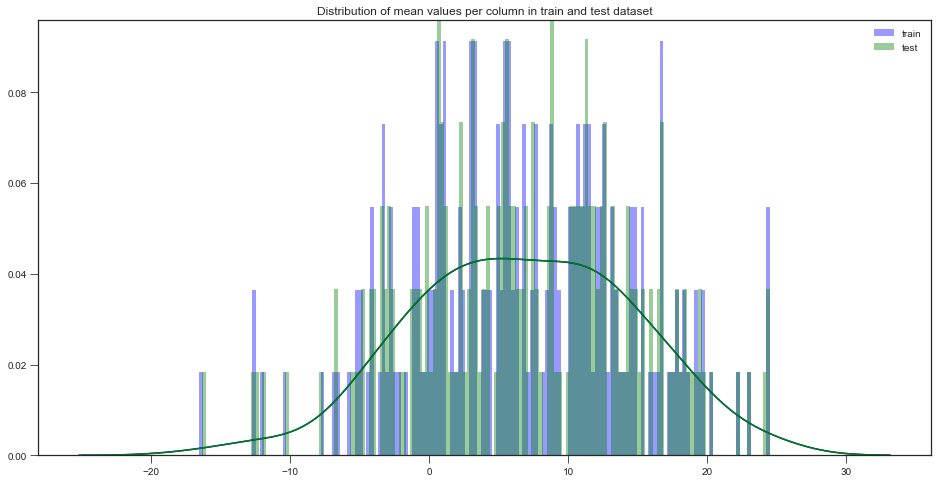
#Distribution of mean values per column in train data

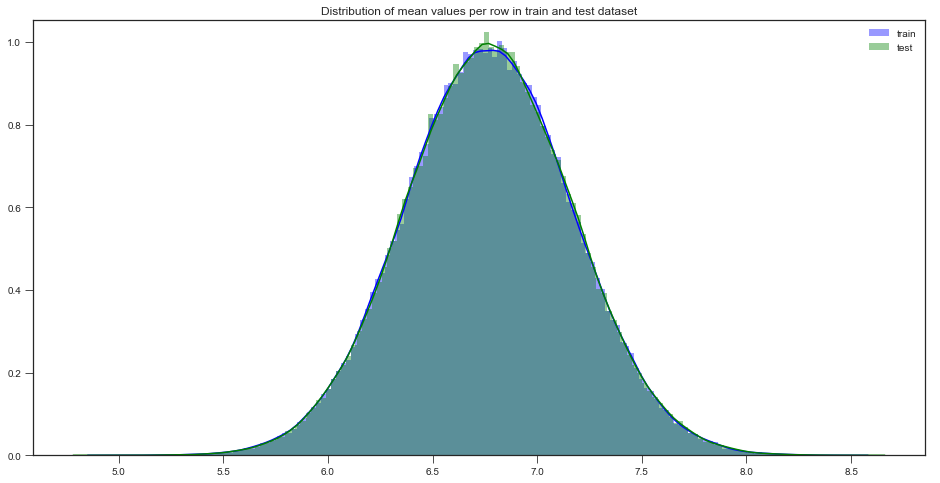
geom\_density(aes(x=train\_mean),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of mean values per column in test data

geom\_density(aes(x=test\_mean),kernel='gaussian',show.legend=TRUE,color='green')+

labs(x='mean values per column',title="Distribution of mean values per row in train and test dataset")





* **Let us see distribution of standard deviation values per row and column in train and test dataset**

train\_sd<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=sd)

test\_sd<-apply(test\_df[,-c(1)],MARGIN=1,FUN=sd)

ggplot()+

#Distribution of sd values per row in train data

geom\_density(data=train\_df[,-c(1,2)],aes(x=train\_sd),kernel='gaussian',show.legend=TRUE,color='red')+theme\_classic()+

#Distribution of mean values per row in test data

geom\_density(data=test\_df[,-c(1)],aes(x=test\_sd),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='sd values per row',title="Distribution of sd values per row in train and test dataset")

#Applying the function to find sd values per column in train and test data.

train\_sd<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=sd)

test\_sd<-apply(test\_df[,-c(1)],MARGIN=2,FUN=sd)

ggplot()+

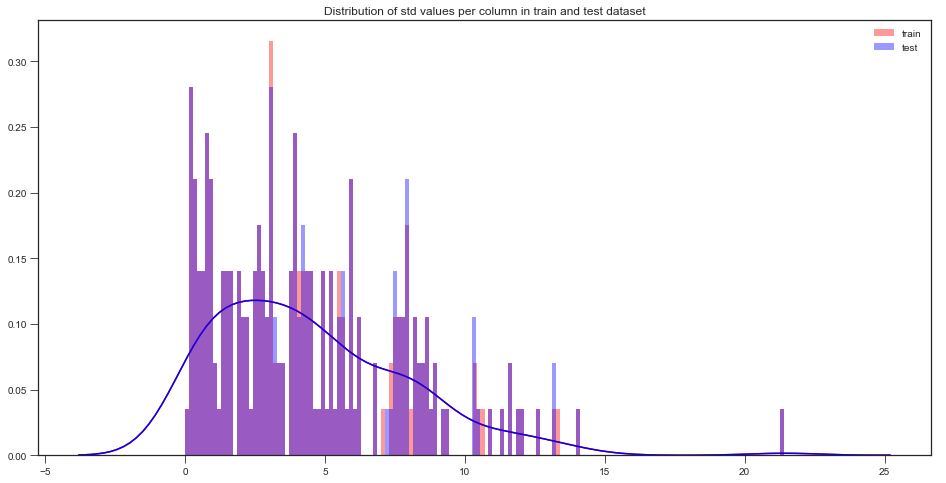
#Distribution of sd values per column in train data

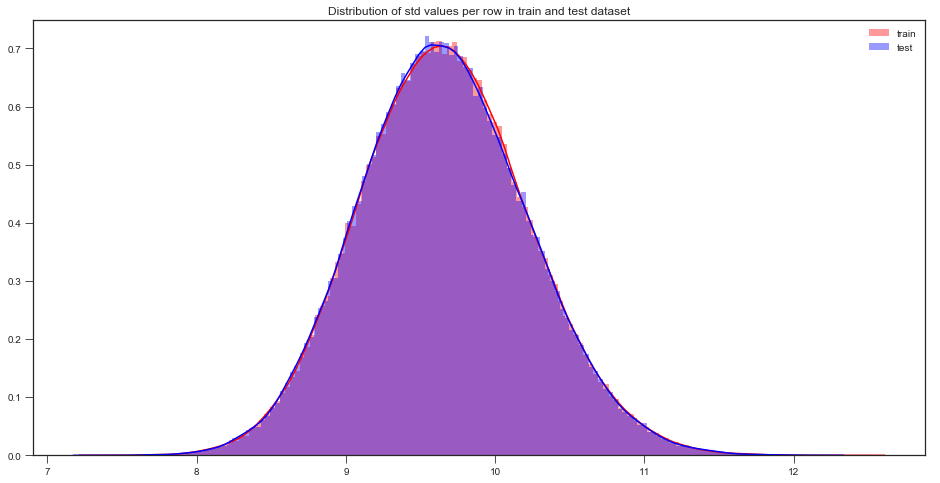
geom\_density(aes(x=train\_sd),kernel='gaussian',show.legend=TRUE,color='red')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_sd),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='sd values per column',title="Distribution of std values per column in train and test dataset")





* **Let us see distribution of skewness values per row and column in train and test dataset**

#Applying the function to find skewness values per row in train and test data.

train\_skew<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=skewness)

test\_skew<-apply(test\_df[,-c(1)],MARGIN=1,FUN=skewness)

ggplot()+

#Distribution of skewness values per row in train data

geom\_density(aes(x=train\_skew),kernel='gaussian',show.legend=TRUE,color='green')+theme\_classic()+

