STAT515_005 FINAL PROJECT GROUP 7

(CREDIT CARD FRAUDULENT DETECTION)

Vedant Shamling Limbhare, Prathyusha Elipay

2023-05-14

Importing necessary libraries:

```
library(tidyverse)
library(kableExtra)
library(geosphere)
library(Hmisc)
library(reshape2)
library(scales)
library(gridExtra)
library(grid)
library(ggcorrplot)
library(caret)
library(ggplot2)
library(ROSE)
library(pROC)
library(themis)
library(xgboost)
library(caTools)
library(rpart)
library(visNetwork)
library(rpart.plot)
library(randomForest)
library(ROCR)
library(tree)
library(sparkline)
library(knitr)
```

Loading the Credit Card Fraud Data-set:

```
cc_fraud_original <- as.data.frame(read_csv("credit_card_fraud.csv"))

str(cc_fraud_original)
```

```
## 'data.frame': 339607 obs. of 15 variables:
## $ trans_date_trans_time: POSIXct, format: "2019-01-01 00:00:44" "2019-01-01 00:00:51" ...
## $ merchant : chr "Heller, Gutmann and Zieme" "Lind-Buckridge" "Kiehn Inc" "Beier-Hyatt" ...
```

```
## $ category : chr "grocery_pos" "entertainment" "grocery_pos" "shopping_pos" ...
## $ amt : num 107.23 220.11 96.29 7.77 6.85 ...
## $ city : chr "Orient" "Malad City" "Grenada" "High Rolls Mountain Park" ...
## $ state : chr "WA" "ID" "CA" "NM" ...
## $ lat : num 48.9 42.2 41.6 32.9 43 ...
## $ long : num -118 -112 -123 -106 -111 ...
## $ city_pop : num 149 4154 589 899 471 ...
## $ job : chr "Special educational needs teacher" "Nature conservation officer" "Systems analyst"
"Naval architect" ...
## $ dob : Date, format: "1978-06-21" "1962-01-19" ...
## $ trans_num : chr "1f76529f8574734946361c461b024d99" "a1a22d70485983eac12b5b88dad1cf95"
"413636e759663f264aae1819a4d4f231" "8a6293af5ed278dea14448ded2685fea" ...
## $ merch_lat : num 49.2 43.2 41.7 32.9 43.8 ...
## $ merch_long : num -118 -112 -122 -107 -111 ...
## $ is_fraud : num 0 0 0 0 0 0 0 0 0 0 ...
```

```
cat("Any NULL values in the dataset:       ", anyNA(cc_fraud_original),
    "\nAny Duplicated values in the dataset: ",
    ↪  as.logical(anyDuplicated(cc_fraud_original)))
```

```
## Any NULL values in the dataset:        FALSE
## Any Duplicated values in the dataset:  FALSE
```

First 10 rows:

```
head(cc_fraud_original, 10)
```

```
##    trans_date_trans_time                     merchant      category    amt
## 1    2019-01-01 00:00:44   Heller, Gutmann and Zieme   grocery_pos 107.23
## 2    2019-01-01 00:00:51             Lind-Buckridge entertainment 220.11
## 3    2019-01-01 00:07:27                  Kiehn Inc   grocery_pos  96.29
## 4    2019-01-01 00:09:03                Beier-Hyatt  shopping_pos   7.77
## 5    2019-01-01 00:21:32                 Bruen-Yost      misc_pos   6.85
## 6    2019-01-01 00:22:06                  Kunze Inc   grocery_pos  90.22
## 7    2019-01-01 00:22:18   Nitzsche, Kessler and Wol  shopping_pos   4.02
## 8    2019-01-01 00:22:36 Kihn, Abernathy and Douglas  shopping_net   3.66
## 9    2019-01-01 00:31:51         Ledner-Pfannerstill gas_transport 102.13
## 10   2019-01-01 00:34:10               Stracke-Lemke   grocery_pos  83.07
##                         city state     lat      long city_pop
## 1                     Orient    WA 48.8878 -118.2105      149
## 2                 Malad City    ID 42.1808 -112.2620     4154
## 3                    Grenada    CA 41.6125 -122.5258      589
## 4  High Rolls Mountain Park    NM 32.9396 -105.8189      899
## 5                    Freedom    WY 43.0172 -111.0292      471
## 6                    Honokaa    HI 20.0827 -155.4880     4878
## 7                  Valentine    NE 42.8062 -100.6215     4005
## 8                    Westfir    OR 43.7575 -122.4810      597
## 9                   Thompson    UT 38.9999 -109.6150       46
## 10                    Conway    WA 48.3400 -122.3456       85
##                                  job        dob
## 1  Special educational needs teacher 1978-06-21
## 2        Nature conservation officer 1962-01-19
```

```
## 3                    Systems analyst 1945-12-21
## 4                    Naval architect 1967-08-30
## 5          Education officer, museum 1967-08-02
## 6                    Physiotherapist 1966-12-03
## 7                   Network engineer 1945-03-15
## 8              Forensic psychologist 1961-05-19
## 9                 Surveyor, minerals 1987-04-23
## 10 Research officer, political party 1984-09-01
##                           trans_num merch_lat merch_long is_fraud
## 1  1f76529f8574734946361c461b024d99  49.15905  -118.1865        0
## 2  a1a22d70485983eac12b5b88dad1cf95  43.15070  -112.1545        0
## 3  413636e759663f264aae1819a4d4f231  41.65752  -122.2303        0
## 4  8a6293af5ed278dea14448ded2685fea  32.86326  -106.5202        0
## 5  f3c43d336e92a44fc2fb67058d5949e3  43.75373  -111.4549        0
## 6  95826e3caa9e0b905294c6dae985aec1  19.56001  -156.0459        0
## 7  20490f3f0966ce74b4aaba8dc2c4ed52  42.47559  -101.2658        0
## 8  870c92b288a974a2faf1f24b05c27e33  44.27819  -121.8152        0
## 9  47238da5b40d126c8abea40a857c7809  39.80731  -109.3483        0
## 10 9b7a0619dcc5c572dc134f2827ed5a6b  48.68211  -122.7199        0
```

Last 10 rows:

```
tail(cc_fraud_original, 10)
```

```
##        trans_date_trans_time                      merchant       category
## 339598   2020-12-31 23:44:29                 Dickinson Lt  personal_care
## 339599   2020-12-31 23:44:51                    Kuhic LLC   shopping_net
## 339600   2020-12-31 23:46:55               O'Keefe-Wisoky    food_dining
## 339601   2020-12-31 23:47:11            Altenwerth-Kilback           home
## 339602   2020-12-31 23:57:18 Larkin, Stracke and Greenfelde  entertainment
## 339603   2020-12-31 23:57:56                Schmidt-Larkin           home
## 339604   2020-12-31 23:58:04     Pouros, Walker and Spence      kids_pets
## 339605   2020-12-31 23:59:07               Reilly and Sons health_fitness
## 339606   2020-12-31 23:59:15                    Rau-Robel      kids_pets
## 339607   2020-12-31 23:59:24               Breitenberg LLC         travel
##           amt               city state     lat      long city_pop
## 339598 108.16            Tomales    CA 38.2427 -122.9145      337
## 339599   4.27    Mountain Center    CA 33.6401 -116.5567     1661
## 339600   2.65             Arnold    MO 38.4305  -90.3870    35439
## 339601  47.05            Burbank    WA 46.1966 -118.9017     3684
## 339602  46.71 Blairsden-Graeagle    CA 39.8127 -120.6405     1725
## 339603  12.68              Wales    AK 64.7556 -165.6723      145
## 339604  13.02          Greenview    CA 41.5403 -122.9366      308
## 339605  43.77              Luray    MO 40.4931  -91.8912      519
## 339606  86.88            Burbank    WA 46.1966 -118.9017     3684
## 339607   7.99               Mesa    ID 44.6255 -116.4493      129
##                                          job        dob
## 339598                Occupational psychologist 1954-07-05
## 339599                        Therapist, music 1988-09-19
## 339600                  Land/geomatics surveyor 1985-03-31
## 339601                                Musician 1981-11-29
## 339602 Chartered legal executive (England and Wales) 1967-05-27
## 339603                 Administrator, education 1939-11-09
```

```
## 339604                            Call centre manager 1958-09-20
## 339605                                  Town planner 1966-02-13
## 339606                                      Musician 1981-11-29
## 339607                                  Cartographer 1965-12-15
##                                trans_num merch_lat merch_long is_fraud
## 339598 f099c1173b18b2431baefdbd8fa8877c  39.14194 -123.81458        0
## 339599 682dfc75e4ccbb80a2de3d77f6dc8f30  33.53378 -117.30295        0
## 339600 cdb9edafe652ee2abc502a804665d1cb  38.74249  -89.48446        0
## 339601 1fa626c083dfb1b3296fec4c5877f5cb  45.71671 -119.88625        0
## 339602 a7105564935ea3977dc61ff9ced3bf5e  38.96354 -120.45712        0
## 339603 a8310343c189e4a5b6316050d2d6b014  65.62359 -165.18603        0
## 339604 bd7071fd5c9510a5594ee196368ac80e  41.97313 -123.55303        0
## 339605 9b1f753c79894c9f4b71f04581835ada  39.94684  -91.33333        0
## 339606 6c5b7c8add471975aa0fec023b2e8408  46.65834 -119.71505        0
## 339607 14392d723bb7737606b2700ac791b7aa  44.47053 -117.08089        0
```

Data Pre-processing

Creating the lists of values from different columns for categorization:

```
cities <- c("Albuquerque", "American Fork", "Arvada", "Aurora", "Azusa", "Ballwin", "Bay
↪  City",
            "Broomfield", "Burbank","Burlington", "Camden", "Campbell", "Caroll",
            "Colorado Springs", "Colton", "Conway", "Corona", "Daly City", "Downey",
            "Espanola","Eugene", "Fullerton", "Glendale", "Huntington Beach",
            ↪  "Independence",
            "Issaquah", "Kansas City", "Kent", "Kirk", "Kirtland Afb", "La Grande",
            "Laguna Hills", "Lake Oswego", "Laramie", "Littleton", "Los Angeles",
            ↪  "Lowell",
            "Malad City", "Manville", "Matthews", "Meadville", "Meridian", "Mesa",
            ↪  "Moab",
            "Moriarty", "Napa", "Newberg", "Newhall","Norwalk", "Oakland", "Odessa",
            ↪  "Omaha",
            "Owensville", "Palmdale", "Parker", "Parker Dam", "Phoenix", "Pleasant Hill",
            "Portland", "Pueblo", "Ravenna", "Red Cliff", "Red River", "Redford",
            ↪  "Riverton",
            "Rock Springs", "Rocky Mount", "Roseland", "Ruidoso", "Sacramento", "Saint
            ↪  Louis",
            "San Diego", "San Jose", "Santa Monica", "Scotts Mills", "Seattle",
            ↪  "Seligman",
            "Shedd", "Smith River", "Spirit Lake", "Stayton", "Sun City", "Syracuse",
            ↪  "Tekoa",
            "Thompson", "Unionville", "Utica", "Vancouver", "Vinton", "Westerville",
            ↪  "Westfir",
            "Wheaton", "Williamsburg", "Woods Cross")

business_jobs <- c("Accountant", "Advertising account planner", "Airline pilot",
                   "Associate Professor", "Buyer, industrial", "Call centre manager",
                   "Chief Marketing Officer", "Comptroller", "Contractor", "Economist",
                   "Education administrator", "Education officer, museum",
                   "Engineering managers", "Freight forwarder", "Hotel manager",
                   "Human resources officer", "Information systems manager",
                   "Insurance broker", "Investment analyst", "Investment banker,
                   ↪  corporate",
```

```
                  "IT consultant", "Lecturer, higher education", "Local government
                  ↪  officer",
                  "Marketing executive", "Occupational hygienist",
                  "Product/process development scientist", "Production manager",
                  "Public house manager", "Public relations account executive",
                  "Retail merchandiser", "Sales executive, IT", "Surveyor, minerals",
                  "Systems analyst", "Tax inspector", "Tourist information centre
                  ↪  manager")

engineering_jobs <- c("Aeronautical engineer", "Agricultural consultant", "Architect",
                      "Architectural technologist",
                      "Armed forces training and education officer", "Building surveyor",
                      "Chemical engineer", "Civil engineer, contracting",
                      "Civil Service administrator", "Civil Service fast streamer",
                      "Colour technologist", "Development worker, international aid",
                      "Electronics engineer", "Engineer, automotive", "Engineer,
                      ↪  biomedical",
                      "Engineer, building services", "Engineer, civil (consulting)",
                      "Engineer, communications", "Engineer, electronics",
                      "Engineer, maintenance", "Engineer, petroleum", "Engineer,
                      ↪  production",
                      "Engineer, site", "Geologist, engineering", "Geoscientist",
                      "Materials engineer", "Metallurgist", "Naval architect",
                      "Network engineer", "Petroleum engineer",
                      "Planning and development surveyor", "Product designer",
                      "Research scientist (physical sciences)", "Scientist, marine",
                      "Scientist, physiological", "Surveyor, land/geomatics",
                      "Surveyor, mining", "Systems developer", "Water engineer",
                      "Wellsite geologist")

healthcare_jobs <- c("Chiropodist", "Clinical cytogeneticist", "Clinical research
↪  associate",
                     "Counselling psychologist", "Counsellor", "Cytogeneticist",
                     "Exercise physiologist", "Fine artist", "Forensic psychologist",
                     "Health physicist", "Health service manager", "Mental health nurse",
                     "Music therapist", "Nurse, children's", "Nurse, mental health",
                     "Occupational psychologist", "Osteopath", "Pharmacist, hospital",
                     "Physiotherapist", "Scientist, audiological",
                     "Scientist, research (maths)",
                     "Therapist, art", "Therapist, horticultural",
                     "Therapist, music", "Therapist, occupational")

creative_jobs <- c("Glass blower/designer",
                   "Historic buildings inspector/conservation officer",
                   "Journalist, newspaper", "Landscape architect", "Location manager",
                   "Magazine features editor", "Museum education officer",
                   "Museum/gallery exhibitions officer", "Musician", "Set designer",
                   "Television/film/video producer", "Video editor")

legal_jobs <- c("Barrister", "Careers information officer",
                "Chartered legal executive (England and Wales)",
                "Chartered public finance accountant", "Community arts worker",
                "Immigration officer", "Intelligence analyst",
```

```
                "Licensed conveyancer", "Public librarian",
                "Research officer, political party", "Solicitor, Scotland",
                "Special educational needs teacher")
```

Creating a new data-set with categorized variables:

```
cc_fraud <- cc_fraud_original %>%
  reframe(area = ifelse(state %in% c("AK", "CA", "HI", "OR", "WA"), "west",
                        ifelse(state %in% c("MO", "NE"), "midwest", "southwest")),
          city_type = ifelse(city %in% cities, "urban", "rural"),
          city_size = ifelse(city_pop < 10000, "small",
                             ifelse(city_pop < 100000, "medium",
                                    ifelse(city_pop < 1000000, "large", "mega"))),
          merchant_type = ifelse(category %in%
                                   c("gas_transport", "grocery_net", "grocery_pos",
                                     "health_fitness", "home", "kids_pets",
                                     "personal_care"),
                                 "essential", "discretionary"),
          trans_amt = round(amt, 2),
          trans_hour = ifelse(format(trans_date_trans_time, "%H") %in% c(6:11),
          "morning",
                              ifelse(format(trans_date_trans_time, "%H") %in% c(12:18),
                                     "afternoon",
                                     ifelse(format(trans_date_trans_time, "%H") %in%
                                     c(17:22),
                                            "evening", "night"))),
          weekday_type = ifelse(format(trans_date_trans_time, "%A") %in%
                                  c("Saturday", "Sunday"), "week_end",
                                ifelse(format(trans_date_trans_time, "%A") %in%
                                         c("Monday", "Tuesday"), "week_start",
                                         "mid_week")),
          cc_holder_age = as.numeric(round(difftime(trans_date_trans_time, dob,
                                                    units = "days") / 365.25, 0)),
          job_type = ifelse(job %in% business_jobs, "business",
                            ifelse(job %in% engineering_jobs, "engineering",
                                   ifelse(job %in% healthcare_jobs,"healthcare",
                                          ifelse(job %in% creative_jobs, "creative",
                                                 ifelse(job %in% legal_jobs,
                                                        "legal", "education"))))),
          is_fraud = is_fraud)

rm(cities, business_jobs, creative_jobs, legal_jobs, engineering_jobs, healthcare_jobs)
```

Retrieving distances in miles between the Geo-coordinates of merchant and transaction location in the original data-set:

```
for(i in 1:nrow(cc_fraud_original)){
  cc_fraud$dist_merch_trans[i] = round(distm(x = c(cc_fraud_original$long[i],
                                                   cc_fraud_original$lat[i]),
                                             y = c(cc_fraud_original$merch_long[i],
                                                   cc_fraud_original$merch_lat[i]),
                                             fun = distGeo)/1609.34, 2)
```

```
}

rm(i)
```