#Distribution of skewness values per column in test data

geom\_density(aes(x=test\_skew),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='skewness values per row',title="Distribution of skewness values per row in train and test dataset")

#Applying the function to find skewness values per column in train and test data.

train\_skew<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=skewness)

test\_skew<-apply(test\_df[,-c(1)],MARGIN=2,FUN=skewness)

ggplot()+

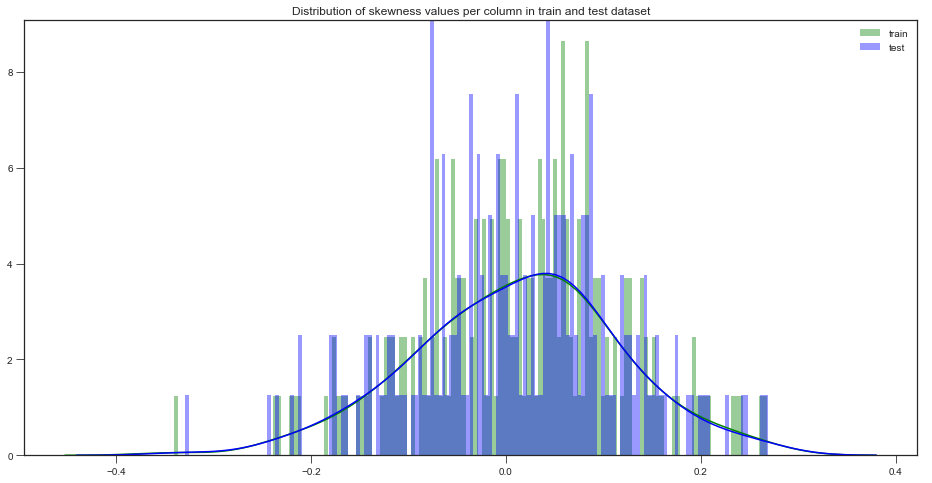
#Distribution of skewness values per column in train data

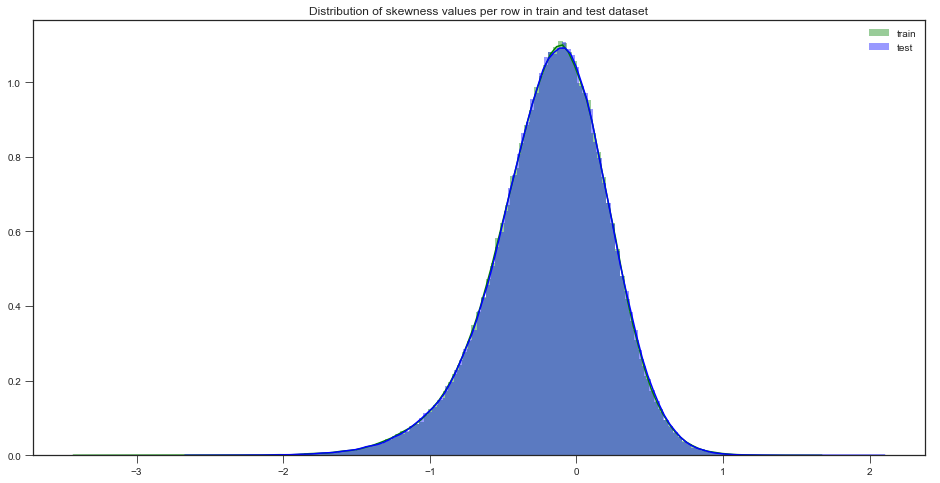
geom\_density(aes(x=train\_skew),kernel='gaussian',show.legend=TRUE,color='green')+theme\_classic()+

#Distribution of skewness values per column in test data

geom\_density(aes(x=test\_skew),kernel='gaussian',show.legend=TRUE,color='blue')+

labs(x='skewness values per column',title="Distribution of skewness values per column in train and test dataset")





* **Let us see distribution of kurtosis values per row and column in train and test dataset**

train\_kurtosis<-apply(train\_df[,-c(1,2)],MARGIN=1,FUN=kurtosis)

test\_kurtosis<-apply(test\_df[,-c(1)],MARGIN=1,FUN=kurtosis)

ggplot()+

#Distribution of sd values per column in train data

geom\_density(aes(x=train\_kurtosis),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_kurtosis),kernel='gaussian',show.legend=TRUE,color='red')+

labs(x='kurtosis values per row',title="Distribution of kurtosis values per row in train and test dataset")

#Applying the function to find kurtosis values per column in train and test data.

train\_kurtosis<-apply(train\_df[,-c(1,2)],MARGIN=2,FUN=kurtosis)

test\_kurtosis<-apply(test\_df[,-c(1)],MARGIN=2,FUN=kurtosis)

ggplot()+

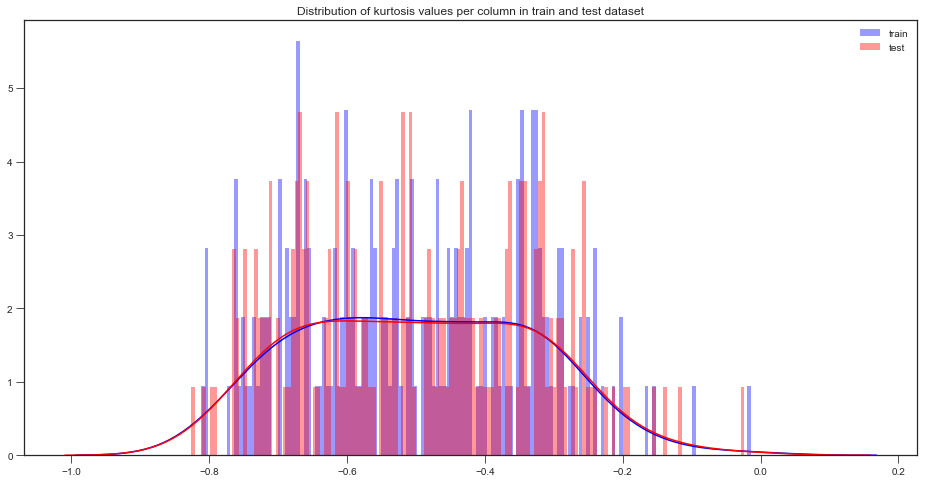
#Distribution of sd values per column in train data

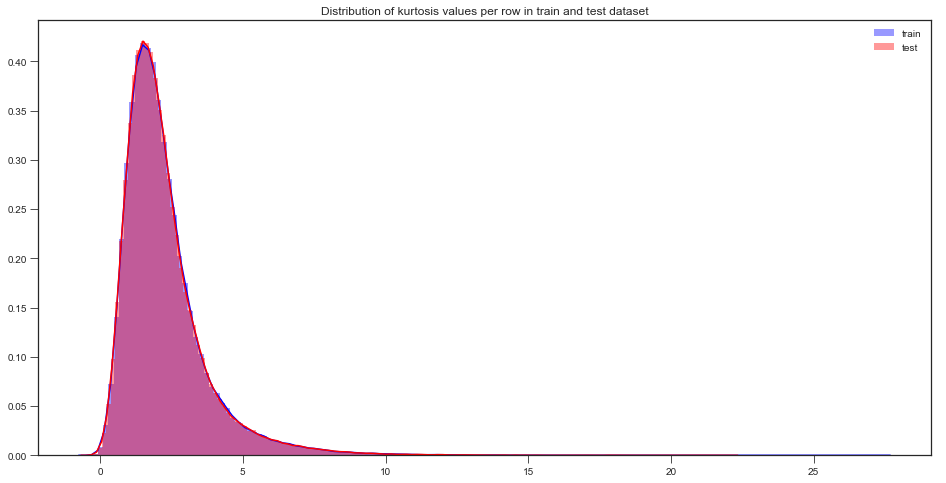
geom\_density(aes(x=train\_kurtosis),kernel='gaussian',show.legend=TRUE,color='blue')+theme\_classic()+