Converting categorical variables into factors:

```
cc_fraud <- mutate(cc_fraud, area = as.factor(area),
                   city_type = as.factor(city_type),
                   city_size = as.factor(city_size),
                   merchant_type = as.factor(merchant_type),
                   trans_hour = as.factor(trans_hour),
                   weekday_type = as.factor(weekday_type),
                   job_type = as.factor(job_type),
                   is_fraud = as.factor(is_fraud))

cc_fraud <- cc_fraud %>% select(1:9, "dist_merch_trans", "is_fraud")

str(cc_fraud)
```

```
## 'data.frame':    339607 obs. of  11 variables:
##  $ area            : Factor w/ 3 levels "midwest","southwest",..: 3 2 3 2 2 3 1 3 2 3 ...
##  $ city_type       : Factor w/ 2 levels "rural","urban": 1 2 1 1 1 1 1 2 2 2 ...
##  $ city_size       : Factor w/ 4 levels "large","medium",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ merchant_type   : Factor w/ 2 levels "discretionary",..: 2 1 2 1 1 2 1 1 2 2 ...
##  $ trans_amt       : num  107.23 220.11 96.29 7.77 6.85 ...
##  $ trans_hour      : Factor w/ 4 levels "afternoon","evening",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ weekday_type    : Factor w/ 3 levels "mid_week","week_end",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ cc_holder_age   : num  41 57 73 51 51 52 74 58 32 34 ...
##  $ job_type        : Factor w/ 6 levels "business","creative",..: 6 3 1 4 1 5 4 5 1 6 ...
##  $ dist_merch_trans: num  18.8 67.2 15.6 41.1 55.2 ...
##  $ is_fraud        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
cat("Any NULL values in the new dataset:       ", anyNA(cc_fraud),
    "\nAny Duplicated values in the new dataset: ", as.logical(anyDuplicated(cc_fraud)))
```

```
## Any NULL values in the new dataset:        FALSE
## Any Duplicated values in the new dataset:  FALSE
```

```
summary(cc_fraud)
```

```
##         area        city_type       city_size            merchant_type
##  midwest  : 89329   rural:148606   large : 43246   discretionary:151212
##  southwest:109723   urban:191001   medium: 59898   essential    :188395
##  west     :140555                  mega  : 11698
##                                    small :224765
##
##
##    trans_amt            trans_hour         weekday_type    cc_holder_age
##  Min.   :    1.00   afternoon:119838   mid_week  :112098   Min.   :17.0
##  1st Qu.:    9.60   evening  : 69302   week_end  :110062   1st Qu.:34.0
```

```
##  Median :   46.46   morning  : 22139   week_start:117447   Median :46.0
##  Mean   :   70.58   night    :128328                       Mean   :47.8
##  3rd Qu.:   83.35                                         3rd Qu.:58.0
##  Max.   :28948.90                                         Max.   :93.0
##         job_type     dist_merch_trans is_fraud
##  business   :81145   Min.   : 0.13    0:337825
##  creative   :19794   1st Qu.:34.03    1:  1782
##  education  :73973   Median :48.25
##  engineering:85005   Mean   :46.97
##  healthcare :51203   3rd Qu.:60.84
##  legal      :28487   Max.   :94.37
```

First 10 rows in the new data-set:

```
head(cc_fraud, 10)
```

```
##         area city_type city_size merchant_type trans_amt trans_hour
## 1       west     rural     small     essential    107.23      night
## 2  southwest     urban     small discretionary    220.11      night
## 3       west     rural     small     essential     96.29      night
## 4  southwest     rural     small discretionary      7.77      night
## 5  southwest     rural     small discretionary      6.85      night
## 6       west     rural     small     essential     90.22      night
## 7    midwest     rural     small discretionary      4.02      night
## 8       west     urban     small discretionary      3.66      night
## 9  southwest     urban     small     essential    102.13      night
## 10      west     urban     small     essential     83.07      night
##    weekday_type cc_holder_age    job_type dist_merch_trans is_fraud
## 1    week_start            41       legal            18.78        0
## 2    week_start            57   education            67.17        0
## 3    week_start            73    business            15.61        0
## 4    week_start            51 engineering            41.11        0
## 5    week_start            51    business            55.18        0
## 6    week_start            52  healthcare            51.10        0
## 7    week_start            74 engineering            39.99        0
## 8    week_start            58  healthcare            48.92        0
## 9    week_start            32    business            57.50        0
## 10   week_start            34       legal            29.22        0
```

Last 10 rows in the data-set:

```
tail(cc_fraud,10)
```

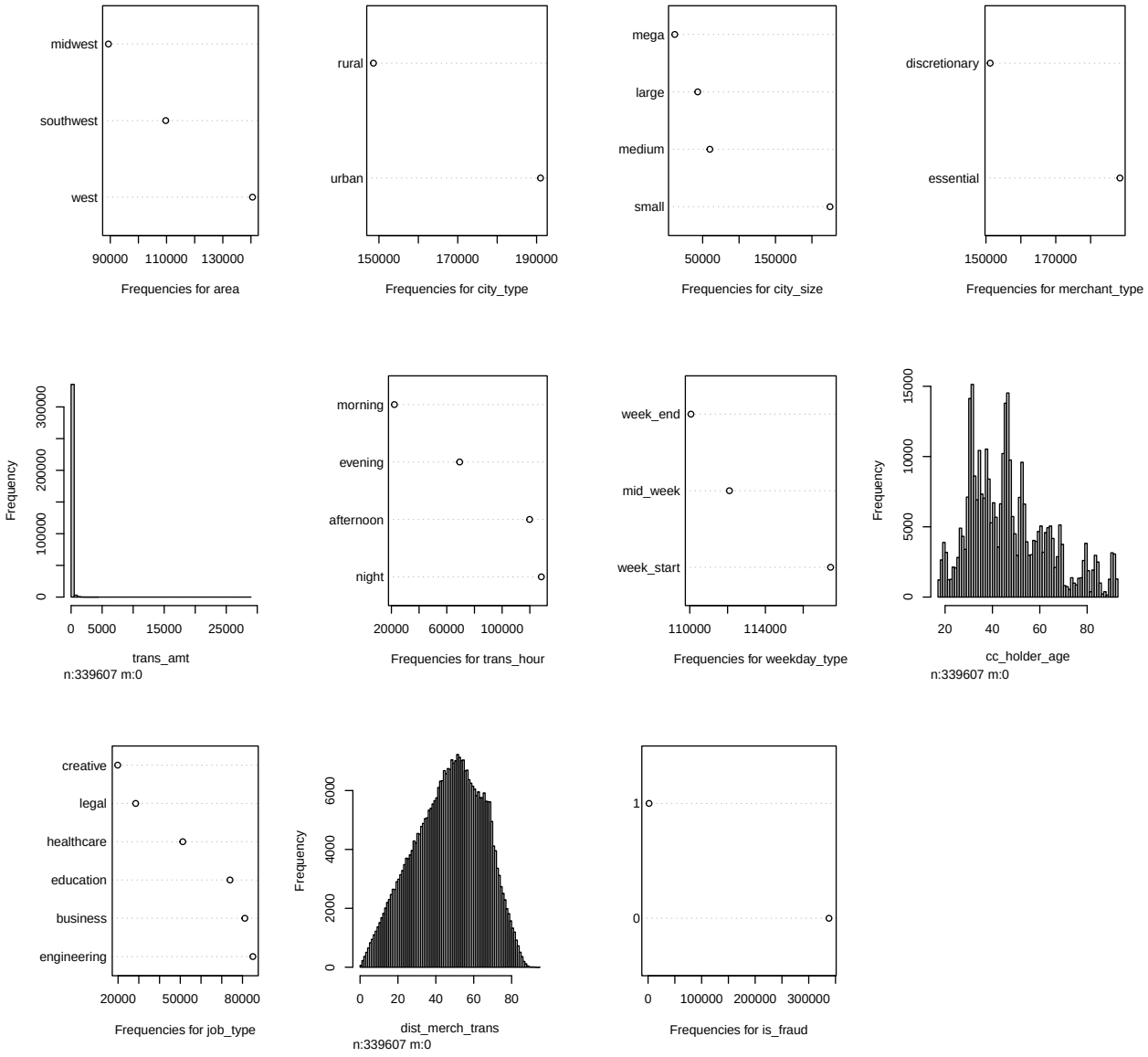
```
##            area city_type city_size merchant_type trans_amt trans_hour
## 339598     west     rural     small     essential    108.16      night
## 339599     west     rural     small discretionary      4.27      night
## 339600  midwest     rural    medium discretionary      2.65      night
## 339601     west     urban     small     essential     47.05      night
## 339602     west     rural     small discretionary     46.71      night
## 339603     west     rural     small     essential     12.68      night
## 339604     west     rural     small     essential     13.02      night
## 339605  midwest     rural     small     essential     43.77      night
```

```
## 339606      west     urban     small     essential     86.88      night
## 339607 southwest     urban     small discretionary      7.99      night
##        weekday_type cc_holder_age   job_type dist_merch_trans is_fraud
## 339598     mid_week            66 healthcare            78.84        0
## 339599     mid_week            32 healthcare            43.66        0
## 339600     mid_week            36  education            53.39        0
## 339601     mid_week            39   creative            57.86        0
## 339602     mid_week            54      legal            59.40        0
## 339603     mid_week            81  education            61.78        0
## 339604     mid_week            62   business            43.67        0
## 339605     mid_week            55  education            47.87        0
## 339606     mid_week            39   creative            50.26        0
## 339607     mid_week            55  education            32.97        0
```

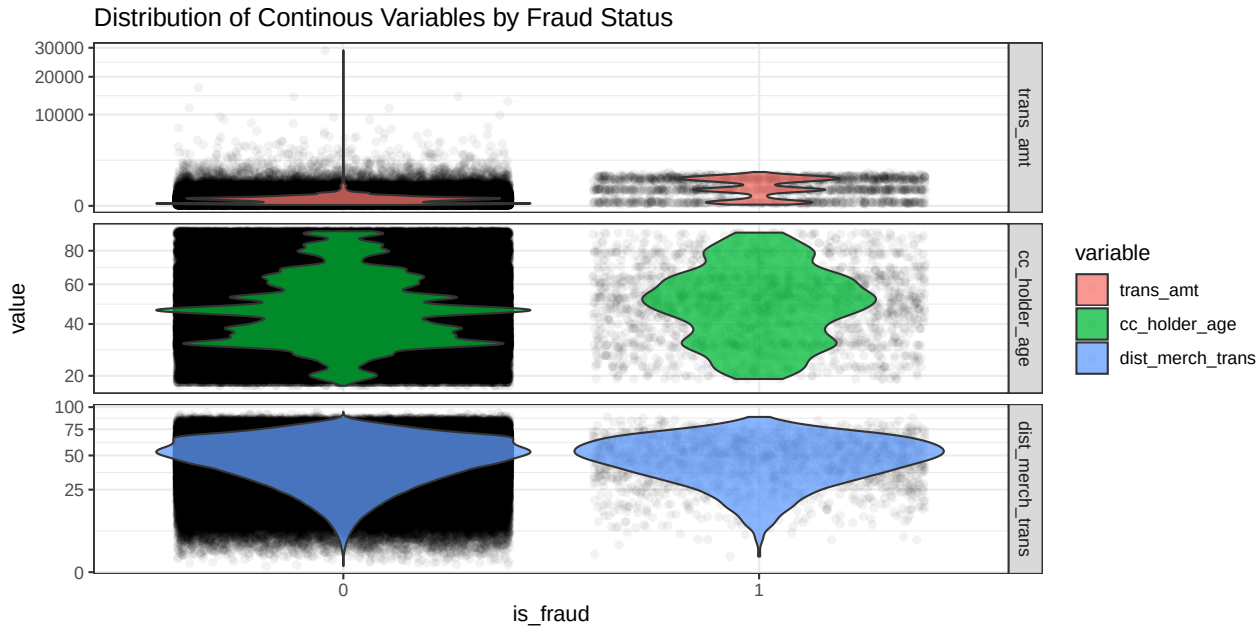
Data Visualization

```
hist.data.frame(cc_fraud, mtitl = "Frequencies of the variables")
```

Frequencies of the variables



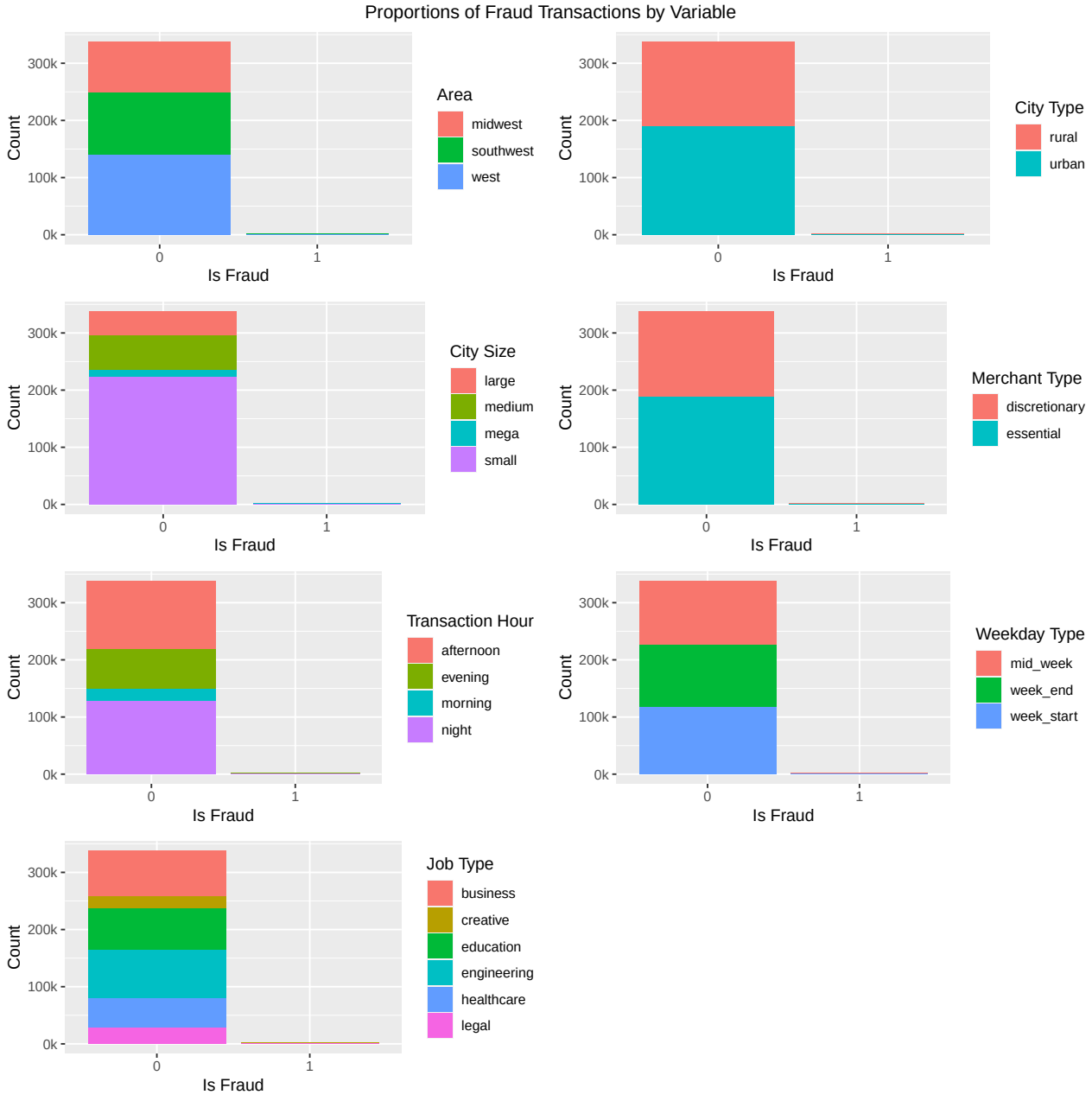

```
tmp <- melt(cc_fraud[, c("trans_amt", "cc_holder_age", "dist_merch_trans", "is_fraud")],
            id.vars="is_fraud")

ggplot(tmp, aes(x = is_fraud, y = value, fill = variable)) +
  geom_jitter(alpha = .05) +
  geom_violin(alpha = .75) +
  facet_grid(variable ~ ., scales = "free_y") +
  scale_y_sqrt() +
  ggtitle("Distribution of Continous Variables by Fraud Status") +
  theme_bw()
```




```
rm(tmp)
```