#Distribution of sd values per column in test data

geom\_density(aes(x=test\_kurtosis),kernel='gaussian',show.legend=TRUE,color='red')+

labs(x='kurtosis values per column',title="Distribution of kurtosis values per column in train and test dataset")





* **Let us do Missing value analysis**

#Finding the missing values in train data

missing\_val<-data.frame(missing\_val=apply(train\_df,2,function(x){sum(is.na(x))}))

missing\_val<-sum(missing\_val)

missing\_val

#Finding the missing values in test data

missing\_val<-data.frame(missing\_val=apply(test\_df,2,function(x){sum(is.na(x))}))

missing\_val<-sum(missing\_val)

missing\_val

Missing values in train data : 0

Missing values in test data : 0

No Missing values to be found

* **Let us see correlation between the attributes**

**Correlations in train data**

**train\_df$target<-as.numeric(train\_df$target)**

**train\_correlations<-cor(train\_df[,c(2:202)])**

**train\_correlations**

level\_0 level\_1 0

0 var\_75 var\_191 2.703975e-08

1 var\_191 var\_75 2.703975e-08

2 var\_173 var\_6 5.942735e-08

3 var\_6 var\_173 5.942735e-08

4 var\_126 var\_109 1.313947e-07

5 var\_109 var\_126 1.313947e-07

6 var\_144 var\_27 1.772502e-07

7 var\_27 var\_144 1.772502e-07

8 var\_177 var\_100 3.116544e-07

9 var\_100 var\_177 3.116544e-07

level\_0 level\_1 0

39790 var\_183 var\_189 0.009359

39791 var\_189 var\_183 0.009359

39792 var\_174 var\_81 0.009490

39793 var\_81 var\_174 0.009490

39794 var\_81 var\_165 0.009714

39795 var\_165 var\_81 0.009714

39796 var\_53 var\_148 0.009788

39797 var\_148 var\_53 0.009788

39798 var\_26 var\_139 0.009844

39799 var\_139 var\_26 0.009844

We can observe that the correlation between the train attributes is very small.

**Correlations in test data**

**test\_correlations<-cor(test\_df[,c(2:201)])**

**test\_correlations**

level\_0 level\_1 0

0 var\_154 var\_175 1.477268e-07

1 var\_175 var\_154 1.477268e-07

2 var\_188 var\_113 1.639749e-07

3 var\_113 var\_188 1.639749e-07

4 var\_131 var\_8 4.695407e-07

5 var\_8 var\_131 4.695407e-07

6 var\_60 var\_189 9.523709e-07

7 var\_189 var\_60 9.523709e-07

8 var\_159 var\_96 1.147835e-06

9 var\_96 var\_159 1.147835e-06

level\_0 level\_1 0

39790 var\_122 var\_164 0.008513

39791 var\_164 var\_122 0.008513

39792 var\_164 var\_2 0.008614

39793 var\_2 var\_164 0.008614

39794 var\_31 var\_132 0.008714

39795 var\_132 var\_31 0.008714

39796 var\_96 var\_143 0.008829

39797 var\_143 var\_96 0.008829

39798 var\_139 var\_75 0.009868

39799 var\_75 var\_139 0.009868

We can observe that the correlation between the test attributes is very small.

**Feature engineering**

Let us do some feature engineering by using

* Permutation importance
* Partial dependence plots

**Variable importance**

Variable importance is used to see top features in dataset based on mean decreases gini.

Let us build simple model to find features which are more important.

#Split the training data using simple random sampling

train\_index<-sample(1:nrow(train\_df),0.75\*nrow(train\_df))

#train data

train\_data<-train\_df[train\_index,]

#validation data

valid\_data<-train\_df[-train\_index,]

#dimension of train and validation data

dim(train\_data)

dim(valid\_data)

Result:

150000 202

50000 202

**Random forest classifier**

#Training the Random forest classifier

set.seed(2732)

#convert to int to factor

train\_data$target<-as.factor(train\_data$target)

#setting the mtry

mtry<-floor(sqrt(200))

#setting the tunegrid

tuneGrid<-expand.grid(.mtry=mtry)

#fitting the ranndom forest

rf<-randomForest(target~.,train\_data[,-c(1)],mtry=mtry,ntree=10,importance=TRUE)

**Feature importance by random forest**

VarImp<-importance(rf,type=2)

VarImp

A matrix: 200 × 1 of type dbl

MeanDecreaseGini

var\_0 182.05386

var\_1 166.18712

var\_2 184.47371

var\_3 125.99692

var\_4 117.87624

var\_5 152.54793

var\_6 199.89162

var\_7 104.03827

var\_8 118.52355

var\_9 150.55568

var\_10 109.91257

var\_11 104.65905

var\_12 259.74504

var\_13 187.70103

var\_14 104.69860

var\_15 125.27934

var\_16 109.06403

var\_17 96.12009

var\_18 153.55425

var\_19 102.51477

var\_20 104.49461

var\_21 181.04580

var\_22 202.74211

var\_23 116.13210

var\_24 135.96180

var\_25 117.22332

var\_26 221.99147

var\_27 119.51102

var\_28 119.17526

var\_29 99.10305

⋮ ⋮

var\_170 181.88053

var\_171 131.76384

var\_172 136.79262

var\_173 148.58144

var\_174 232.19529

var\_175 123.37790

var\_176 107.91208

var\_177 163.95640

var\_178 107.89886

var\_179 190.91391

var\_180 134.37007

var\_181 100.42960

var\_182 94.66617

var\_183 102.94877

var\_184 165.84735

var\_185 105.54899

var\_186 116.12305

var\_187 112.72131

var\_188 141.37134

var\_189 106.29706

var\_190 187.20892

var\_191 167.29808

var\_192 125.99321

var\_193 116.38995

var\_194 106.05944

var\_195 125.59821

var\_196 111.33002

var\_197 139.14980

var\_198 205.00126

var\_199 114.45181

**We can observe that the top important features are var\_12, var\_26, var\_22,v var\_174, var\_198 and so on based on Mean decrease gini.**

**Partial dependence plots**

Partial dependence plot gives a graphical depiction of the marginal effect of a variable on the class probability or classification.While feature importance shows what variables most affect predictions, but partial dependence plots show how a feature affects predictions.

Let us calculate partial dependence plots on random forest

Let us plot the learned dtr model

**Partial dependence plot**

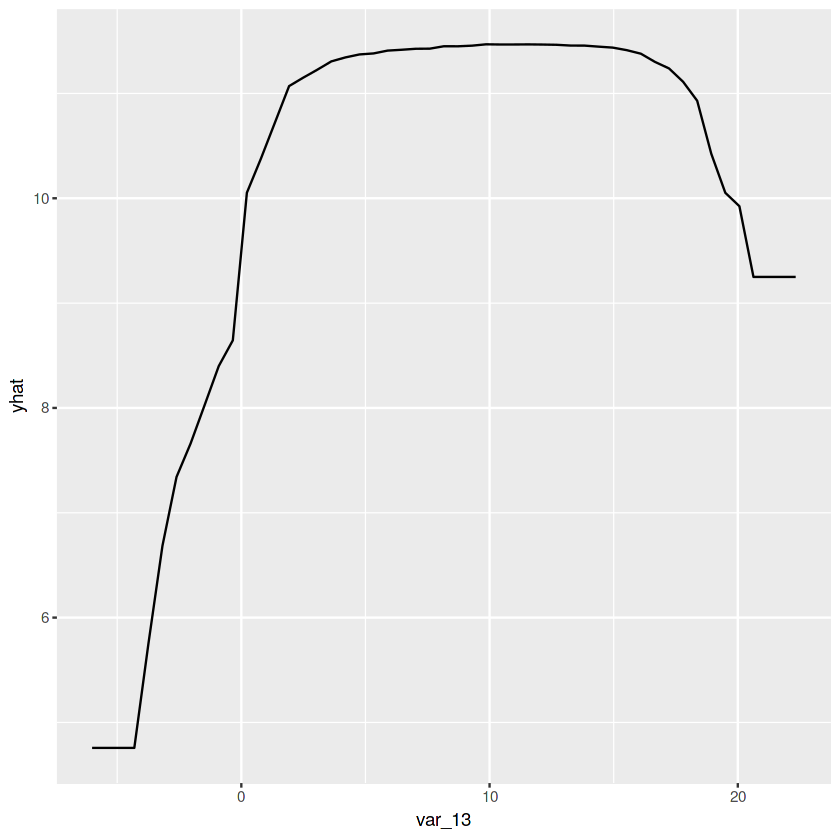
Let us see impact of the main features which are discovered in the previous section by using pdp package.