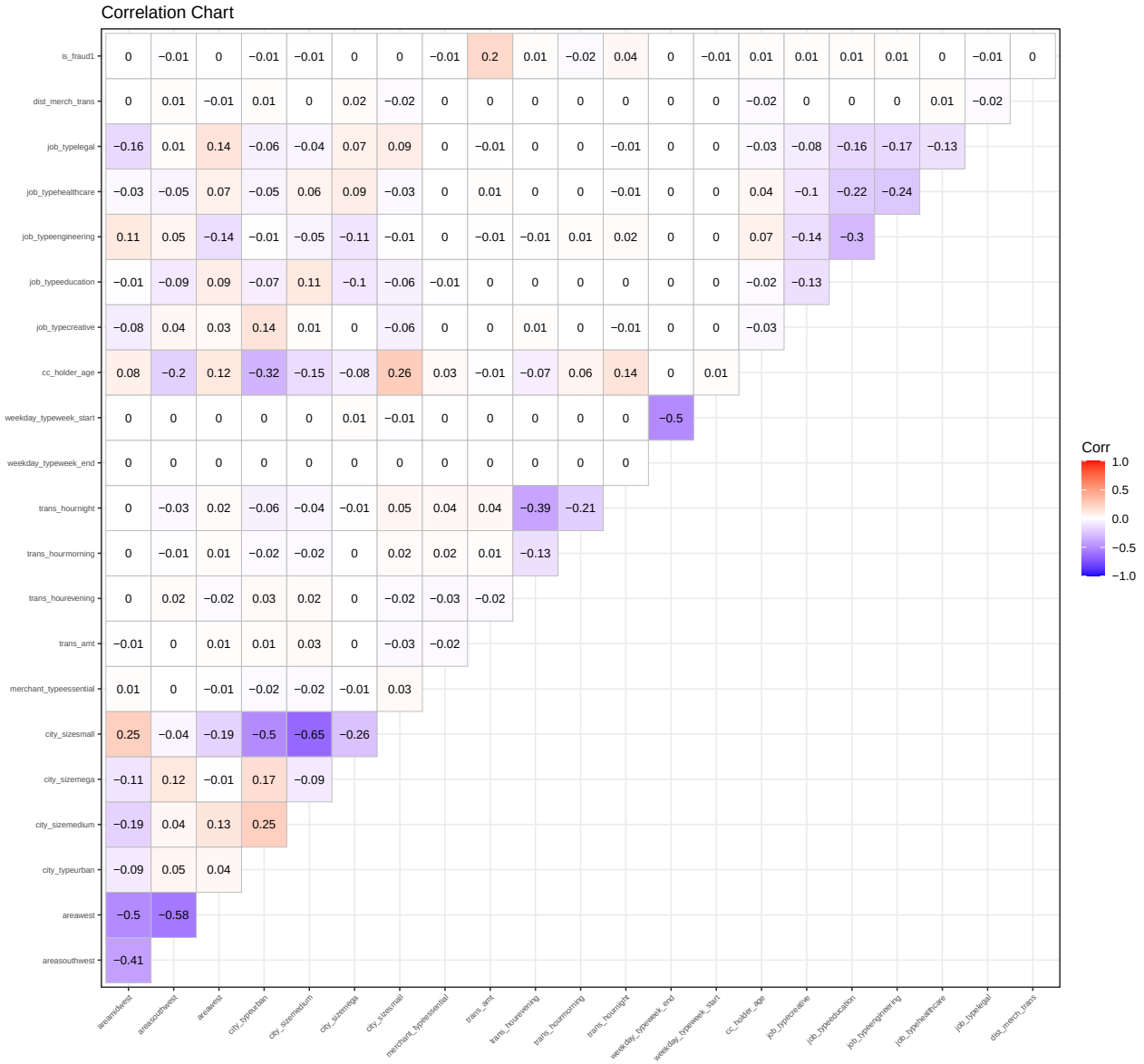
```
p = list()
p[[1]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=area))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "Area")

p[[2]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=city_type))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "City Type")

p[[3]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=city_size))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "City Size")

p[[4]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=merchant_type))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "Merchant Type")

p[[5]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=trans_hour))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "Transaction Hour")

p[[6]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=weekday_type))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "Weekday Type")
```

```
p[[7]] <- ggplot(data=cc_fraud,aes(x=is_fraud))+geom_bar(aes(fill=job_type))+
  scale_y_continuous(breaks = seq(0, 400000, 100000),
                     labels = label_number(scale = 1e-3, suffix = "k")) +
  labs(x = "Is Fraud", y = "Count", fill = "Job Type")

grid.arrange(grobs = p, ncol=2,
             top= textGrob("Proportions of Fraud Transactions by Variable"))
```




```
rm(p)
```

```
model.matrix(~0+., data=cc_fraud) %>%
  cor(use="pairwise.complete.obs") %>%
  ggcorrplot(show.diag=FALSE, type="upper", lab=TRUE, lab_size=3,
             ggtheme = theme_bw,tl.cex = 6, tl.srt = 45,
            title = "Correlation Chart")
```



Attaching the new data-set:

```
attach(cc_fraud)
```

Creating the training and testing subsets:

```
set.seed(111)
train_index = sample(c(TRUE,FALSE),size=nrow(cc_fraud),prob=c(0.8,0.2),replace=TRUE)
ccf.train = cc_fraud[train_index,]
ccf.test = cc_fraud[!train_index,]
is_fraud.test = is_fraud[!train_index]
rm(train_index)

summary(ccf.test)
```

```
##         area       city_type      city_size           merchant_type
##  midwest  :17916   rural:29834   large : 8735   discretionary:30205
##  southwest:22029   urban:38320   medium:12089   essential    :37949
##  west     :28209                 mega  : 2306
##                                  small :45024
##
##
##    trans_amt            trans_hour        weekday_type   cc_holder_age
##  Min.   :    1.00   afternoon:23980   mid_week  :22417   Min.   :17.0
##  1st Qu.:    9.67   evening  :13860   week_end  :22109   1st Qu.:34.0
##  Median :   46.59   morning  : 4535   week_start:23628   Median :46.0
##  Mean   :   70.09   night    :25779                      Mean   :47.8
##  3rd Qu.:   83.73                                       3rd Qu.:58.0
##  Max.   :16837.08                                       Max.   :93.0
##         job_type     dist_merch_trans is_fraud
##  business   :16259   Min.   : 0.21    0:67784
##  creative   : 3990   1st Qu.:34.08    1:  370
##  education  :14990   Median :48.27
##  engineering:16927   Mean   :46.99
##  healthcare :10274   3rd Qu.:60.95
##  legal      : 5714   Max.   :92.71
```

```
summary(ccf.train)
```

```
##         area         city_type       city_size            merchant_type
##  midwest  : 71413   rural:118772   large : 34511   discretionary:121007
##  southwest: 87694   urban:152681   medium: 47809   essential    :150446
##  west     :112346                  mega  :  9392
##                                    small :179741
##
##
##    trans_amt            trans_hour         weekday_type   cc_holder_age
##  Min.   :    1.00   afternoon: 95858   mid_week  :89681   Min.   :17.0
##  1st Qu.:    9.59   evening  : 55442   week_end  :87953   1st Qu.:34.0
##  Median :   46.42   morning  : 17604   week_start:93819   Median :46.0
##  Mean   :   70.70   night    :102549                      Mean   :47.8
##  3rd Qu.:   83.25                                        3rd Qu.:58.0
##  Max.   :28948.90                                        Max.   :93.0
##         job_type     dist_merch_trans is_fraud
##  business   :64886   Min.   : 0.13    0:270041
##  creative   :15804   1st Qu.:34.01    1:  1412
##  education  :58983   Median :48.24
```

```
##  engineering:68078   Mean   :46.96
##  healthcare :40929   3rd Qu.:60.82
##  legal      :22773   Max.   :94.37
```

Implementations by Prathyusha Elipay

Logistic Regression Models with sampling methods

Normal Sampling:

```
log.m1 = glm(is_fraud~.,data=ccf.train,family=binomial)

log.m1.probs = predict(log.m1, ccf.test, type="response")
log.m1.preds = ifelse(log.m1.probs > 0.5, 1, 0)
log.m1.error = round(mean(is_fraud.test != log.m1.preds), 4)
log.m1.roc = roc(is_fraud.test, log.m1.probs)
log.m1.auc = round(log.m1.roc$auc, 4)
log.m1.cm = confusionMatrix(as.factor(log.m1.preds), reference = is_fraud.test,
                            positive = '1')
log.m1.accuracy = round(log.m1.cm$overall["Accuracy"],4)
log.m1.sensitivity = round(log.m1.cm$byClass["Sensitivity"], 4)
log.m1.specificity = round(log.m1.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m1.auc,
    "\nAccuracy:    ", log.m1.accuracy,
    "\nSensitivity: ", log.m1.sensitivity,
    "\nSpecificity: ", log.m1.specificity)
```

```
## AUC:          0.8539
## Accuracy:     0.9944
## Sensitivity:  0
## Specificity:  0.9998
```

```
summary(log.m1)
```

```
##
## Call:
## glm(formula = is_fraud ~ ., family = binomial, data = ccf.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.1203  -0.0922  -0.0316   4.1272
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -7.789e+00  2.258e-01 -34.497  < 2e-16 ***
## areasouthwest          6.463e-02  7.731e-02   0.836  0.40318
## areawest               1.538e-01  7.356e-02   2.091  0.03650 *
## city_typeurban        -1.212e-01  6.990e-02  -1.734  0.08300 .
## city_sizemedium       -6.183e-01  1.069e-01  -5.786 7.19e-09 ***
## city_sizemega         -3.610e-01  1.958e-01  -1.843  0.06530 .
```

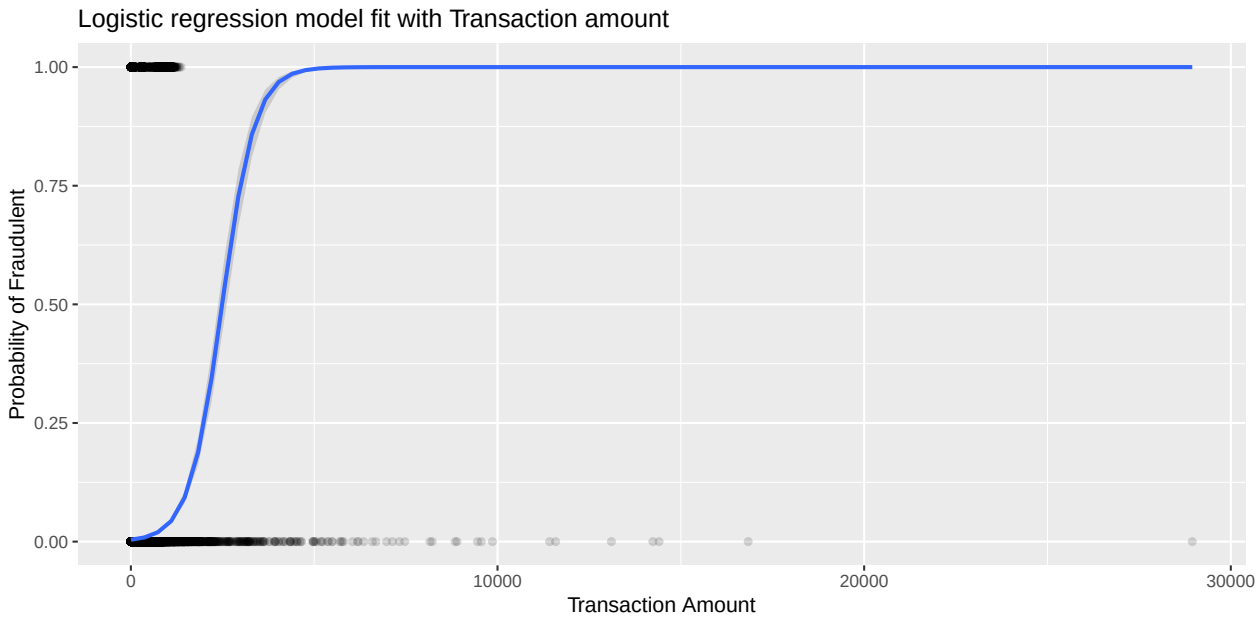
```
## city_sizesmall         -2.282e-01  9.203e-02  -2.479  0.01316 *
## merchant_typeessential -2.137e-01  5.765e-02  -3.706  0.00021 ***
## trans_amt               2.372e-03  6.512e-05  36.430  < 2e-16 ***
## trans_hourevening       2.519e+00  1.435e-01  17.555  < 2e-16 ***
## trans_hourmorning       1.075e-01  3.277e-01   0.328  0.74287
## trans_hournight         2.753e+00  1.390e-01  19.806  < 2e-16 ***
## weekday_typeweek_end   -1.747e-01  6.680e-02  -2.616  0.00890 **
## weekday_typeweek_start -2.943e-01  6.782e-02  -4.339 1.43e-05 ***
## cc_holder_age           4.030e-03  1.785e-03   2.258  0.02394 *
## job_typecreative        5.973e-01  1.193e-01   5.006 5.54e-07 ***
## job_typeeducation       4.011e-01  8.761e-02   4.579 4.67e-06 ***
## job_typeengineering     4.125e-01  8.389e-02   4.917 8.77e-07 ***
## job_typehealthcare      1.132e-01  1.021e-01   1.109  0.26754
## job_typelegal          -1.677e-01  1.392e-01  -1.205  0.22829
## dist_merch_trans        1.389e-03  1.543e-03   0.900  0.36794
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 17667  on 271452  degrees of freedom
## Residual deviance: 14979  on 271432  degrees of freedom
## AIC: 15021
##
## Number of Fisher Scoring iterations: 10
```

```
log.m1.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 67770   370
##          1    14     0
##
##                Accuracy : 0.9944
##                  95% CI : (0.9938, 0.9949)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 0.7762
##
##                   Kappa : -4e-04
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9997935
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9945700
##              Prevalence : 0.0054289
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0002054
##       Balanced Accuracy : 0.4998967
##
```

```
##        'Positive' Class : 1
##
```

```
p <- cc_fraud %>%
  mutate(prob = ifelse(is_fraud == "1", 1, 0)) %>%
  ggplot(aes(trans_amt, prob)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit with Transaction amount") +
  xlab("Transaction Amount") +
  ylab("Probability of Fraudulent")

p
```




```
rm(p)
```

```
ccf.train.over <- ovun.sample(is_fraud~., data = ccf.train, method = "over",
                              N=2*nrow(subset(ccf.train, is_fraud == 0)), seed =
                              ↪  345)$data

ccf.train.under <- ovun.sample(is_fraud~., data = ccf.train, method = "under",
                              N=2*nrow(subset(ccf.train, is_fraud == 1)), seed =
                              ↪  345)$data

ccf.train.mixed <- ovun.sample(is_fraud ~ ., data = ccf.train, method = "both", p=0.5,
                                 N=nrow(ccf.train), seed = 345)$data

ccf.train.rose <- ROSE(is_fraud ~ ., data = ccf.train, seed = 345)$data

cat("Over Sampling  (0, 1):  ", table(ccf.train.over$is_fraud),
```

```
    "\nUnder Sampling (0, 1):  ", table(ccf.train.under$is_fraud),
    "\nMixed Sampling (0, 1):  ", table(ccf.train.mixed$is_fraud),
    "\nROSE Sampling  (0, 1):  ", table(ccf.train.rose$is_fraud))
```

Creating new subsets with sampling methods from ROSE library:

```
## Over Sampling  (0, 1):   270041 270041
## Under Sampling (0, 1):   1412 1412
## Mixed Sampling (0, 1):   135679 135774
## ROSE Sampling  (0, 1):   135679 135774
```

Over Sampling:

```
log.m2 = glm(is_fraud~.,data=ccf.train.over, family="binomial")

log.m2.probs = predict(log.m2, ccf.test, type="response")
log.m2.preds = ifelse(log.m2.probs > 0.5, 1, 0)
log.m2.error = round(mean(is_fraud.test != log.m2.preds), 4)
log.m2.roc = roc(is_fraud.test, log.m1.probs)
log.m2.auc = round(log.m2.roc$auc, 4)
log.m2.cm = confusionMatrix(as.factor(log.m2.preds), reference = is_fraud.test,
                            positive = '1')
log.m2.accuracy = round(log.m2.cm$overall["Accuracy"],4)
log.m2.sensitivity = round(log.m2.cm$byClass["Sensitivity"], 4)
log.m2.specificity = round(log.m2.cm$byClass["Specificity"], 4)
log.m2.kappa = round(log.m2.cm$byClass["Kappa"], 4)

cat("AUC:         ", log.m2.auc,
    "\nAccuracy:    ", log.m2.accuracy,
    "\nSensitivity: ", log.m2.sensitivity,
    "\nSpecificity: ", log.m2.specificity)
```

```
## AUC:          0.8539
## Accuracy:     0.8792
## Sensitivity:  0.7081
## Specificity:  0.8802
```

Under Sampling:

```
log.m3 = glm(is_fraud~.,data=ccf.train.under, family="binomial")

log.m3.probs = predict(log.m3, ccf.test, type="response")
log.m3.preds = ifelse(log.m3.probs > 0.5, 1, 0)
log.m3.error = round(mean(is_fraud.test != log.m3.preds), 4)
log.m3.roc = roc(is_fraud.test, log.m3.probs)
log.m3.auc = round(log.m3.roc$auc, 4)
log.m3.cm = confusionMatrix(as.factor(log.m3.preds), reference = is_fraud.test,
                            positive = '1')
log.m3.accuracy = round(log.m3.cm$overall["Accuracy"],4)
log.m3.sensitivity = round(log.m3.cm$byClass["Sensitivity"], 4)
log.m3.specificity = round(log.m3.cm$byClass["Specificity"], 4)
```

```
cat("AUC:         ", log.m3.auc,
    "\nAccuracy:    ", log.m3.accuracy,
    "\nSensitivity: ", log.m3.sensitivity,
    "\nSpecificity: ", log.m3.specificity)
```

```
## AUC:          0.8981
## Accuracy:     0.8717
## Sensitivity:  0.7135
## Specificity:  0.8725
```