**#We will plot "var\_13"**

**par.var\_13 <- partial(rf, pred.var = c("var\_13"), chull = TRUE)**

**plot.var\_13 <- autoplot(par.var\_13, contour = TRUE)**

**plot.var\_13**

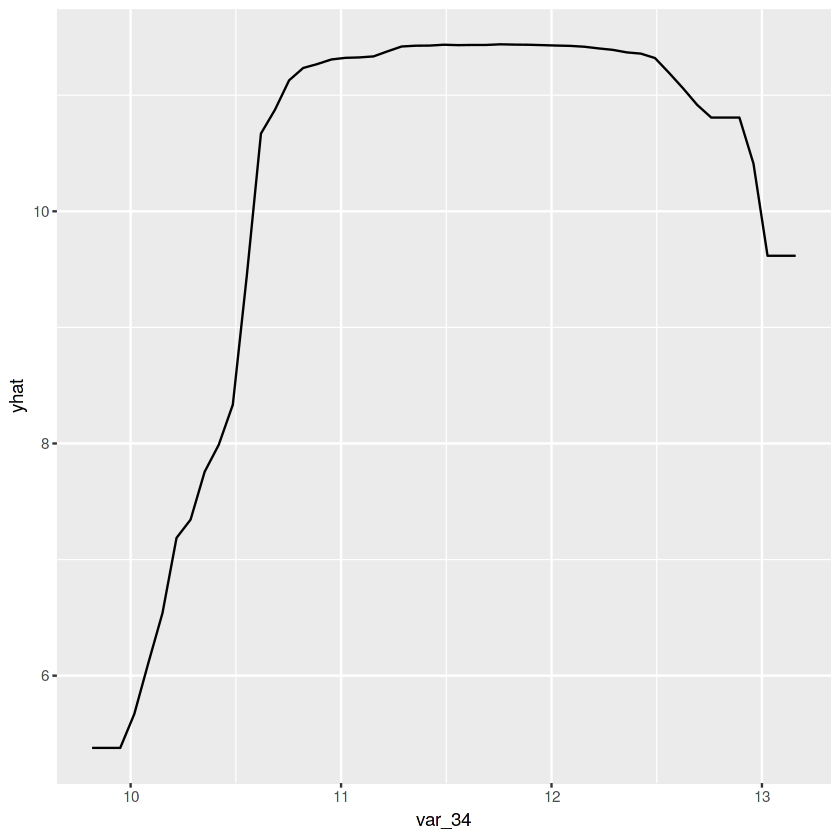
****

* The y\_axis does not show the predictor value instead how the value changing with the change in given predictor variable.
* The blue shaded area indicates the level of confidence of 'var\_13'
* On y-axis having a positive value means for that particular value of predictor variable it is less likely to predict the correct class and having a positive value means it has positive impact on predicting the correct class.

**#We will plot "var\_34"**

**par.var\_34 <- partial(rf, pred.var = c("var\_34"), chull = TRUE)**

**plot.var\_34 <- autoplot(par.var\_34, contour = TRUE)plot.var\_34**

****

* The y\_axis does not show the predictor value instead how the value changing with the change in given predictor variable.
* The blue shaded area indicates the level of confidence of 'var\_34'.
* On y-axis having a positive value means for that particular value of predictor variable it is less likely to predict the correct class and having a positive value means it has positive impact on predicting the correct class.

**Handling of imbalanced data**

Now we are going to explore 5 different approaches for dealing with imbalanced datasets.

* Change the performance metric
* Oversample minority class
* Undersample majority class
* Synthetic Minority Oversampling Technique(SMOTE)
* LightGBM

Now let us start with simple Logistic regression model.

**Split the data using CreateDataPartition**

**#Split the data using CreateDataPartition**

**set.seed(689)**

**#train.index<-createDataPartition(train\_df$target,p=0.8,list=FALSE)**

**train.index<-sample(1:nrow(train\_df),0.8\*nrow(train\_df))**

**#train data**

**train.data<-train\_df[train.index,]**

**#validation data**

**valid.data<-train\_df[-train.index,]**

**#dimension of train data**

**dim(train.data)**

**#dimension of validation data**

**dim(valid.data)**

**#target classes in train data**

**table(train.data$target)**

**#target classes in validation data**

**table(valid.data$target)**

Result:

160000 202

40000 202

1 2

143850 16150

1 2

36052 3948

**Logistic Regression model**

**Training and validation dataset**

#Training dataset

X\_t<-as.matrix(train.data[,-c(1,2)])

y\_t<-as.matrix(train.data$target)

#validation dataset

X\_v<-as.matrix(valid.data[,-c(1,2)])

y\_v<-as.matrix(valid.data$target)

#test dataset

test<-as.matrix(test\_df[,-c(1)])

#Logistic regression model

set.seed(667) # to reproduce results

lr\_model <-glmnet(X\_t,y\_t, family = "binomial")

summary(lr\_model)

**Length Class Mode**

**a0 61 -none- numeric**

**beta 12200 dgCMatrix S4**

**df 61 -none- numeric**

**dim 2 -none- numeric**

**lambda 61 -none- numeric**

**dev.ratio 61 -none- numeric**

**nulldev 1 -none- numeric**

**npasses 1 -none- numeric**

**jerr 1 -none- numeric**

**offset 1 -none- logical**

**classnames 2 -none- character**

**call 4 -none- call**

**nobs 1 -none- numeric**

**Cross validation prediction**

set.seed(8909)

cv\_lr <- cv.glmnet(X\_t,y\_t,family = "binomial", type.measure = "class")

cv\_lr

$lambda

[1] 2.506402e-02 2.283740e-02 2.080859e-02 1.896001e-02 1.727565e-02

[6] 1.574093e-02 1.434255e-02 1.306840e-02 1.190744e-02 1.084961e-02

[11] 9.885764e-03 9.007540e-03 8.207335e-03 7.478218e-03 6.813874e-03

[16] 6.208548e-03 5.656998e-03 5.154446e-03 4.696539e-03 4.279311e-03

[21] 3.899149e-03 3.552759e-03 3.237142e-03 2.949563e-03 2.687532e-03

[26] 2.448779e-03 2.231236e-03 2.033020e-03 1.852412e-03 1.687849e-03

[31] 1.537905e-03 1.401282e-03 1.276796e-03 1.163369e-03 1.060018e-03

[36] 9.658490e-04 8.800456e-04 8.018648e-04 7.306294e-04 6.657223e-04

[41] 6.065813e-04 5.526943e-04 5.035945e-04 4.588565e-04 4.180930e-04

[46] 3.809508e-04 3.471081e-04 3.162720e-04 2.881753e-04 2.625746e-04

[51] 2.392482e-04 2.179940e-04 1.986280e-04 1.809825e-04 1.649045e-04

[56] 1.502548e-04 1.369066e-04 1.247442e-04 1.136623e-04 1.035648e-04

[61] 9.436441e-05

$cvm

[1] 0.10093750 0.10093750 0.10093750 0.10093750 0.10093750 0.10093750

[7] 0.10093750 0.10093750 0.10093750 0.10093750 0.10091250 0.10085000

[13] 0.10068125 0.10028750 0.09962500 0.09871250 0.09780625 0.09663125

[19] 0.09574375 0.09461875 0.09363125 0.09255625 0.09159375 0.09091250

[25] 0.09010625 0.08946250 0.08885625 0.08843125 0.08795000 0.08762500

[31] 0.08733750 0.08712500 0.08671875 0.08656875 0.08660000 0.08653125

[37] 0.08631875 0.08619375 0.08616875 0.08613125 0.08609375 0.08611250

[43] 0.08610625 0.08605000 0.08605000 0.08598750 0.08593750 0.08590000

[49] 0.08593125 0.08590625 0.08591875 0.08592500 0.08596250 0.08597500

[55] 0.08601250 0.08597500 0.08596250 0.08595000 0.08591250 0.08592500

[61] 0.08591875

$cvsd

[1] 0.0006551081 0.0006551081 0.0006551081 0.0006551081 0.0006551081

[6] 0.0006551081 0.0006551081 0.0006551081 0.0006551081 0.0006551081

[11] 0.0006552539 0.0006503338 0.0006554029 0.0006445687 0.0006644494

[16] 0.0006875758 0.0007000403 0.0007111128 0.0006339943 0.0006533468

[21] 0.0006559986 0.0005515485 0.0005623650 0.0005950607 0.0006020112

[26] 0.0005880907 0.0005891857 0.0006286318 0.0006841966 0.0006843362

[31] 0.0006600715 0.0006974034 0.0006320608 0.0006271110 0.0006235539

[36] 0.0006323354 0.0006441814 0.0006398547 0.0006199833 0.0006202213

[41] 0.0006288251 0.0006419777 0.0006466426 0.0006070906 0.0005930588

[46] 0.0005997540 0.0005728977 0.0005721548 0.0006015352 0.0006066222

[51] 0.0006121633 0.0006349076 0.0006358230 0.0006473235 0.0006364097

[56] 0.0006520667 0.0006411930 0.0006331277 0.0006379356 0.0006304485

[61] 0.0006352664

$cvup

[1] 0.10159261 0.10159261 0.10159261 0.10159261 0.10159261 0.10159261

[7] 0.10159261 0.10159261 0.10159261 0.10159261 0.10156775 0.10150033

[13] 0.10133665 0.10093207 0.10028945 0.09940008 0.09850629 0.09734236

[19] 0.09637774 0.09527210 0.09428725 0.09310780 0.09215611 0.09150756

[25] 0.09070826 0.09005059 0.08944544 0.08905988 0.08863420 0.08830934

[31] 0.08799757 0.08782240 0.08735081 0.08719586 0.08722355 0.08716359

[37] 0.08696293 0.08683360 0.08678873 0.08675147 0.08672258 0.08675448

[43] 0.08675289 0.08665709 0.08664306 0.08658725 0.08651040 0.08647215

[49] 0.08653279 0.08651287 0.08653091 0.08655991 0.08659832 0.08662232

[55] 0.08664891 0.08662707 0.08660369 0.08658313 0.08655044 0.08655545

[61] 0.08655402

$cvlo

[1] 0.10028239 0.10028239 0.10028239 0.10028239 0.10028239 0.10028239

[7] 0.10028239 0.10028239 0.10028239 0.10028239 0.10025725 0.10019967

[13] 0.10002585 0.09964293 0.09896055 0.09802492 0.09710621 0.09592014

[19] 0.09510976 0.09396540 0.09297525 0.09200470 0.09103139 0.09031744

[25] 0.08950424 0.08887441 0.08826706 0.08780262 0.08726580 0.08694066

[31] 0.08667743 0.08642760 0.08608669 0.08594164 0.08597645 0.08589891

[37] 0.08567457 0.08555390 0.08554877 0.08551103 0.08546492 0.08547052

[43] 0.08545961 0.08544291 0.08545694 0.08538775 0.08536460 0.08532785

[49] 0.08532971 0.08529963 0.08530659 0.08529009 0.08532668 0.08532768

[55] 0.08537609 0.08532293 0.08532131 0.08531687 0.08527456 0.08529455

[61] 0.08528348

$nzero

s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19

0 2 3 7 13 23 31 41 48 58 66 76 85 93 103 113 119 125 128 134

s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39

137 145 149 151 153 162 164 165 169 171 174 177 179 181 181 184 184 186 187 188

s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53 s54 s55 s56 s57 s58 s59

188 189 190 193 194 194 195 195 195 195 197 197 197 198 198 198 198 198 198 199

s60

199

$name

class

"Misclassification Error"

$glmnet.fit

Call: glmnet(x = X\_t, y = y\_t, family = "binomial")

Df %Dev Lambda

[1,] 0 1.251e-12 2.506e-02

[2,] 2 1.828e-03 2.284e-02

[3,] 3 5.371e-03 2.081e-02

[4,] 7 1.082e-02 1.896e-02

[5,] 13 2.063e-02 1.728e-02

[6,] 23 3.424e-02 1.574e-02

[7,] 31 5.196e-02 1.434e-02

[8,] 41 7.056e-02 1.307e-02

[9,] 48 8.968e-02 1.191e-02

[10,] 58 1.081e-01 1.085e-02

[11,] 66 1.259e-01 9.886e-03

[12,] 76 1.429e-01 9.008e-03

[13,] 85 1.591e-01 8.207e-03

[14,] 93 1.738e-01 7.478e-03

[15,] 103 1.874e-01 6.814e-03

[16,] 113 2.000e-01 6.209e-03

[17,] 119 2.117e-01 5.657e-03

[18,] 125 2.220e-01 5.154e-03

[19,] 128 2.310e-01 4.697e-03

[20,] 134 2.389e-01 4.279e-03

[21,] 137 2.460e-01 3.899e-03

[22,] 145 2.523e-01 3.553e-03

[23,] 149 2.578e-01 3.237e-03

[24,] 151 2.626e-01 2.950e-03

[25,] 153 2.668e-01 2.688e-03

[26,] 162 2.705e-01 2.449e-03

[27,] 164 2.737e-01 2.231e-03

[28,] 165 2.765e-01 2.033e-03

[29,] 169 2.789e-01 1.852e-03

[30,] 171 2.809e-01 1.688e-03

[31,] 174 2.827e-01 1.538e-03

[32,] 177 2.842e-01 1.401e-03

[33,] 179 2.855e-01 1.277e-03

[34,] 181 2.866e-01 1.163e-03

[35,] 181 2.876e-01 1.060e-03

[36,] 184 2.884e-01 9.658e-04

[37,] 184 2.891e-01 8.800e-04

[38,] 186 2.896e-01 8.019e-04

[39,] 187 2.901e-01 7.306e-04

[40,] 188 2.906e-01 6.657e-04

[41,] 188 2.909e-01 6.066e-04

[42,] 189 2.912e-01 5.527e-04

[43,] 190 2.914e-01 5.036e-04

[44,] 193 2.917e-01 4.589e-04

[45,] 194 2.918e-01 4.181e-04

[46,] 194 2.920e-01 3.810e-04

[47,] 195 2.921e-01 3.471e-04

[48,] 195 2.922e-01 3.163e-04

[49,] 195 2.923e-01 2.882e-04

[50,] 195 2.924e-01 2.626e-04

[51,] 197 2.924e-01 2.392e-04

[52,] 197 2.925e-01 2.180e-04

[53,] 197 2.925e-01 1.986e-04

[54,] 198 2.926e-01 1.810e-04

[55,] 198 2.926e-01 1.649e-04

[56,] 198 2.926e-01 1.503e-04

[57,] 198 2.926e-01 1.369e-04

[58,] 198 2.926e-01 1.247e-04

[59,] 198 2.927e-01 1.137e-04

[60,] 199 2.927e-01 1.036e-04

[61,] 199 2.927e-01 9.436e-05

$lambda.min

[1] 0.000316272

$lambda.1se

[1] 0.0008800456

attr(,"class")