Mixed Sampling:

```
log.m4 = glm(is_fraud~.,data=ccf.train.mixed, family="binomial")

log.m4.probs = predict(log.m4, ccf.test, type="response")
log.m4.preds = ifelse(log.m4.probs > 0.5, 1, 0)
log.m4.error = round(mean(is_fraud.test != log.m4.preds), 4)
log.m4.roc = roc(is_fraud.test, log.m4.probs)
log.m4.auc = round(log.m4.roc$auc, 4)
log.m4.cm = confusionMatrix(as.factor(log.m4.preds), reference = is_fraud.test)
log.m4.accuracy = round(log.m4.cm$overall["Accuracy"],4)
log.m4.sensitivity = round(log.m4.cm$byClass["Sensitivity"], 4)
log.m4.specificity = round(log.m4.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m4.auc,
    "\nAccuracy:    ", log.m4.accuracy,
    "\nSensitivity: ", log.m4.sensitivity,
    "\nSpecificity: ", log.m4.specificity)
```

```
## AUC:          0.9004
## Accuracy:     0.8784
## Sensitivity:  0.8793
## Specificity:  0.7081
```

ROSE Sampling:

```
log.m5 = glm(is_fraud~.,data=ccf.train.rose, family="binomial")

log.m5.probs = predict(log.m5, ccf.test, type="response")
log.m5.preds = ifelse(log.m5.probs > 0.5, 1, 0)
log.m5.error = round(mean(is_fraud.test != log.m5.preds), 4)
log.m5.roc = roc(is_fraud.test, log.m5.probs)
log.m5.auc = round(log.m5.roc$auc, 4)
log.m5.cm = confusionMatrix(as.factor(log.m5.preds), reference = is_fraud.test,
                            positive = '1')
log.m5.accuracy = round(log.m5.cm$overall["Accuracy"],4)
log.m5.sensitivity = round(log.m5.cm$byClass["Sensitivity"], 4)
log.m5.specificity = round(log.m5.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m5.auc,
```

```
    "\nAccuracy:    ", log.m5.accuracy,
    "\nSensitivity: ", log.m5.sensitivity,
    "\nSpecificity: ", log.m5.specificity)
```

```
## AUC:          0.9024
## Accuracy:     0.8651
## Sensitivity:  0.7162
## Specificity:  0.8659
```

```
cat("Predictions of Logistic regression with different sampling methods:")
```

Sampled Models comparison

```
## Predictions of Logistic regression with different sampling methods:
```

```
list(
  Normal_Sampling = round(table(is_fraud.test, log.m1.preds)/nrow(ccf.test),4),
  Over_Sampling = round(table(is_fraud.test, log.m2.preds)/nrow(ccf.test),4),
  Under_Sampling = round(table(is_fraud.test, log.m3.preds)/nrow(ccf.test),4),
  Mixed_Sampling = round(table(is_fraud.test, log.m4.preds)/nrow(ccf.test),4),
  ROSE_Sampling = round(table(is_fraud.test, log.m5.preds)/nrow(ccf.test),4))
```

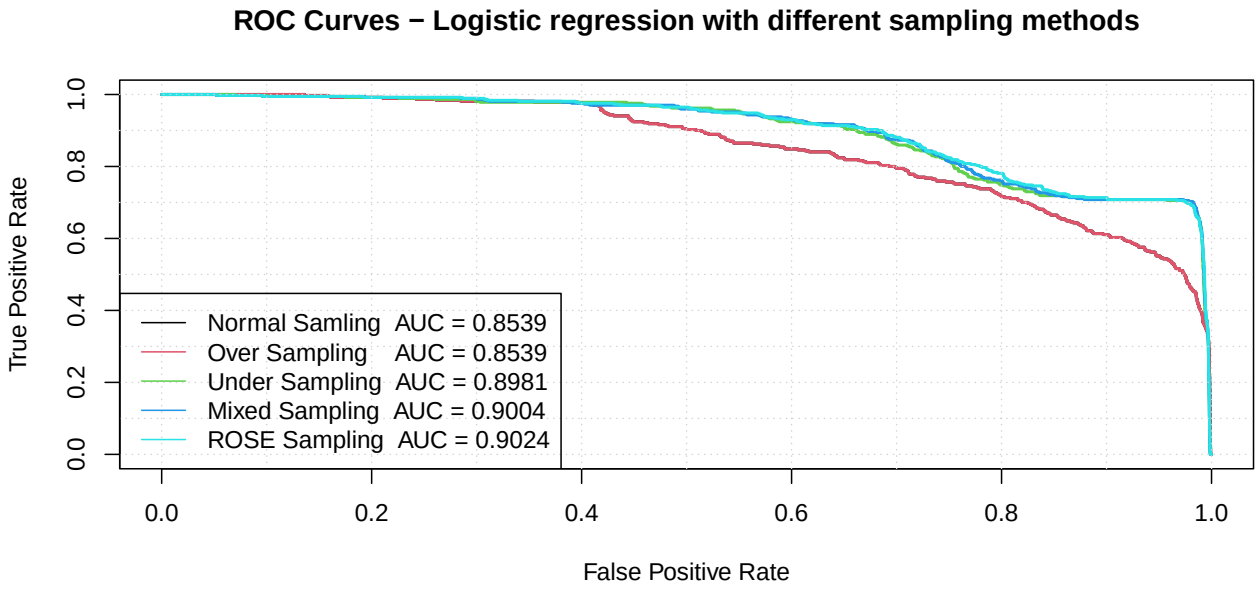
```
## $Normal_Sampling
##              log.m1.preds
## is_fraud.test      0      1
##             0 0.9944 0.0002
##             1 0.0054 0.0000
##
## $Over_Sampling
##              log.m2.preds
## is_fraud.test      0      1
##             0 0.8754 0.1192
##             1 0.0016 0.0038
##
## $Under_Sampling
##              log.m3.preds
## is_fraud.test      0      1
##             0 0.8678 0.1268
##             1 0.0016 0.0039
##
## $Mixed_Sampling
##              log.m4.preds
## is_fraud.test      0      1
##             0 0.8745 0.1200
##             1 0.0016 0.0038
##
## $ROSE_Sampling
##              log.m5.preds
## is_fraud.test      0      1
```

```
##             0 0.8612 0.1333
##             1 0.0015 0.0039
```

```
cat("Mean errors",
    "\nNormal Sampling: ", log.m1.error,
    "\nOver Sampling:   ", log.m2.error,
    "\nUnder Sampling:  ", log.m3.error,
    "\nMixed Sampling:  ", log.m4.error,
    "\nROSE Sampling:   ", log.m5.error)
```

```
## Mean errors
## Normal Sampling:  0.0056
## Over Sampling:    0.1208
## Under Sampling:   0.1283
## Mixed Sampling:   0.1216
## ROSE Sampling:    0.1349
```

```
model_labels <- c("Normal Samling","Over Sampling  " ,"Under Sampling", "Mixed Sampling",
↪
                  "ROSE Sampling")

plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Logistic regression with different sampling methods")

legend_labels <- c()
auc_values <- c()

for (i in 1:5) {
  roc_obj <- eval(parse(text = paste0("log.m", i, ".roc")))
  lines(roc_obj, col = i)
  auc_values[i] <- roc_obj$auc
  legend_labels <- c(legend_labels, paste(model_labels[i], " AUC =", round(auc_values[i],
↪  4)))
}

abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = 1:5, lwd = 1)
```




```
rm(auc_values, roc_obj, i, model_labels,legend_labels)
```

Logistic Models with Cross Validation and different sampling methods

Defining train-control for Cross Validation with 10 folds using different sampling methods:

```
ctrlspecs <- trainControl(method = "cv", number = 10, selectionFunction = "best",
                          summaryFunction = twoClassSummary, classProbs = TRUE)
```

Normal Sampling:

```
log.m6 = train(make.names(is_fraud)~. , data=ccf.train, method="glm", family=binomial,
               trControl=ctrlspecs, metric = c("ROC"))

log.m6.probs = predict(log.m6, ccf.test, type = "prob")[, 2]
log.m6.preds = ifelse(log.m6.probs > 0.5, 1, 0)
log.m6.error = round(mean(is_fraud.test != log.m6.preds), 4)
log.m6.roc = roc(is_fraud.test, log.m6.probs)
log.m6.auc = round(log.m6.roc$auc, 4)
log.m6.cm = confusionMatrix(as.factor(log.m6.preds), reference = is_fraud.test,
                            positive = '1')
log.m6.accuracy = round(log.m6.cm$overall["Accuracy"],4)
log.m6.sensitivity = round(log.m6.cm$byClass["Sensitivity"], 4)
log.m6.specificity = round(log.m6.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m6.auc,
    "\nAccuracy:    ", log.m6.accuracy,
    "\nSensitivity: ", log.m6.sensitivity,
    "\nSpecificity: ", log.m6.specificity)
```

```
## AUC:          0.8539
## Accuracy:     0.9944
```

```
## Sensitivity:  0
## Specificity:  0.9998
```

Up Sampling

```
ctrlspecs$sampling <- "up"

log.m7 = train(make.names(is_fraud)~. , data=ccf.train, method="glm", family=binomial,
               trControl=ctrlspecs, metric = c("ROC"))

log.m7.probs = predict(log.m7, ccf.test, type = "prob")[, 2]
log.m7.preds = ifelse(log.m7.probs > 0.5, 1, 0)
log.m7.error = round(mean(is_fraud.test != log.m7.preds), 4)
log.m7.roc = roc(is_fraud.test, log.m7.probs)
log.m7.auc = round(log.m7.roc$auc, 4)
log.m7.cm = confusionMatrix(as.factor(log.m7.preds), reference = is_fraud.test,
                            positive = '1')
log.m7.accuracy = round(log.m7.cm$overall["Accuracy"],4)
log.m7.sensitivity = round(log.m7.cm$byClass["Sensitivity"], 4)
log.m7.specificity = round(log.m7.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m7.auc,
    "\nAccuracy:    ", log.m7.accuracy,
    "\nSensitivity: ", log.m7.sensitivity,
    "\nSpecificity: ", log.m7.specificity)
```

```
## AUC:          0.9001
## Accuracy:     0.8806
## Sensitivity:  0.7081
## Specificity:  0.8816
```

Down Sampling

```
ctrlspecs$sampling <- "down"

log.m8 = train(make.names(is_fraud)~. , data=ccf.train, method="glm", family=binomial,
               trControl=ctrlspecs, metric = c("ROC"))

log.m8.probs = predict(log.m8, ccf.test, type = "prob")[, 2]
log.m8.preds = ifelse(log.m8.probs > 0.5, 1, 0)
log.m8.error = round(mean(is_fraud.test != log.m8.preds), 4)
log.m8.roc = roc(is_fraud.test, log.m8.probs)
log.m8.auc = round(log.m8.roc$auc, 4)
log.m8.cm = confusionMatrix(as.factor(log.m8.preds), reference = is_fraud.test,
                            positive = '1')
log.m8.accuracy = round(log.m8.cm$overall["Accuracy"],4)
log.m8.sensitivity = round(log.m8.cm$byClass["Sensitivity"], 4)
log.m8.specificity = round(log.m8.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m8.auc,
    "\nAccuracy:    ", log.m8.accuracy,
    "\nSensitivity: ", log.m8.sensitivity,
    "\nSpecificity: ", log.m8.specificity)
```

```
## AUC:          0.8977
## Accuracy:     0.8684
## Sensitivity:  0.7108
## Specificity:  0.8692
```

SMOTE Sampling:

```
ctrlspecs$sampling <- "smote"

log.m9 = train(make.names(is_fraud)~. , data=ccf.train, method="glm", family=binomial,
               trControl=ctrlspecs, metric = c("ROC"))

log.m9.probs = predict(log.m9, ccf.test, type = "prob")[, 2]
log.m9.preds = ifelse(log.m9.probs > 0.5, 1, 0)
log.m9.error = round(mean(is_fraud.test != log.m9.preds), 4)
log.m9.roc = roc(is_fraud.test, log.m9.probs)
log.m9.auc = round(log.m9.roc$auc, 4)
log.m9.cm = confusionMatrix(as.factor(log.m9.preds), reference = is_fraud.test,
                            positive = '1')
log.m9.accuracy = round(log.m9.cm$overall["Accuracy"],4)
log.m9.sensitivity = round(log.m9.cm$byClass["Sensitivity"], 4)
log.m9.specificity = round(log.m9.cm$byClass["Specificity"], 4)

cat("AUC:         ", log.m9.auc,
    "\nAccuracy:    ", log.m9.accuracy,
    "\nSensitivity: ", log.m9.sensitivity,
    "\nSpecificity: ", log.m9.specificity)
```

```
## AUC:          0.9021
## Accuracy:     0.8653
## Sensitivity:  0.7189
## Specificity:  0.8661
```

ROSE Sampling:

```
ctrlspecs$sampling <- "rose"

log.m10 = train(make.names(is_fraud)~. , data=ccf.train, method="glm", family=binomial,
               trControl=ctrlspecs, metric = c("ROC"))

log.m10.probs = predict(log.m10, ccf.test, type = "prob")[, 2]
log.m10.preds = ifelse(log.m10.probs > 0.5, 1, 0)
log.m10.error = round(mean(is_fraud.test != log.m10.preds), 4)
log.m10.roc = roc(is_fraud.test, log.m10.probs)
log.m10.auc = round(log.m10.roc$auc, 4)
log.m10.cm = confusionMatrix(as.factor(log.m10.preds), reference = is_fraud.test,
                             positive = '1')
log.m10.accuracy = round(log.m10.cm$overall["Accuracy"],4)
log.m10.sensitivity = round(log.m10.cm$byClass["Sensitivity"], 4)
log.m10.specificity = round(log.m10.cm$byClass["Specificity"], 4)

rm(ctrlspecs)
```

```
cat("AUC:         ", log.m10.auc,
    "\nAccuracy:    ", log.m10.accuracy,
    "\nSensitivity: ", log.m10.sensitivity,
    "\nSpecificity: ", log.m10.specificity)
```

```
## AUC:          0.8985
## Accuracy:     0.9109
## Sensitivity:  0.7108
## Specificity:  0.912
```

```
cat("Predictions of Logistic regression using Cross Validation(k=10) with different
↪  sampling methods:")
```

Cross Validation Models comparison

```
## Predictions of Logistic regression using Cross Validation(k=10) with different sampling methods:
```

```
list(
  Normal_Sampling = round(table(is_fraud.test, log.m6.preds)/nrow(ccf.test),4),
  Up_Sampling = round(table(is_fraud.test, log.m7.preds)/nrow(ccf.test),4),
  Down_Sampling = round(table(is_fraud.test, log.m8.preds)/nrow(ccf.test),4),
  SMOTE_Sampling = round(table(is_fraud.test, log.m9.preds)/nrow(ccf.test),4),
  ROSE_Sampling = round(table(is_fraud.test, log.m10.preds)/nrow(ccf.test),4)
)
```

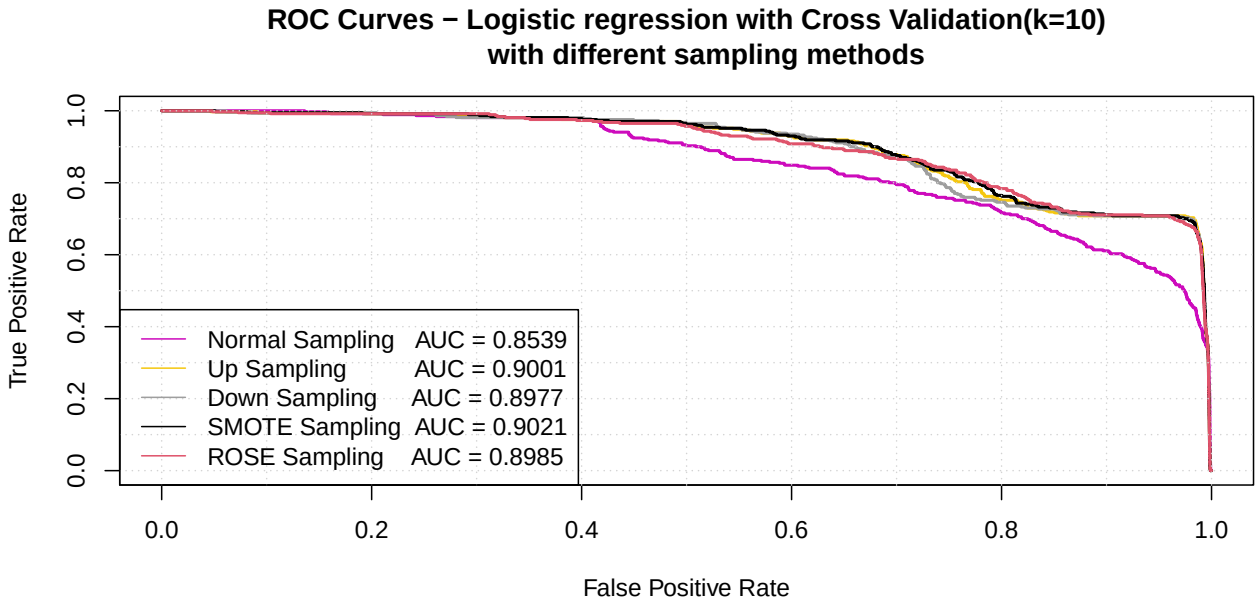
```
## $Normal_Sampling
##              log.m6.preds
## is_fraud.test      0      1
##             0 0.9944 0.0002
##             1 0.0054 0.0000
##
## $Up_Sampling
##              log.m7.preds
## is_fraud.test      0      1
##             0 0.8768 0.1178
##             1 0.0016 0.0038
##
## $Down_Sampling
##              log.m8.preds
## is_fraud.test      0      1
##             0 0.8645 0.1301
##             1 0.0016 0.0039
##
## $SMOTE_Sampling
##              log.m9.preds
## is_fraud.test      0      1
##             0 0.8614 0.1332
##             1 0.0015 0.0039
```

```
##
## $ROSE_Sampling
##              log.m10.preds
## is_fraud.test      0      1
##             0 0.9071 0.0875
##             1 0.0016 0.0039
```

```
cat("Mean errors",
    "\nNormal Sampling: ", log.m6.error,
    "\nUp Sampling:     ", log.m7.error,
    "\nDown Sampling:   ", log.m8.error,
    "\nSMOTE Sampling:  ", log.m9.error,
    "\nROSE Sampling:   ", log.m10.error)
```

```
## Mean errors
## Normal Sampling:  0.0056
## Up Sampling:      0.1194
## Down Sampling:    0.1316
## SMOTE Sampling:   0.1347
## ROSE Sampling:    0.0891
```

```
model_labels <- c("Normal Sampling ", "Up Sampling        " ,"Down Sampling   ", "SMOTE
↪  Sampling", "ROSE Sampling  ")

plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Logistic regression with Cross Validation(k=10)
     with different sampling methods")

legend_labels <- c()
auc_values <- c()

index <- 1
for (i in 6:10) {
  roc_obj <- eval(parse(text = paste0("log.m", i, ".roc")))
  lines(roc_obj, col = i)
  auc_values[i] <- roc_obj$auc
  legend_labels <- c(legend_labels, paste(model_labels[index], " AUC =",
                                          round(auc_values[i], 4)))
  index = index+1
}

abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = 6:10, lwd = 1)
```




```
rm(auc_values, roc_obj, i, model_labels, legend_labels, index)
```

XG Boost Model with Logistic Objective

```
xgb.train <- ccf.train
xgb.train$is_fraud <- as.numeric(xgb.train$is_fraud)-1
xgb.train <- model.matrix(~.-1, data = xgb.train,
                                contrasts.arg = lapply(xgb.train[, sapply(xgb.train,
                                                                          is.factor)],
                                                       contrasts, contrasts=FALSE))
xgb.train.x <- xgb.train[, -28]
xgb.train.y <- xgb.train[, 28]

xgb.test <- ccf.test
xgb.test$is_fraud <- as.numeric(xgb.test$is_fraud)-1
xgb.test <- model.matrix(~ . -1, data = xgb.test,
                                contrasts.arg = lapply(xgb.test[, sapply(xgb.test,
                                                                         is.factor)],
                                                       contrasts, contrasts=FALSE))

xgb.test.x <- xgb.test[, -28]
```

Creating XG Boost testing and training sets: XG Boost with logistic objective:

```
xgb.m1 <- xgboost(data = as.matrix(xgb.train.x), label = xgb.train.y,
                  objective = "binary:logistic", subsample = 0.8, eval_metric = "auc",
                  nrounds = 100, verbose = 0)

xgb.m1.probs = predict(xgb.m1, as.matrix(xgb.test.x))
xgb.m1.preds = ifelse(xgb.m1.probs > 0.5, 1, 0)
```

```
xgb.m1.error = round(mean(is_fraud.test != xgb.m1.preds), 4)
xgb.m1.roc = roc(is_fraud.test, xgb.m1.probs)
xgb.m1.auc = round(xgb.m1.roc$auc, 4)
xgb.m1.cm = confusionMatrix(as.factor(xgb.m1.preds), reference = is_fraud.test,
                            positive = '1')
xgb.m1.accuracy = round(xgb.m1.cm$overall["Accuracy"],4)
xgb.m1.sensitivity = round(xgb.m1.cm$byClass["Sensitivity"], 4)
xgb.m1.specificity = round(xgb.m1.cm$byClass["Specificity"], 4)

cat("AUC:         ", xgb.m1.auc,
    "\nAccuracy:    ", xgb.m1.accuracy,
    "\nSensitivity: ", xgb.m1.sensitivity,
    "\nSpecificity: ", xgb.m1.specificity)
```

```
## AUC:          0.9906
## Accuracy:     0.997
## Sensitivity:  0.5649
## Specificity:  0.9993
```

Best Models from each phase Comparison

```
cat("Best Predictions:")
```

```
## Best Predictions:
```

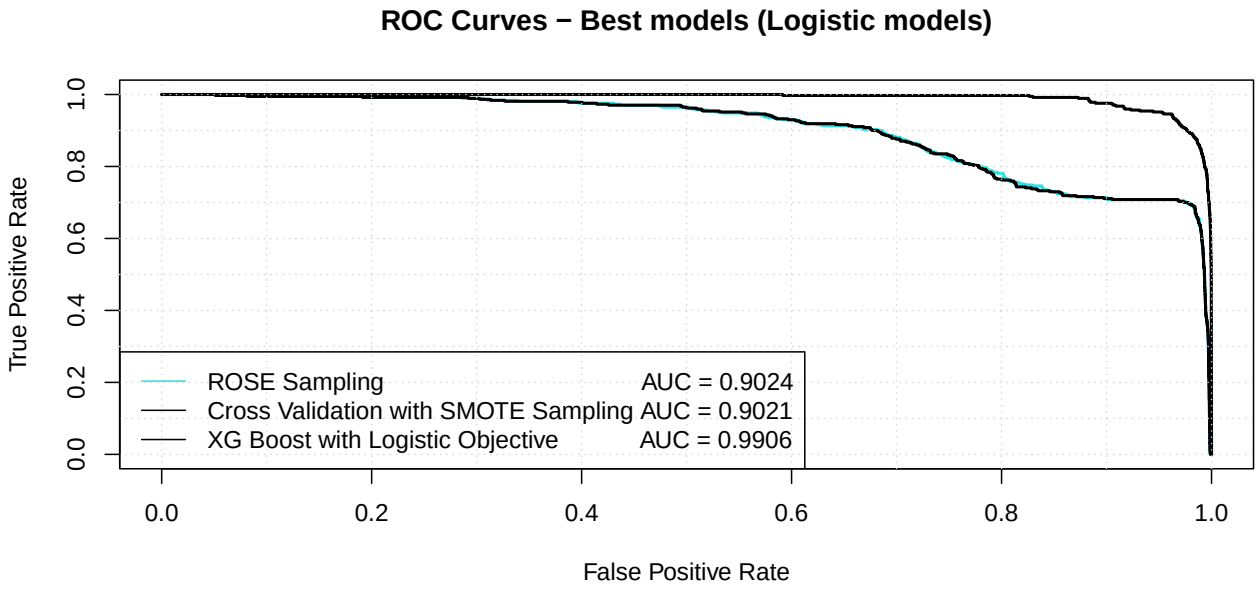
```
list(
  ROSESampling = round(table(is_fraud.test, log.m5.preds)/nrow(ccf.test),4),
  CV_SMOTESampling = round(table(is_fraud.test, log.m9.preds)/nrow(ccf.test),4),
  XGB_Logistic = round(table(is_fraud.test, xgb.m1.preds)/nrow(ccf.test),4))
```

```
## $ROSESampling
##              log.m5.preds
## is_fraud.test      0      1
##             0 0.8612 0.1333
##             1 0.0015 0.0039
##
## $CV_SMOTESampling
##              log.m9.preds
## is_fraud.test      0      1
##             0 0.8614 0.1332
##             1 0.0015 0.0039
##
## $XGB_Logistic
##              xgb.m1.preds
## is_fraud.test      0      1
##             0 0.9939 0.0007
##             1 0.0024 0.0031
```

```
cat("Best Models mean errors comparision",
    "\nROSE Sampling:                          ", log.m5.error,
    "\nCross Validation with SMOTE Sampling: ", log.m9.error,
    "\nXGB with Logistic objective:          ", xgb.m1.error)
```

```
## Best Models mean errors comparision
## ROSE Sampling:                          0.1349
## Cross Validation with SMOTE Sampling:  0.1347
## XGB with Logistic objective:           0.003
```

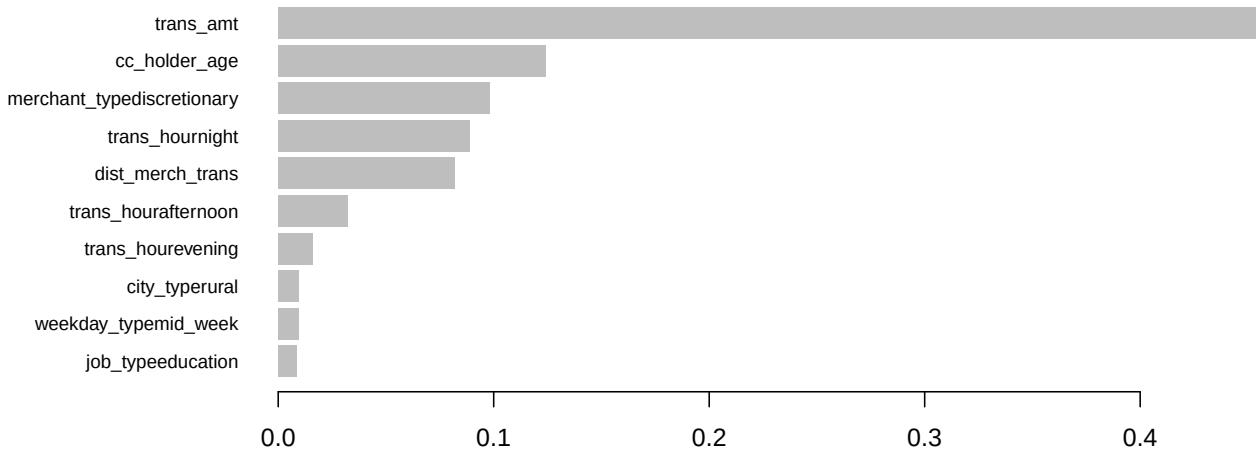
```
model_labels <- c("ROSE Sampling                                        ", "Cross Validation
↪  with SMOTE Sampling",
                  "XG Boost with Logistic Objective           ")

plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Best models (Logistic models)")

legend_labels <- c()
auc_values <- c()

index <- 1
for (i in c(5,9,1)) {
  ifelse(i==1,
         roc_obj <- eval(parse(text = paste0("xgb.m", i, ".roc"))),
         roc_obj <- eval(parse(text = paste0("log.m", i, ".roc")))
         )
  lines(roc_obj, col = i)
  auc_values[i] <- roc_obj$auc
  legend_labels <- c(legend_labels, paste(model_labels[index], "AUC =",
                                          round(auc_values[i], 4)))
  index = index+1
}

abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = c(5,9,1), lwd = 1)
```




```
rm(auc_values, roc_obj, i, model_labels, legend_labels, index)
```

Important Variables from XG Boost:

```
xgb.plot.importance(xgb.importance(model = xgb.m1)[1:10,])
```



Implementations by Vedant Shamling Limbhare

Decision Tree Models with Sampling methods

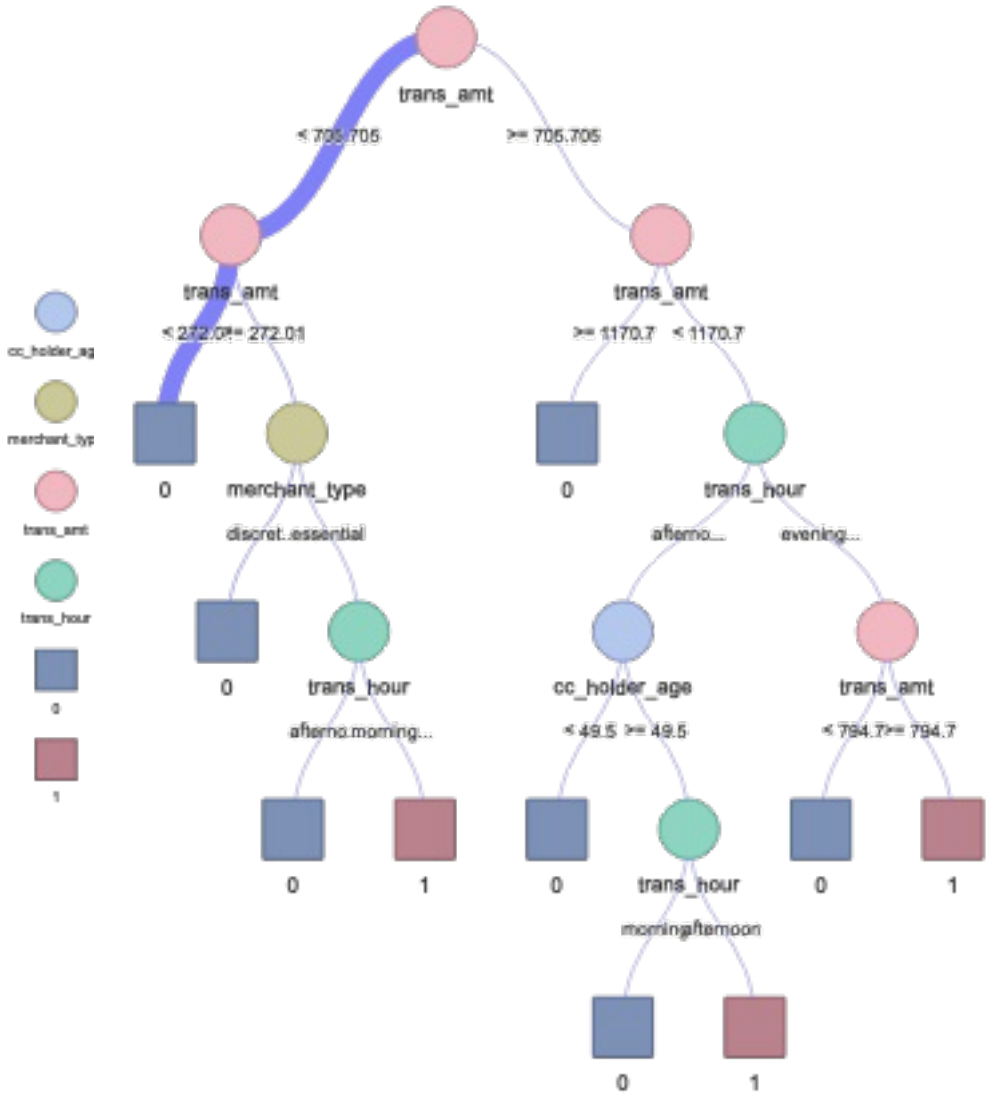
Normal Sampling:

```
dt.m1 <- rpart(is_fraud ~ ., data = ccf.train, method = "class", minbucket = 20)

dt.m1.probs = predict(dt.m1, ccf.test, type = "prob")[, 2]
dt.m1.preds = ifelse(dt.m1.probs > 0.5, 1, 0)
dt.m1.error = round(mean(is_fraud.test != dt.m1.preds), 4)
```

```
dt.m1.roc = roc(is_fraud.test, dt.m1.probs)
dt.m1.auc = round(dt.m1.roc$auc, 4)
dt.m1.cm = confusionMatrix(as.factor(dt.m1.preds), reference = is_fraud.test,
                           positive = '1')
dt.m1.accuracy = round(dt.m1.cm$overall["Accuracy"],4)
dt.m1.sensitivity = round(dt.m1.cm$byClass["Sensitivity"], 4)
dt.m1.specificity = round(dt.m1.cm$byClass["Specificity"], 4)

cat("AUC:         ", dt.m1.auc,
    "\nAccuracy:    ", dt.m1.accuracy,
    "\nSensitivity: ", dt.m1.sensitivity,
    "\nSpecificity: ", dt.m1.specificity)
```

```
## AUC:          0.8347
## Accuracy:     0.9962
## Sensitivity:  0.5135
## Specificity:  0.9988
```

```
visTree(dt.m1)
```