[1] "cv.glmnet"

**Plotting the missclassification error vs log(lambda) where lambda is regularization parameter**

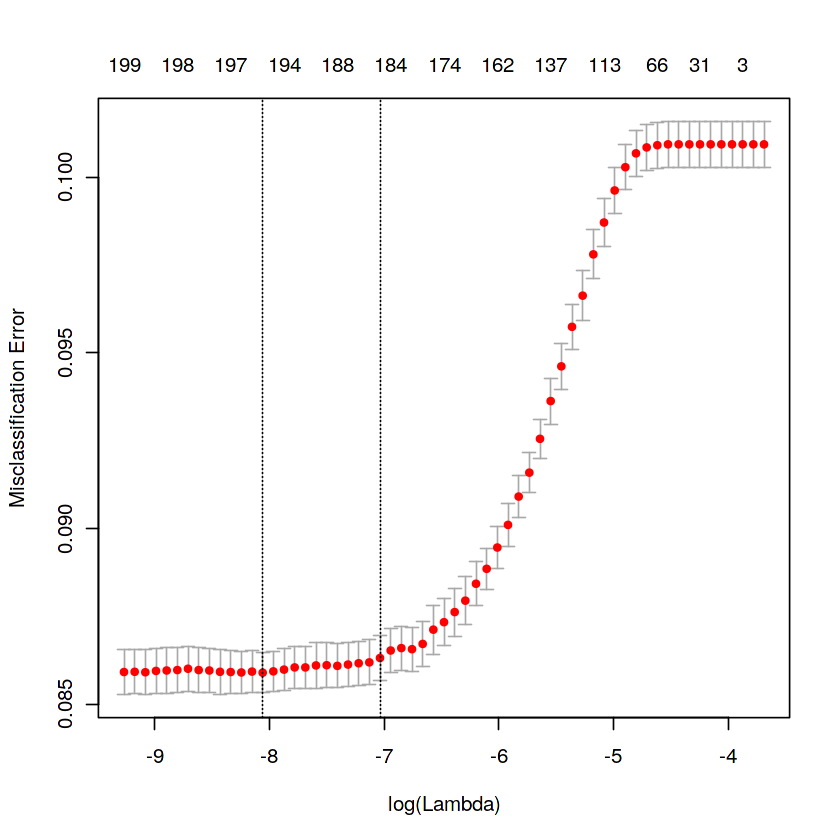
**#Minimum lambda**

**cv\_lr$lambda.min**

**#plot the auc score vs log(lambda)**

**plot(cv\_lr)**

0.000316272006587771



We can observe that miss classification error increases as increasing the log(Lambda).

**Model performance on validation dataset**

#Model performance on validation dataset

set.seed(5363)

cv\_predict.lr<-predict(cv\_lr,X\_v,s = "lambda.min", type = "class")

cv\_predict.lr

A matrix: 40000 × 1 of type chr

1

2 2

3 1

8 1

9 1

19 1

21 1

25 1

28 1

30 1

32 1

36 1

41 1

44 1

47 1

50 1

51 1

56 1

61 1

72 1

89 1

92 1

93 1

94 1

100 1

102 1

103 1

105 1

112 1

115 1

118 1

⋮ ⋮

199866 1

199876 1

199877 1

199878 1

199894 1

199897 1

199905 1

199907 1

199913 1

199914 1

199916 1

199919 1

199922 1

199937 1

199938 1

199939 1

199944 1

199950 1

199952 1

199955 1

199957 1

199959 1

199965 1

199966 1

199967 1

199980 1

199988 1

199990 1

199991 1

199995 1

**Accuracy of the model is not the best metric to use when evaluating the imbalanced datasets as it may be misleading. So, we are going to change the performance metric.**

**Confusion Matrix**

#Confusion matrix

set.seed(689)

#actual target variable

target<-valid.data$target

#convert to factor

target<-as.factor(target)

#predicted target variable

#convert to factor

cv\_predict.lr<-as.factor(cv\_predict.lr)

confusionMatrix(data=cv\_predict.lr,reference=target)

onvert to factor

cv\_predict.lr<-as.factor(cv\_predict.lr)

confusionMatrix(data=cv\_predict.lr,reference=target)

Confusion Matrix and Statistics

Reference

Prediction 1 2

1 35618 2973

2 434 975

Accuracy : 0.9148

95% CI : (0.912, 0.9175)

No Information Rate : 0.9013

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.3292

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.9880

Specificity : 0.2470

Pos Pred Value : 0.9230

Neg Pred Value : 0.6920

Prevalence : 0.9013

Detection Rate : 0.8904

Detection Prevalence : 0.9648

Balanced Accuracy : 0.6175

'Positive' Class : 1

**Reciever operating characteristics(ROC)-Area under curve(AUC) score and curve**

set.seed(892)

cv\_predict.lr<-as.numeric(cv\_predict.lr)

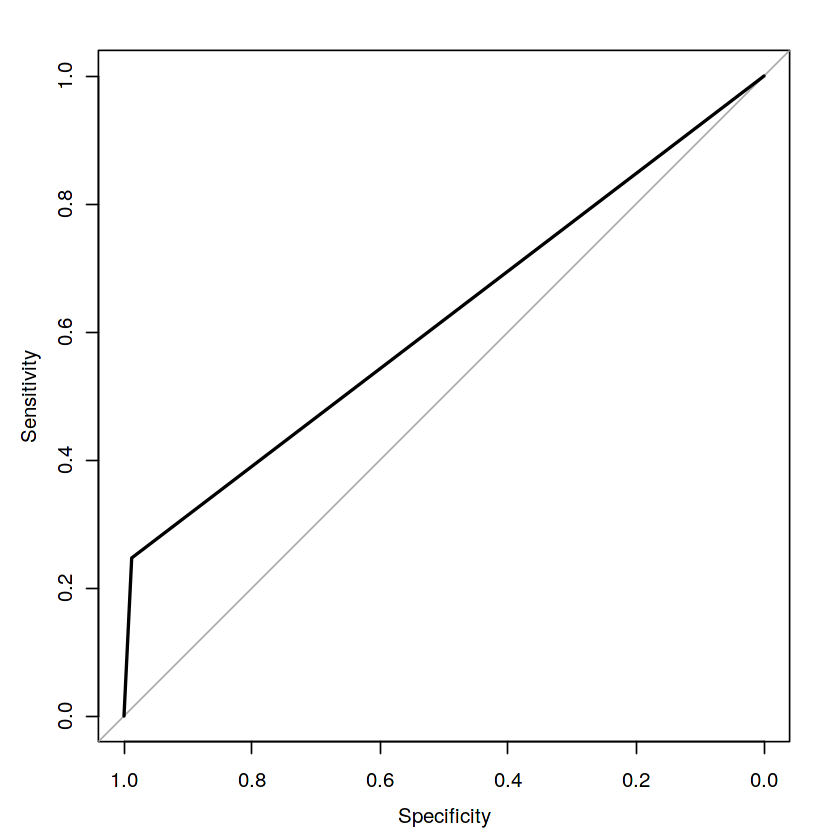
roc(data=valid.data[,-c(1,2)],response=target,predictor=cv\_predict.lr,auc=TRUE,plot=TRUE)

Call:

roc.default(response = target, predictor = cv\_predict.lr, auc = TRUE, plot = TRUE, data = valid.data[, -c(1, 2)])

Data: cv\_predict.lr in 36052 controls (target 1) < 3948 cases (target 2).

Area under the curve: 0.6175



**Model performance on test data**

set.seed(763)

lr\_pred<-predict(lr\_model,test,type='class')

**Oversample minority class:**

* It can be defined as adding more copies of minority class.
* It can be a good choice when we don't have a ton of data to work with.
* Drawback is that we are adding information.This may leads to overfitting and poor performance on test data.