Export as png

```
dt.m1.cm
```

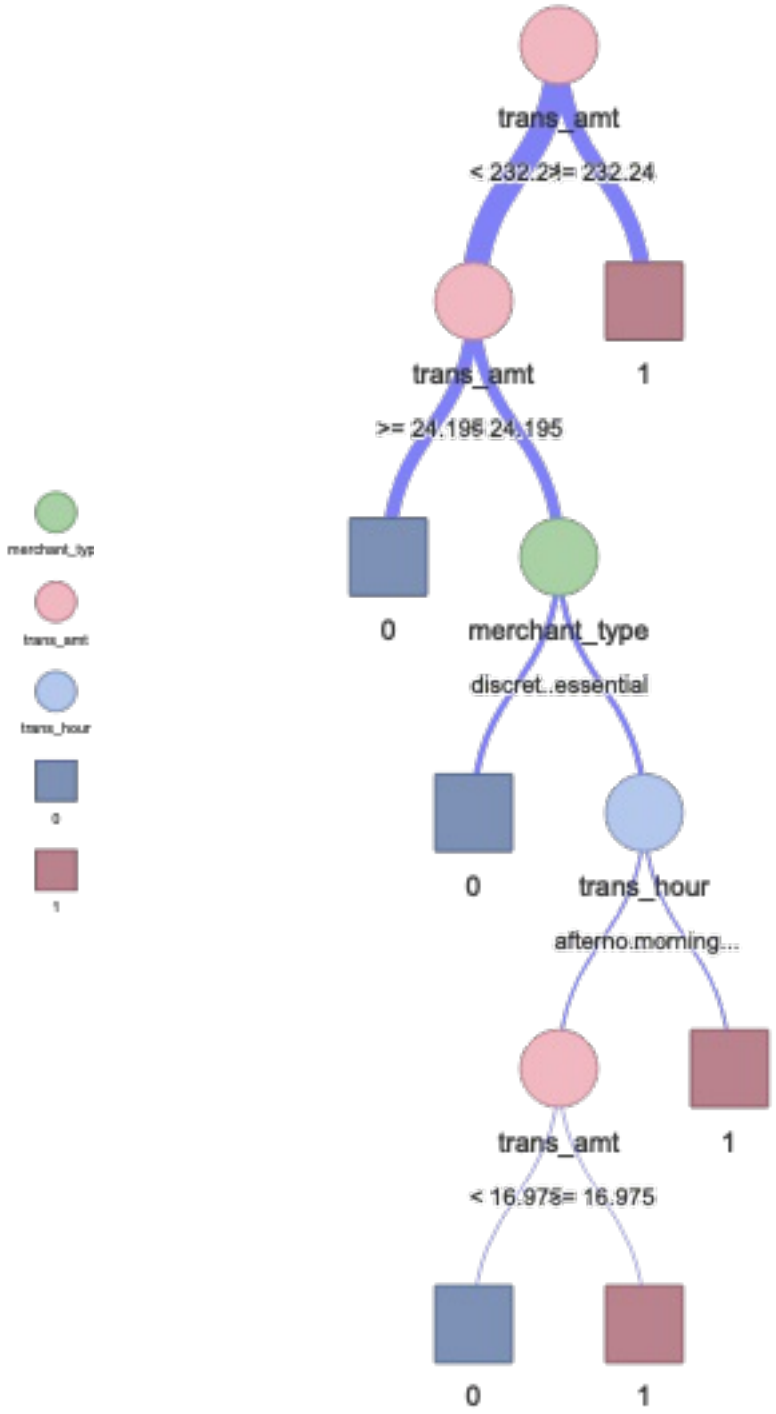
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 67705   180
##          1    79   190
##
##                Accuracy : 0.9962
##                  95% CI : (0.9957, 0.9966)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 6.041e-10
##
##                   Kappa : 0.5928
##
##  Mcnemar's Test P-Value : 5.175e-10
##
##             Sensitivity : 0.513514
##             Specificity : 0.998835
##          Pos Pred Value : 0.706320
##          Neg Pred Value : 0.997348
##              Prevalence : 0.005429
##          Detection Rate : 0.002788
##    Detection Prevalence : 0.003947
##       Balanced Accuracy : 0.756174
##
##        'Positive' Class : 1
##
```

Over Sampling:

```
dt.m2 <- rpart(is_fraud ~ ., data = ccf.train.over, method = "class", minbucket = 20)

dt.m2.probs = predict(dt.m2, ccf.test, type = "prob")[, 2]
dt.m2.preds = ifelse(dt.m2.probs > 0.5, 1, 0)
dt.m2.error = round(mean(is_fraud.test != dt.m2.preds), 4)
dt.m2.roc = roc(is_fraud.test, dt.m2.probs)
dt.m2.auc = round(dt.m2.roc$auc, 4)
dt.m2.cm = confusionMatrix(as.factor(dt.m2.preds), reference = is_fraud.test,
                           positive = '1')
dt.m2.accuracy = round(dt.m2.cm$overall["Accuracy"],4)
dt.m2.sensitivity = round(dt.m2.cm$byClass["Sensitivity"], 4)
dt.m2.specificity = round(dt.m2.cm$byClass["Specificity"], 4)

cat("AUC:         ", dt.m2.auc,
    "\nAccuracy:    ", dt.m2.accuracy,
    "\nSensitivity: ", dt.m2.sensitivity,
    "\nSpecificity: ", dt.m2.specificity)
```

```
## AUC:          0.9441
## Accuracy:     0.935
## Sensitivity:  0.9162
## Specificity:  0.9351
```

```
visTree(dt.m2)
```



Export as png

```
dt.m2.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 63383    31
##          1  4401   339
##
##                Accuracy : 0.935
##                  95% CI : (0.9331, 0.9368)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1239
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.916216
##             Specificity : 0.935073
##          Pos Pred Value : 0.071519
##          Neg Pred Value : 0.999511
##              Prevalence : 0.005429
##          Detection Rate : 0.004974
##    Detection Prevalence : 0.069548
##       Balanced Accuracy : 0.925645
##
##        'Positive' Class : 1
##
```

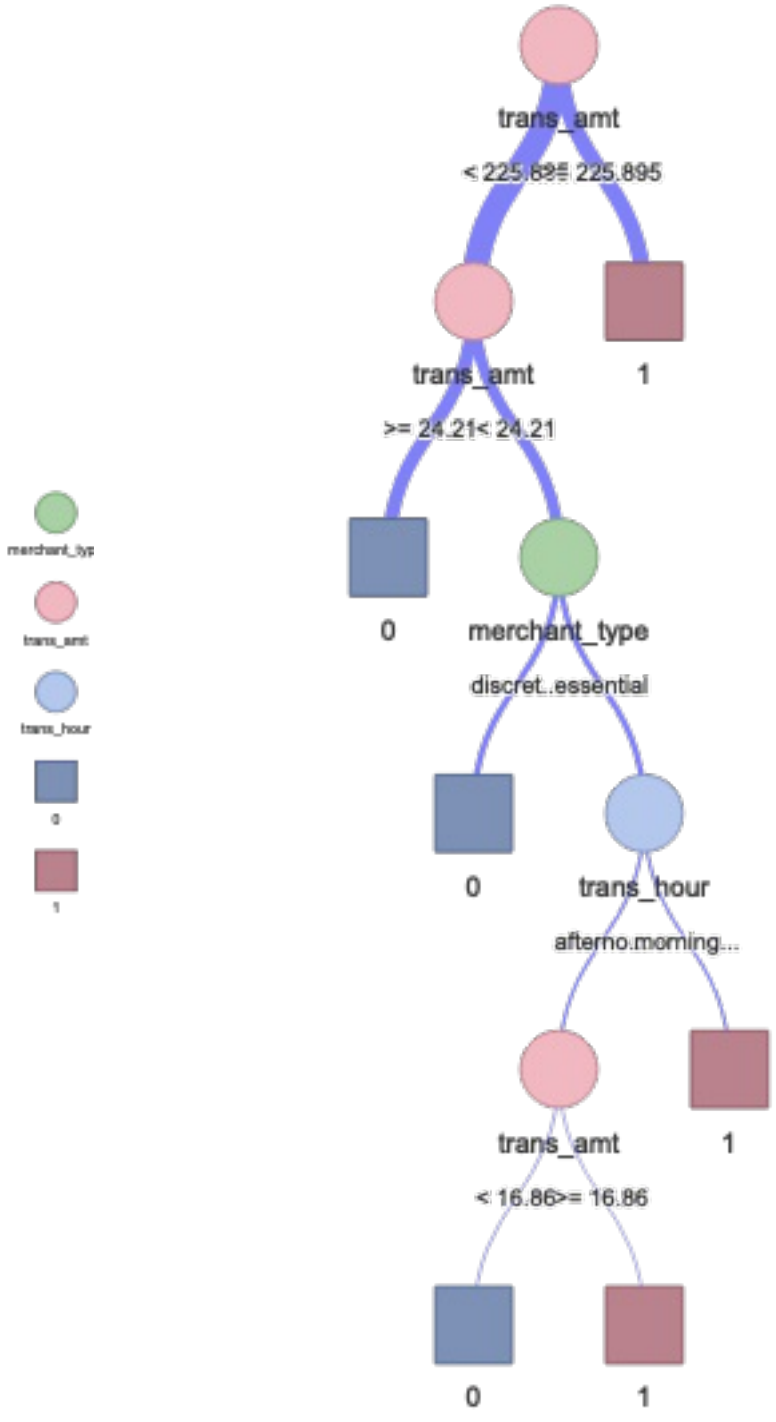
Under sampling:

```
dt.m3 <- rpart(is_fraud ~ ., data = ccf.train.under, method = "class", minbucket = 20)

dt.m3.probs = predict(dt.m3, ccf.test, type = "prob")[, 2]
dt.m3.preds = ifelse(dt.m3.probs > 0.5, 1, 0)
dt.m3.error = round(mean(is_fraud.test != dt.m3.preds), 4)
dt.m3.roc = roc(is_fraud.test, dt.m3.probs)
dt.m3.auc = round(dt.m3.roc$auc, 4)
dt.m3.cm = confusionMatrix(as.factor(dt.m3.preds), reference = is_fraud.test, positive =
↪  '1')
dt.m3.accuracy = round(dt.m3.cm$overall["Accuracy"],4)
dt.m3.sensitivity = round(dt.m3.cm$byClass["Sensitivity"], 4)
dt.m3.specificity = round(dt.m3.cm$byClass["Specificity"], 4)

cat("AUC:         ", dt.m3.auc,
    "\nAccuracy:    ", dt.m3.accuracy,
    "\nSensitivity: ", dt.m3.sensitivity,
    "\nSpecificity: ", dt.m3.specificity)
```

```
## AUC:          0.9461
## Accuracy:     0.9326
```

```
## Sensitivity:  0.9216
## Specificity:  0.9326
```

```
visTree(dt.m3)
```



Export as png

```
dt.m3.cm
```

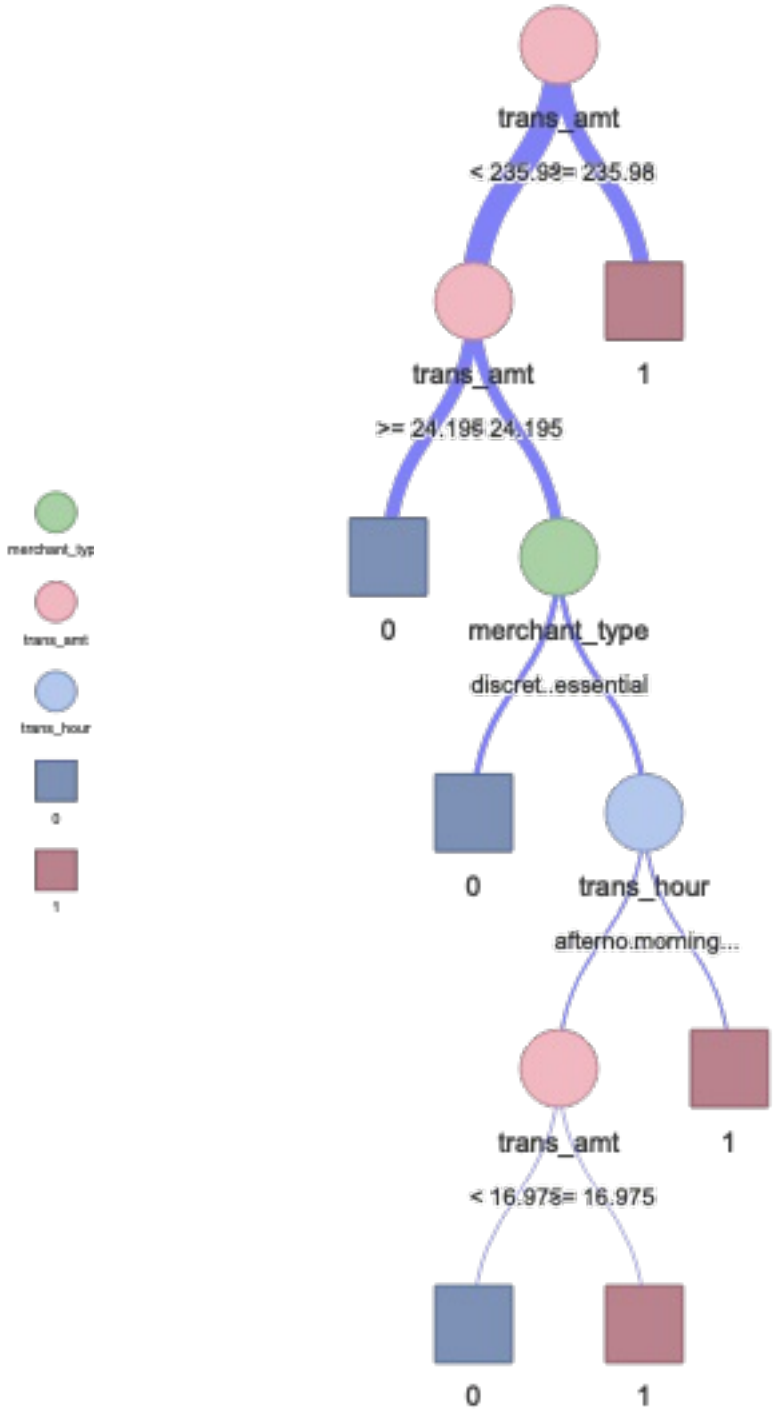
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 63218    29
##          1  4566   341
##
##                Accuracy : 0.9326
##                  95% CI : (0.9307, 0.9345)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1204
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.921622
##             Specificity : 0.932639
##          Pos Pred Value : 0.069493
##          Neg Pred Value : 0.999541
##              Prevalence : 0.005429
##          Detection Rate : 0.005003
##    Detection Prevalence : 0.071999
##       Balanced Accuracy : 0.927130
##
##        'Positive' Class : 1
##
```

Mixed Sampling:

```
dt.m4 <- rpart(is_fraud ~ ., data = ccf.train.mixed, method = "class", minbucket = 20)

dt.m4.probs = predict(dt.m4, ccf.test, type = "prob")[, 2]
dt.m4.preds = ifelse(dt.m4.probs > 0.5, 1, 0)
dt.m4.error = round(mean(is_fraud.test != dt.m4.preds), 4)
dt.m4.roc = roc(is_fraud.test, dt.m4.probs)
dt.m4.auc = round(dt.m4.roc$auc, 4)
dt.m4.cm = confusionMatrix(as.factor(dt.m4.preds), reference = is_fraud.test,
                           positive = '1')
dt.m4.accuracy = round(dt.m4.cm$overall["Accuracy"],4)
dt.m4.sensitivity = round(dt.m4.cm$byClass["Sensitivity"], 4)
dt.m4.specificity = round(dt.m4.cm$byClass["Specificity"], 4)

cat("AUC:         ", dt.m4.auc,
    "\nAccuracy:    ", dt.m4.accuracy,
    "\nSensitivity: ", dt.m4.sensitivity,
    "\nSpecificity: ", dt.m4.specificity)
```

```
## AUC:          0.9448
## Accuracy:     0.9362
## Sensitivity:  0.9162
## Specificity:  0.9363
```

```
visTree(dt.m4)
```



Export as png

```
dt.m4.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 63464    31
##          1  4320   339
##
##                Accuracy : 0.9362
##                  95% CI : (0.9343, 0.938)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.126
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.916216
##             Specificity : 0.936268
##          Pos Pred Value : 0.072762
##          Neg Pred Value : 0.999512
##              Prevalence : 0.005429
##          Detection Rate : 0.004974
##    Detection Prevalence : 0.068360
##       Balanced Accuracy : 0.926242
##
##        'Positive' Class : 1
##
```

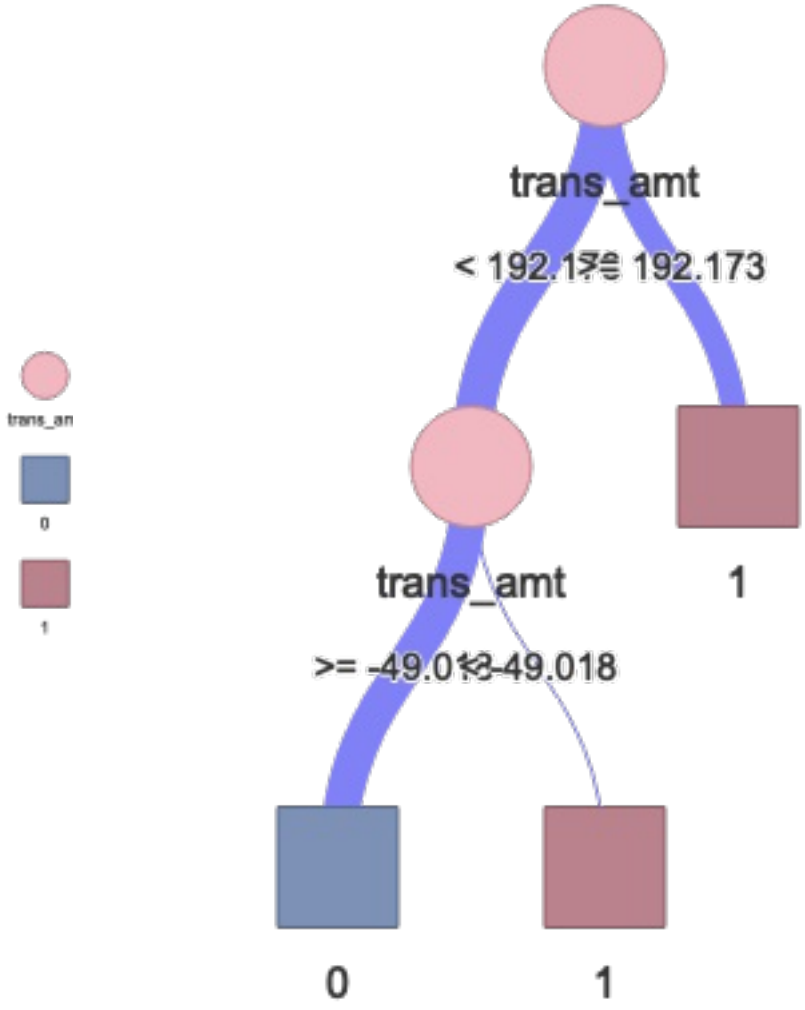
ROSE Sampling:

```
dt.m5 <- rpart(is_fraud ~ ., data = ccf.train.rose, method = "class", minbucket = 20)

dt.m5.probs = predict(dt.m5, ccf.test, type = "prob")[, 2]
dt.m5.preds = ifelse(dt.m5.probs > 0.5, 1, 0)
dt.m5.error = round(mean(is_fraud.test != dt.m5.preds), 4)
dt.m5.roc = roc(is_fraud.test, dt.m5.probs)
dt.m5.auc = round(dt.m5.roc$auc, 4)
dt.m5.cm = confusionMatrix(as.factor(dt.m5.preds), reference = is_fraud.test,
                           positive = '1')
dt.m5.accuracy = round(dt.m5.cm$overall["Accuracy"],4)
dt.m5.sensitivity = round(dt.m5.cm$byClass["Sensitivity"], 4)
dt.m5.specificity = round(dt.m5.cm$byClass["Specificity"], 4)

cat("AUC:         ", dt.m5.auc,
    "\nAccuracy:    ", dt.m5.accuracy,
    "\nSensitivity: ", dt.m5.sensitivity,
    "\nSpecificity: ", dt.m5.specificity)
```

```
## AUC:          0.8357
## Accuracy:     0.9485
```

```
## Sensitivity:  0.7216
## Specificity:  0.9497
```

```
visTree(dt.m5)
```



Export as png

```
dt.m5.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 64374   103
##          1  3410   267
##
##                Accuracy : 0.9485
##                  95% CI : (0.9468, 0.9501)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1233
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.721622
##             Specificity : 0.949693
##          Pos Pred Value : 0.072614
##          Neg Pred Value : 0.998403
##              Prevalence : 0.005429
##          Detection Rate : 0.003918
##    Detection Prevalence : 0.053951
##       Balanced Accuracy : 0.835657
##
##        'Positive' Class : 1
##
```

```
cat("Predictions of Decision Tree with different sampling methods:")
```

Decision Tree Models Comparison

```
## Predictions of Decision Tree with different sampling methods:
```

```
list(
  Normal_Sampling = round(table(is_fraud.test, dt.m1.preds)/nrow(ccf.test),4),
  Over_Sampling = round(table(is_fraud.test, dt.m2.preds)/nrow(ccf.test),4),
  Under_Sampling = round(table(is_fraud.test, dt.m3.preds)/nrow(ccf.test),4),
  Mixed_Sampling = round(table(is_fraud.test, dt.m4.preds)/nrow(ccf.test),4),
  ROSE_Sampling = round(table(is_fraud.test, dt.m5.preds)/nrow(ccf.test),4))
```

```
## $Normal_Sampling
##              dt.m1.preds
## is_fraud.test      0      1
##             0 0.9934 0.0012
##             1 0.0026 0.0028
##
## $Over_Sampling
##              dt.m2.preds
## is_fraud.test      0      1
```

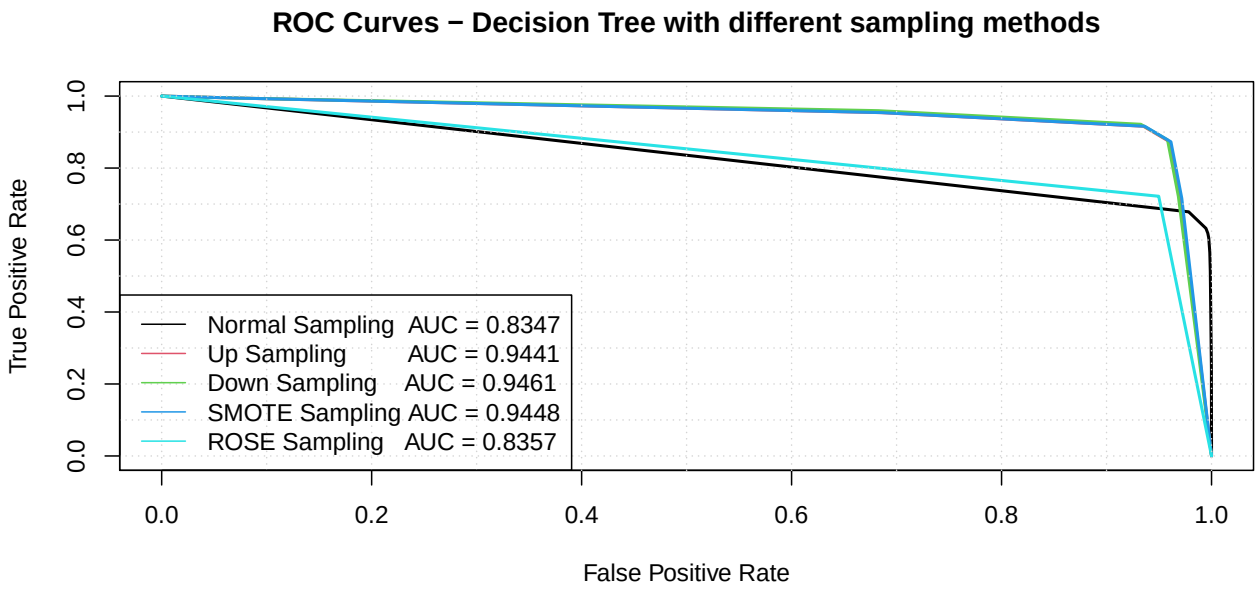
```
##             0 0.9300 0.0646
##             1 0.0005 0.0050
##
## $Under_Sampling
##              dt.m3.preds
## is_fraud.test      0      1
##             0 0.9276 0.0670
##             1 0.0004 0.0050
##
## $Mixed_Sampling
##              dt.m4.preds
## is_fraud.test      0      1
##             0 0.9312 0.0634
##             1 0.0005 0.0050
##
## $ROSE_Sampling
##              dt.m5.preds
## is_fraud.test      0      1
##             0 0.9445 0.0500
##             1 0.0015 0.0039
```

```r
cat("Mean errors",
    "\nNormal Sampling: ", dt.m1.error,
    "\nOver Sampling:   ", dt.m2.error,
    "\nUnder Sampling:  ", dt.m3.error,
    "\nMixed Sampling:  ", dt.m4.error,
    "\nROSE Sampling:   ", dt.m5.error)
```

```
## Mean errors
## Normal Sampling:  0.0038
## Over Sampling:    0.065
## Under Sampling:   0.0674
## Mixed Sampling:   0.0638
## ROSE Sampling:    0.0515
```

```r
model_labels <- c("Normal Sampling ", "Up Sampling        " ,"Down Sampling   ", "SMOTE
↪  Sampling", "ROSE Sampling  ")

plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Decision Tree with different sampling methods")

legend_labels <- c()
auc_values <- c()

for (i in 1:5) {
  roc_obj <- roc_obj <- eval(parse(text = paste0("dt.m", i, ".roc")))
  lines(roc_obj, col = i)
  auc_values[i] <- auc(roc_obj)
  legend_labels <- c(legend_labels, paste(model_labels[i], "AUC =", round(auc_values[i],
↪  4)))
}
```

```
abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = 1:5, lwd = 1)
```




```
rm(auc_values, roc_obj, i, model_labels, legend_labels)
```

Random Forests Models with Sampling methods

Normal Sampling:

```
rf.m1 <- randomForest(is_fraud ~ ., data = ccf.train, ntree = 200, nodesize = 20)

rf.m1.probs = predict(rf.m1, ccf.test, type = "prob")[, 2]
rf.m1.preds = ifelse(rf.m1.probs > 0.5, 1, 0)
rf.m1.error = round(mean(is_fraud.test != rf.m1.preds), 4)
rf.m1.roc = roc(is_fraud.test, rf.m1.probs)
rf.m1.auc = round(rf.m1.roc$auc, 4)
rf.m1.cm = confusionMatrix(as.factor(rf.m1.preds), reference = is_fraud.test,
                           positive = '1')
rf.m1.accuracy = round(rf.m1.cm$overall["Accuracy"],4)
rf.m1.sensitivity = round(rf.m1.cm$byClass["Sensitivity"], 4)
rf.m1.specificity = round(rf.m1.cm$byClass["Specificity"], 4)

cat("AUC:         ", rf.m1.auc,
    "\nAccuracy:    ", rf.m1.accuracy,
    "\nSensitivity: ", rf.m1.sensitivity,
    "\nSpecificity: ", rf.m1.specificity)
```

```
## AUC:          0.9567
## Accuracy:     0.9967
## Sensitivity:  0.4838
## Specificity:  0.9995
```

Over Sampling:

```
rf.m2 <- randomForest(is_fraud ~ ., data = ccf.train.over, ntree = 100, nodesize = 20)

rf.m2.probs = predict(rf.m2, ccf.test, type = "prob")[, 2]
rf.m2.preds = ifelse(rf.m2.probs > 0.5, 1, 0)
rf.m2.error = round(mean(is_fraud.test != rf.m2.preds), 4)
rf.m2.roc = roc(is_fraud.test, rf.m2.probs)
rf.m2.auc = round(rf.m2.roc$auc, 4)
rf.m2.cm = confusionMatrix(as.factor(rf.m2.preds), reference = is_fraud.test,
                           positive = '1')
rf.m2.accuracy = round(rf.m2.cm$overall["Accuracy"],4)
rf.m2.sensitivity = round(rf.m2.cm$byClass["Sensitivity"], 4)
rf.m2.specificity = round(rf.m2.cm$byClass["Specificity"], 4)

cat("AUC:         ", rf.m2.auc,
    "\nAccuracy:    ", rf.m2.accuracy,
    "\nSensitivity: ", rf.m2.sensitivity,
    "\nSpecificity: ", rf.m2.specificity)
```

```
## AUC:          0.9768
## Accuracy:     0.997
## Sensitivity:  0.6243
## Specificity:  0.999
```

Under Sampling:

```
rf.m3 <- randomForest(is_fraud ~ ., data = ccf.train.under, ntree = 500,
                      nodesize = 20)

rf.m3.probs = predict(rf.m3, ccf.test, type = "prob")[, 2]
rf.m3.preds = ifelse(rf.m3.probs > 0.5, 1, 0)
rf.m3.error = round(mean(is_fraud.test != rf.m3.preds), 4)
rf.m3.roc = roc(is_fraud.test, rf.m3.probs)
rf.m3.auc = round(rf.m3.roc$auc, 4)
rf.m3.cm = confusionMatrix(as.factor(rf.m3.preds), reference = is_fraud.test,
                           positive = '1')
rf.m3.accuracy = round(rf.m3.cm$overall["Accuracy"],4)
rf.m3.sensitivity = round(rf.m3.cm$byClass["Sensitivity"], 4)
rf.m3.specificity = round(rf.m3.cm$byClass["Specificity"], 4)

cat("AUC:         ", rf.m3.auc,
    "\nAccuracy:    ", rf.m3.accuracy,
    "\nSensitivity: ", rf.m3.sensitivity,
    "\nSpecificity: ", rf.m3.specificity)
```

```
## AUC:          0.9817
## Accuracy:     0.9558
## Sensitivity:  0.9135
## Specificity:  0.956
```

Mixed Sampling:

```
rf.m4 <- randomForest(is_fraud ~ ., data = ccf.train.mixed, ntree = 200,
                      nodesize = 20)

rf.m4.probs = predict(rf.m4, ccf.test, type = "prob")[, 2]
rf.m4.preds = ifelse(rf.m4.probs > 0.5, 1, 0)
rf.m4.error = round(mean(is_fraud.test != rf.m4.preds), 4)
rf.m4.roc = roc(is_fraud.test, rf.m4.probs)
rf.m4.auc = round(rf.m4.roc$auc, 4)
rf.m4.cm = confusionMatrix(as.factor(rf.m4.preds), reference = is_fraud.test,
                           positive = '1')
rf.m4.accuracy = round(rf.m4.cm$overall["Accuracy"],4)
rf.m4.sensitivity = round(rf.m4.cm$byClass["Sensitivity"], 4)
rf.m4.specificity = round(rf.m4.cm$byClass["Specificity"], 4)

cat("AUC:         ", rf.m4.auc,
    "\nAccuracy:    ", rf.m4.accuracy,
    "\nSensitivity: ", rf.m4.sensitivity,
    "\nSpecificity: ", rf.m4.specificity)
```

```
## AUC:          0.9833
## Accuracy:     0.9962
## Sensitivity:  0.6838
## Specificity:  0.9979
```

Rose Sampling:

```
rf.m5 <- randomForest(is_fraud ~ ., data = ccf.train.rose, ntree = 200, nodesize = 20)

rf.m5.probs = predict(rf.m5, ccf.test, type = "prob")[, 2]
rf.m5.preds = ifelse(rf.m5.probs > 0.5, 1, 0)
rf.m5.error = round(mean(is_fraud.test != rf.m5.preds), 4)
rf.m5.roc = roc(is_fraud.test, rf.m5.probs)
rf.m5.auc = round(rf.m5.roc$auc, 4)
rf.m5.cm = confusionMatrix(as.factor(rf.m5.preds), reference = is_fraud.test,
                           positive = '1')
rf.m5.accuracy = round(rf.m5.cm$overall["Accuracy"],4)
rf.m5.sensitivity = round(rf.m5.cm$byClass["Sensitivity"], 4)
rf.m5.specificity = round(rf.m5.cm$byClass["Specificity"], 4)

cat("AUC:         ", rf.m5.auc,
    "\nAccuracy:    ", rf.m5.accuracy,
    "\nSensitivity: ", rf.m5.sensitivity,
    "\nSpecificity: ", rf.m5.specificity)
```

```
## AUC:          0.9438
## Accuracy:     0.9749
## Sensitivity:  0.7459
## Specificity:  0.9762
```

```
cat("Predictions of Random Forests with different sampling methods:")
```

Random Forests Models Comparison

```
## Predictions of Random Forests with different sampling methods:
```