**Undersample majority class:**

* It can be defined as removing some observations of the majority class.
* It can be a good choice when we have a ton of data -think million of rows.
* Drawback is that we are removing information that may be valuable.This may leads to underfitting and poor performance on test data.

Both Oversampling and undersampling techniques have some drawbacks. So, we are not going to use this models for this problem and also we will use other best algorithms.

**Random Oversampling Examples(ROSE)**

It creates a sample of synthetic data by enlarging the features space of minority and majority class examples.

**#Random Oversampling Examples(ROSE)**

set.seed(699)

train.rose <- ROSE(target~., data =train.data[,-c(1)],seed=32)$data

#target classes in balanced train data

table(train.rose$target)

valid.rose <- ROSE(target~., data =valid.data[,-c(1)],seed=42)$data

#target classes in balanced valid data

table(valid.rose$target)

**1 2**

**79990 80010**

**1 2**

**20012 19988**

Let us see how baseline logistic regression model performs on synthetic data points.

**#Logistic regression model**

**set.seed(462)**

**lr\_rose <-glmnet(as.matrix(train.rose),as.matrix(train.rose$target), family = "binomial")**

**summary(lr\_rose)**

Length Class Mode

a0 72 -none- numeric

beta 14472 dgCMatrix S4

df 72 -none- numeric

dim 2 -none- numeric

lambda 72 -none- numeric

dev.ratio 72 -none- numeric

nulldev 1 -none- numeric

npasses 1 -none- numeric

jerr 1 -none- numeric

offset 1 -none- logical

classnames 2 -none- character

call 4 -none- call

nobs 1 -none- numeric

**Cross validation prediction**

**set.seed(473)**

**cv\_rose = cv.glmnet(as.matrix(valid.rose),as.matrix(valid.rose$target),family = "binomial", type.measure = "class")**

**cv\_rose**

$lambda

[1] 0.4999999100 0.4555812961 0.4151087094 0.3782315958 0.3446305433

[6] 0.3140145157 0.2861183316 0.2607003676 0.2375404655 0.2164380253

[11] 0.1972102676 0.1796906510 0.1637274288 0.1491823353 0.1359293878

[16] 0.1238537957 0.1128509658 0.1028255971 0.0936908544 0.0853676171

[21] 0.0777837933 0.0708736955 0.0645774717 0.0588405871 0.0536133516

[26] 0.0488504892 0.0445107464 0.0405565343 0.0369536036 0.0336707473

[31] 0.0306795309 0.0279540460 0.0254706856 0.0232079401 0.0211462106

[36] 0.0192676396 0.0175559556 0.0159963329 0.0145752627 0.0132804366

[41] 0.0121006392 0.0110256518 0.0100461632 0.0091536898 0.0083405012

[46] 0.0075995541 0.0069244306 0.0063092833 0.0057487840 0.0052380778

[51] 0.0047727414 0.0043487442 0.0039624138 0.0036104039 0.0032896655

[56] 0.0029974207 0.0027311381 0.0024885113 0.0022674389 0.0020660058

[61] 0.0018824676 0.0017152343 0.0015628577 0.0014240177 0.0012975119

[66] 0.0011822445 0.0010772172 0.0009815202 0.0008943246 0.0008148753

[71] 0.0007424840 0.0006765238

$cvm

[1] 0.5047 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[11] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[21] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[31] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[41] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[51] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[61] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

[71] 0.0000 0.0000

$cvsd

[1] 0.001737575 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[7] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[13] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[19] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[25] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[31] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[37] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[43] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[49] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[55] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[61] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

[67] 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000

$cvup

[1] 0.5064376 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[8] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[22] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[29] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[36] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[43] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[50] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[57] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[64] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[71] 0.0000000 0.0000000

$cvlo

[1] 0.5029624 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[8] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[22] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[29] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[36] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[43] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[50] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[57] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[64] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[71] 0.0000000 0.0000000

$nzero

s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53 s54 s55 s56 s57 s58 s59

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71

1 1 1 1 1 1 1 1 1 1 1 1

$name

class

"Misclassification Error"

$glmnet.fit

Call: glmnet(x = as.matrix(valid.rose), y = as.matrix(valid.rose$target), family = "binomial")

Df %Dev Lambda

[1,] 0 -6.854e-13 0.5000000

[2,] 1 1.228e-01 0.4556000

[3,] 1 2.262e-01 0.4151000

[4,] 1 3.144e-01 0.3782000

[5,] 1 3.904e-01 0.3446000

[6,] 1 4.562e-01 0.3140000

[7,] 1 5.138e-01 0.2861000

[8,] 1 5.642e-01 0.2607000

[9,] 1 6.087e-01 0.2375000

[10,] 1 6.481e-01 0.2164000

[11,] 1 6.831e-01 0.1972000

[12,] 1 7.142e-01 0.1797000

[13,] 1 7.420e-01 0.1637000

[14,] 1 7.669e-01 0.1492000

[15,] 1 7.892e-01 0.1359000

[16,] 1 8.092e-01 0.1239000

[17,] 1 8.272e-01 0.1129000

[18,] 1 8.435e-01 0.1028000

[19,] 1 8.581e-01 0.0936900

[20,] 1 8.713e-01 0.0853700

[21,] 1 8.832e-01 0.0777800

[22,] 1 8.939e-01 0.0708700

[23,] 1 9.037e-01 0.0645800

[24,] 1 9.125e-01 0.0588400

[25,] 1 9.205e-01 0.0536100

[26,] 1 9.277e-01 0.0488500

[27,] 1 9.343e-01 0.0445100

[28,] 1 9.403e-01 0.0405600

[29,] 1 9.457e-01 0.0369500

[30,] 1 9.506e-01 0.0336700

[31,] 1 9.550e-01 0.0306800

[32,] 1 9.591e-01 0.0279500

[33,] 1 9.628e-01 0.0254700

[34,] 1 9.661e-01 0.0232100

[35,] 1 9.692e-01 0.0211500

[36,] 1 9.719e-01 0.0192700

[37,] 1 9.744e-01 0.0175600

[38,] 1 9.767e-01 0.0160000

[39,] 1 9.788e-01 0.0145800

[40,] 1 9.807e-01 0.0132800

[41,] 1 9.824e-01 0.0121000

[42,] 1 9.840e-01 0.0110300

[43,] 1 9.854e-01 0.0100500

[44,] 1 9.867e-01 0.0091540

[45,] 1 9.879e-01 0.0083410

[46,] 1 9.890e-01 0.0076000

[47,] 1 9.900e-01 0.0069240

[48,] 1 9.909e-01 0.0063090

[49,] 1 9.917e-01 0.0057490

[50,] 1 9.924e-01 0.0052380

[51,] 1 9.931e-01 0.0047730

[52,] 1 9.937e-01 0.0043490

[53,] 1 9.943e-01 0.0039620

[54,] 1 9.948e-01 0.0036100

[55,] 1 9.952e-01 0.0032900

[56,] 1 9.957e-01 0.0029970

[57,] 1 9.961e-01 0.0027310

[58,] 1 9.964e-01 0.0024890

[59,] 1 9.967e-01 0.0022670

[60,] 1 9.970e-01 0.0020660

[61,] 1 9.973e-01 0.0018820

[62,] 1 9.975e-01 0.0017150

[63,] 1 9.977e-01 0.0015630

[64,] 1 9.979e-01 0.0014240

[65,] 1 9.981e-01 0.0012980

[66,] 1 9.983e-01 0.0011820

[67,] 1 9.984e-01 0.0010770

[68,] 1 9.986e-01 0.0009815

[69,] 1 9.987e-01 0.0008943

[70,] 1 9.988e-01 0.0008149

[71,] 1 9.989e-01 0.0007425

[72,] 1 9.990e-01 0.0006765

$lambda.min

[1] 0.4555813

$lambda.1se

[1] 0.4555813

attr(,"class")

[1] "cv.glmnet"

**Plotting the misclassification error vs log(lambda) where lambda is regularization parameter**

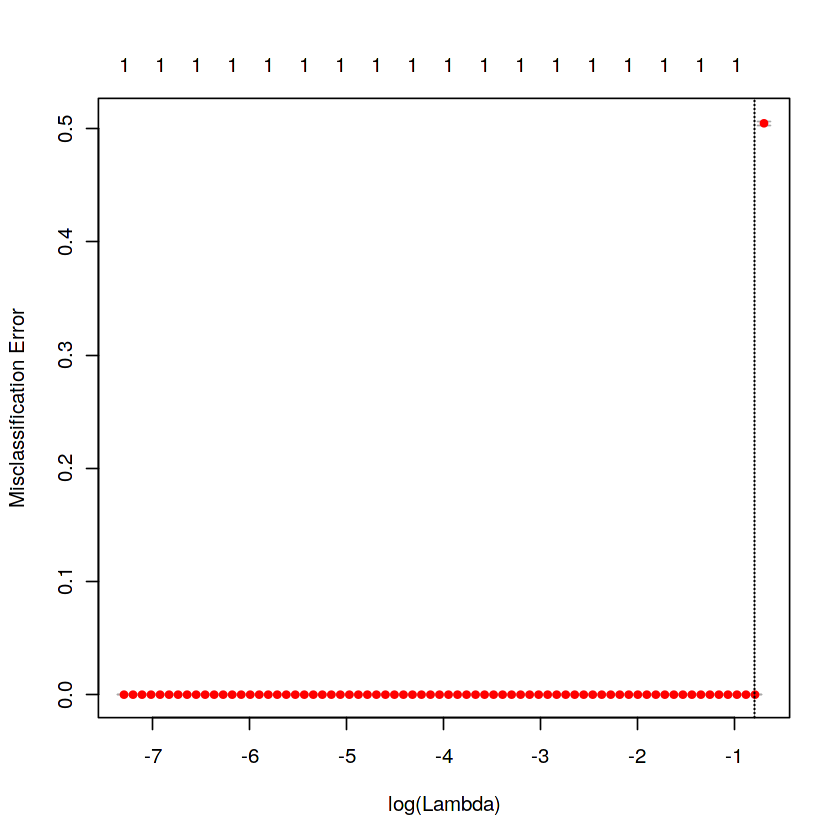
**#Minimum lambda**

**cv\_rose$lambda.min**

**#plot the auc score vs log(lambda)**

**plot(cv\_rose)**

**Result:** 0.455581296093087



**Model performance on validation data**

**#Model performance on validation dataset**

**set.seed(442)**

**cv\_predict.rose<-predict(cv\_rose,as.matrix(valid.rose),s = "lambda.min", type = "class")**

**cv\_predict.rose**

**s")**

**cv\_predict.rose**

**A matrix: 40000 × 1 of type chr**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**1**

**⋮**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**2**

**Confusion Matrix**

**set.seed(478)**

**#actual target variable**

**target<-valid.rose$target**

**#convert to factor**

**target<-as.factor(target)**

**#predicted target variable**

**#convert to factor**

**cv\_predict.rose<-as.factor(cv\_predict.rose)**

**#Confusion matrix**

**confusionMatrix(data=cv\_predict.rose,reference=target)**

**Confusion Matrix and Statistics**

**Reference**

**Prediction 1 2**

**1 20012 0**

**2 0 19988**

**Accuracy : 1**

**95% CI : (0.9999, 1)**

**No Information Rate : 0.5003**

**P-Value [Acc > NIR] : < 2.2e-16**

**Kappa : 1**

**Mcnemar's Test P-Value : NA**

**Sensitivity : 1.0000**

**Specificity : 1.0000**

**Pos Pred Value : 1.0000**

**Neg Pred Value : 1.0000**

**Prevalence : 0.5003**

**Detection Rate : 0.5003**

**Detection Prevalence : 0.5003**

**Balanced Accuracy : 1.0000**

**'Positive' Class : 1**

**Receiver operating characteristics(ROC)-Area under curve(AUC) score and curve**

**set.seed(843)**

**#convert to numeric**

**cv\_predict.rose<-as.numeric(cv\_predict.rose)**

**roc(data=valid.rose[,-c(1,2)],response=target,predictor=cv\_predict.rose,auc=TRUE,plot=TRUE)**

Call:

roc.default(response = target, predictor = cv\_predict.rose, auc = TRUE, plot = TRUE, data = valid.rose[, -c(1, 2)])

Data: cv\_predict.rose in 20012 controls (target 1) < 19988 cases (target 2).

Area under the curve: 1



We can observe that ROSE model is performing well on imbalance data compare to baseline logistic regression.

**LightGBM**

LightGBM is a gradient boosting framework that uses tree based learning algorithms. We are going to use LightGBM model.

Let us build LightGBM model

**Training and validation dataset**

#Convert data frame to matrix

set.seed(5432)

X\_train<-as.matrix(train.data[,-c(1,2)])

y\_train<-as.matrix(train.data$target)

X\_valid<-as.matrix(valid.data[,-c(1,2)])

y\_valid<-as.matrix(valid.data$target)

test\_data<-as.matrix(test\_df[,-c(1)])

#training data

lgb.train <- lgb.Dataset(data=X\_train, label=y\_train)

#Validation data

lgb.valid <- lgb.Dataset(data=X\_valid,label=y\_valid)

**Choosing best hyperparameters**

#Selecting best hyperparameters

set.seed(653)

lgb.grid = list(objective = "binary",

metric = "auc",

boost='gbdt',

max\_depth=-1,

boost\_from\_average='false',

min\_sum\_hessian\_in\_leaf = 12,

feature\_fraction = 0.05,

bagging\_fraction = 0.45,

bagging\_freq = 5,

learning\_rate=0.02,

tree\_learner='serial',

num\_leaves=20,

num\_threads=5,

min\_data\_in\_bin=150,

min\_gain\_to\_split = 30,

min\_data\_in\_leaf = 90,

verbosity=-1,

is\_unbalance = TRUE)

**Training the lgbm model**

set.seed(7663)

lgbm.model <- lgb.train(params = lgb.grid, data = lgb.train, nrounds =10000,eval\_freq =1000,

valids=list(val1=lgb.train,val2=lgb.valid),early\_stopping\_rounds = 5000)

[1]: val1's auc:1 val2's auc:1

[1001]: val1's auc:1 val2's auc:1

[2001]: val1's auc:1 val2's auc:1

[3001]: val1's auc:1 val2's auc:1

[4001]: val1's auc:1 val2's auc:1

[5001]: val1's auc:1 val2's auc:1

**lgbm model performance on test data**

#lgbm model performance on test data

set.seed(6532)

lgbm\_pred\_prob <- predict(lgbm.model,test\_data)

print(lgbm\_pred\_prob)

#Convert to binary output (1 and 0) with threshold 0.5

lgbm\_pred<-ifelse(lgbm\_pred\_prob>0.5,1,0)

print(lgbm\_pred)

**Let us plot the important features**

set.seed(6521)

#feature importance plot

tree\_imp <- lgb.importance(lgbm.model, percentage = TRUE)

lgb.plot.importance(tree\_imp, top\_n = 50, measure = "Frequency", left\_margin = 10)

We tried model with logistic regression,ROSE and lightgbm. But,lightgbm is performing well on imbalanced data compared to other models based on scores of roc\_auc\_score.

**Submission:**

sub\_df<-data.frame(ID\_code=test\_df$ID\_code,lgb\_predict\_prob=lgbm\_pred\_prob,lgb\_predict=lgbm\_pred)

write.csv(sub\_df,'submission.CSV',row.names=F)

head(sub\_df)