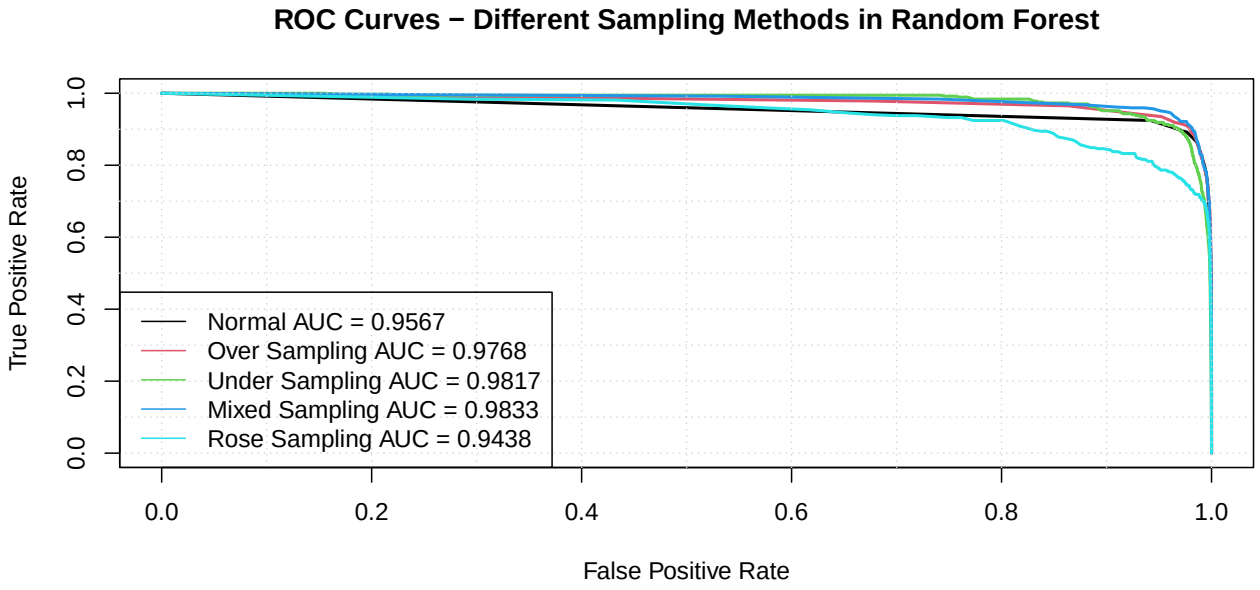
```
list(
  Normal_Sampling = round(table(is_fraud.test, rf.m1.preds)/nrow(ccf.test),4),
  Over_Sampling = round(table(is_fraud.test, rf.m2.preds)/nrow(ccf.test),4),
  Under_Sampling = round(table(is_fraud.test, rf.m3.preds)/nrow(ccf.test),4),
  Mixed_Sampling = round(table(is_fraud.test, rf.m4.preds)/nrow(ccf.test),4),
  ROSE_Sampling = round(table(is_fraud.test, rf.m5.preds)/nrow(ccf.test),4))
```

```
## $Normal_Sampling
##              rf.m1.preds
## is_fraud.test      0      1
##             0 0.9941 0.0005
##             1 0.0028 0.0026
##
## $Over_Sampling
##              rf.m2.preds
## is_fraud.test      0      1
##             0 0.9936 0.0010
##             1 0.0020 0.0034
##
## $Under_Sampling
##              rf.m3.preds
## is_fraud.test      0      1
##             0 0.9508 0.0437
##             1 0.0005 0.0050
##
## $Mixed_Sampling
##              rf.m4.preds
## is_fraud.test      0      1
##             0 0.9925 0.0020
##             1 0.0017 0.0037
##
## $ROSE_Sampling
##              rf.m5.preds
## is_fraud.test      0      1
##             0 0.9709 0.0237
##             1 0.0014 0.0040
```

```
cat("Mean errors",
    "\nNormal Sampling: ", rf.m1.error,
    "\nOver Sampling:   ", rf.m2.error,
    "\nUnder Sampling:  ", rf.m3.error,
    "\nMixed Sampling:  ", rf.m4.error,
    "\nROSE Sampling:   ", rf.m5.error)
```

```
## Mean errors
## Normal Sampling:  0.0033
## Over Sampling:    0.003
## Under Sampling:   0.0442
## Mixed Sampling:   0.0038
## ROSE Sampling:    0.0251
```

```r
model_labels <- c("Normal", "Over Sampling" ,"Under Sampling", "Mixed Sampling",
                  "Rose Sampling")

plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Different Sampling Methods in Random Forest")

legend_labels <- c()
auc_values <- c()

for (i in 1:5) {
  roc_obj <- roc_obj <- eval(parse(text = paste0("rf.m", i, ".roc")))
  lines(roc_obj, col = i)
  auc_values[i] <- auc(roc_obj)
  legend_labels <- c(legend_labels, paste(model_labels[i], "AUC =", round(auc_values[i],
↪  4)))
}

abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = 1:5, lwd = 1)
```



```
rm(auc_values, roc_obj, i, model_labels, legend_labels)
```

XG Boost with logistic objective and GBTree booster

```
xgb.train.dm <- xgb.DMatrix(data = as.matrix(xgb.train.x), label = xgb.train.y)
xgb.test.dm <- xgb.DMatrix(data = as.matrix(xgb.test.x))

params <- list(
  booster = "gbtree",
  objective = "binary:logistic",
  colsample_bytree = 0.8,
  subsample = 0.8,
  max_depth = 6,
  eta = 0.3,
  min_child_weight = 1,
  gamma = 0,
  alpha = 0)

custom_eval_metric <- function(preds, dtrain) {
  labels <- getinfo(dtrain, "label")
  auc <- caret::defaultSummary(
    data.frame(obs = labels, pred = preds)
  )$AUC
  return(list(metric = "auc", value = auc))
}

xgb.m2 <- xgb.train(
  params = params,
  data = xgb.train.dm,
  nthread = 4,
  nrounds = 100,
  verbose = 0,
  feval = custom_eval_metric)

rm(params, custom_eval_metric)

xgb.m2.probs <- predict(xgb.m2, xgb.test.dm)
xgb.m2.preds <- ifelse(xgb.m2.probs > 0.5, 1, 0)
xgb.m2.error = round(mean(is_fraud.test != xgb.m2.preds), 4)
xgb.m2.roc = roc(is_fraud.test, xgb.m2.probs)
xgb.m2.auc = round(xgb.m2.roc$auc, 4)
xgb.m2.cm = confusionMatrix(as.factor(xgb.m2.preds), reference = is_fraud.test,
                            positive = '1')
xgb.m2.accuracy = round(xgb.m2.cm$overall["Accuracy"],4)
xgb.m2.sensitivity = round(xgb.m2.cm$byClass["Sensitivity"], 4)
xgb.m2.specificity = round(xgb.m2.cm$byClass["Specificity"], 4)

cat("AUC:         ", xgb.m2.auc,
    "\nAccuracy:    ", xgb.m2.accuracy,
    "\nSensitivity: ", xgb.m2.sensitivity,
    "\nSpecificity: ", xgb.m2.specificity)
```

```
## AUC:          0.9914
## Accuracy:     0.997
## Sensitivity:  0.5622
## Specificity:  0.9994
```

Best Models from each phase Comparison

```
cat("Best Predictions:")
```

```
## Best Predictions:
```

```
list(
  DecisionTree_UnderSampling = table(is_fraud.test, dt.m3.preds),
  RandomForests_MixedSampling = table(is_fraud.test, rf.m4.preds),
  XGB_GBTree = table(is_fraud.test, xgb.m2.preds)
)
```

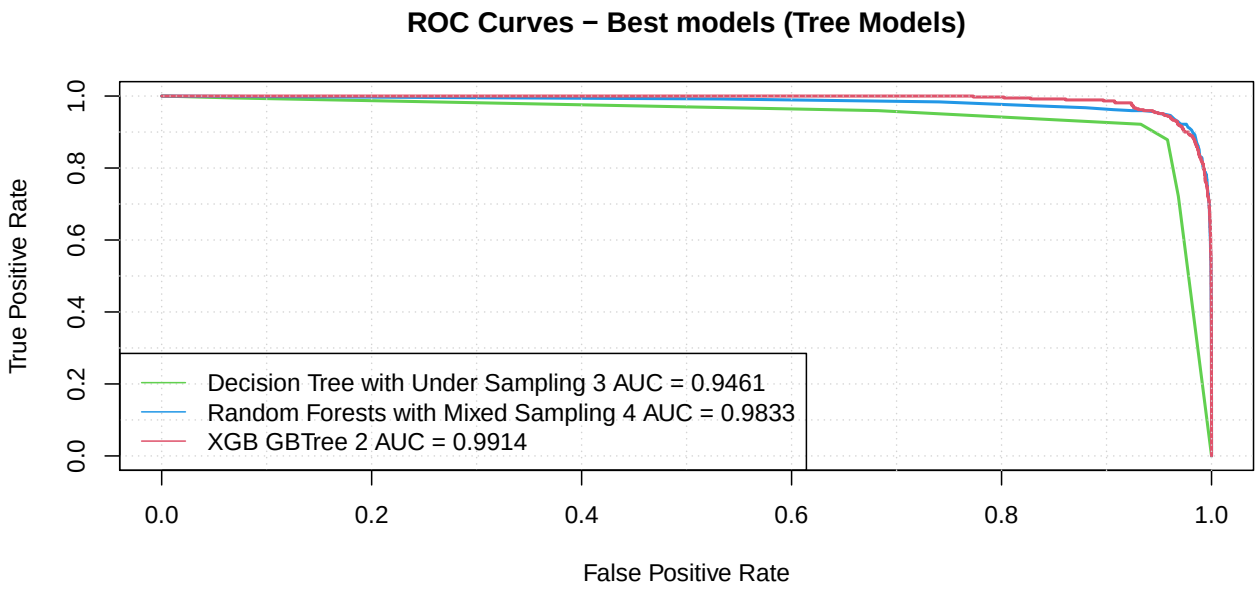
```
## $DecisionTree_UnderSampling
##              dt.m3.preds
## is_fraud.test     0     1
##             0 63218  4566
##             1    29   341
##
## $RandomForests_MixedSampling
##              rf.m4.preds
## is_fraud.test     0     1
##             0 67645   139
##             1   117   253
##
## $XGB_GBTree
##              xgb.m2.preds
## is_fraud.test     0     1
##             0 67743    41
##             1   162   208
```

```
cat("Best Models Mean errors comparision",
    "\nDecision Tree with Under Sampling:               ", dt.m3.error,
    "\nRandom Forests with Mixed Sampling:              ", rf.m4.error,
    "\nXGB with logistic objective and gbtree booster:  ", xgb.m2.error)
```

```
## Best Models Mean errors comparision
## Decision Tree with Under Sampling:               0.0674
## Random Forests with Mixed Sampling:              0.0038
## XGB with logistic objective and gbtree booster:  0.003
```

```
model_labels <- c("Decision Tree with Under Sampling", "Random Forests with Mixed
↪  Sampling",
                  "XGB GBTree")
```

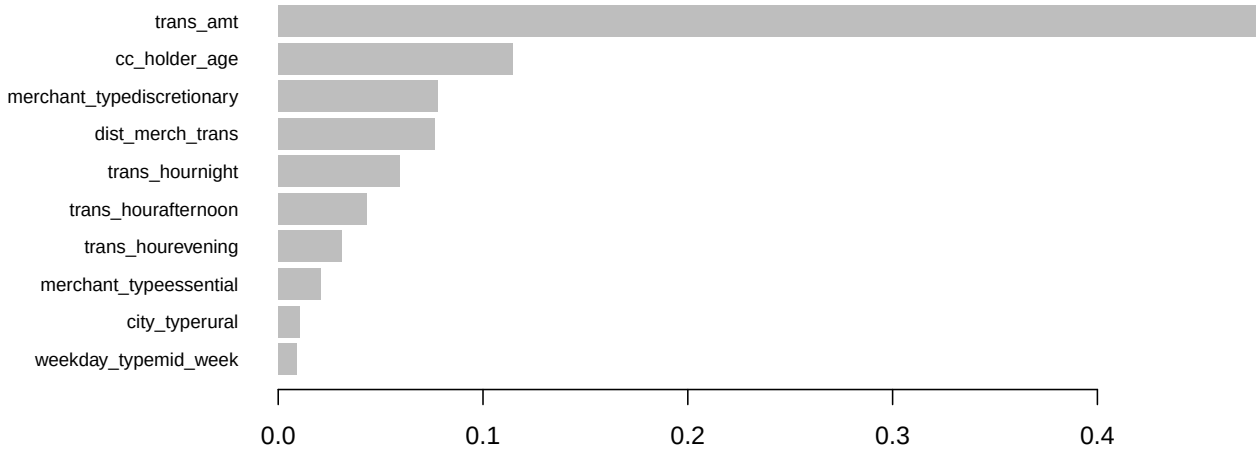
```
plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC Curves - Best models (Tree Models)")

legend_labels <- c()
auc_values <- c()

index <- 1
for (i in c(3,4,2)) {
  ifelse(i==3,
         roc_obj <- eval(parse(text = paste0("dt.m", i, ".roc"))),
         ifelse(i==4,
                roc_obj <- eval(parse(text = paste0("rf.m", i, ".roc"))),
                roc_obj <- eval(parse(text = paste0("xgb.m", i, ".roc")))))
  lines(roc_obj, col = i)
  auc_values[i] <- roc_obj$auc
  legend_labels <- c(legend_labels, paste(model_labels[index], i, "AUC =",
                                          round(auc_values[i], 4)))
  index = index+1
}

abline(v = seq(0, 1, 0.1), h = seq(0, 1, 0.1), col = "lightgray", lty = "dotted")

legend("bottomleft", legend = legend_labels, col = c(3,4,2), lwd = 1)
```




```
rm(auc_values, roc_obj, i, model_labels, legend_labels, index)
```

Important Variables from XG Boost:

```
xgb.plot.importance(xgb.importance(model = xgb.m2)[1:10])
```



All Models Summary

```
model_summary <- data.frame(Model = character(10),
                            AUC = numeric(10),
                            Accuracy = numeric(10),
                            Sensitivity = numeric(10),
                            Specificity = numeric(10),
                            MeanError = numeric(10))

lab1 <- c("Normal Sampling", "Over Sampling", "Under Sampling", "Mixed Sampling", "ROSE
↪  Sampling",
          "Cross Validation using Normal Sampling", "Cross Validation using Over
          ↪  Sampling",
          "Cross Validation using Under Sampling", "Cross Validation using SMOTE
          ↪  Sampling",
          "Cross Validation using ROSE Sampling")

lab2 <- c("Logistic Objective", "Logistic Objective and GBTree Booster")

for (i in 1:10) {
  auc <- eval(parse(text = paste0("log.m", i, ".auc")))
  accuracy <- eval(parse(text = paste0("log.m", i, ".accuracy")))
  sensitivity <- eval(parse(text = paste0("log.m", i, ".sensitivity")))
  specificity <- eval(parse(text = paste0("log.m", i, ".specificity")))
  mean_error <- eval(parse(text = paste0("log.m", i, ".error")))

  model_summary[i, "Model"] <- paste0("Logistic Regression: ", lab1[i])
  model_summary[i, "AUC"] <- auc
  model_summary[i, "Accuracy"] <- accuracy
  model_summary[i, "Sensitivity"] <- sensitivity
  model_summary[i, "Specificity"] <- specificity
  model_summary[i, "MeanError"] <- mean_error
}

for (i in 1:5) {
  auc <- eval(parse(text = paste0("dt.m", i, ".auc")))
  accuracy <- eval(parse(text = paste0("dt.m", i, ".accuracy")))
```

```
  sensitivity <- eval(parse(text = paste0("dt.m", i, ".sensitivity")))
  specificity <- eval(parse(text = paste0("dt.m", i, ".specificity")))
  mean_error <- eval(parse(text = paste0("dt.m", i, ".error")))

  model_summary[i+10, "Model"] <- paste0("Decision Tree: ", lab1[i])
  model_summary[i+10, "AUC"] <- auc
  model_summary[i+10, "Accuracy"] <- accuracy
  model_summary[i+10, "Sensitivity"] <- sensitivity
  model_summary[i+10, "Specificity"] <- specificity
  model_summary[i+10, "MeanError"] <- mean_error
}

for (i in 1:5) {
  auc <- eval(parse(text = paste0("rf.m", i, ".auc")))
  accuracy <- eval(parse(text = paste0("rf.m", i, ".accuracy")))
  sensitivity <- eval(parse(text = paste0("rf.m", i, ".sensitivity")))
  specificity <- eval(parse(text = paste0("rf.m", i, ".specificity")))
  mean_error <- eval(parse(text = paste0("rf.m", i, ".error")))

  model_summary[i+15, "Model"] <- paste0("Random Forests: ", lab1[i])
  model_summary[i+15, "AUC"] <- auc
  model_summary[i+15, "Accuracy"] <- accuracy
  model_summary[i+15, "Sensitivity"] <- sensitivity
  model_summary[i+15, "Specificity"] <- specificity
  model_summary[i+15, "MeanError"] <- mean_error
}

for (i in 1:2) {
  auc <- eval(parse(text = paste0("xgb.m", i, ".auc")))
  accuracy <- eval(parse(text = paste0("xgb.m", i, ".accuracy")))
  sensitivity <- eval(parse(text = paste0("xgb.m", i, ".sensitivity")))
  specificity <- eval(parse(text = paste0("xgb.m", i, ".specificity")))
  mean_error <- eval(parse(text = paste0("xgb.m", i, ".error")))

  model_summary[i+20, "Model"] <- paste0("XG Booster: ", lab2[i])
  model_summary[i+20, "AUC"] <- auc
  model_summary[i+20, "Accuracy"] <- accuracy
  model_summary[i+20, "Sensitivity"] <- sensitivity
  model_summary[i+20, "Specificity"] <- specificity
  model_summary[i+20, "MeanError"] <- mean_error
}

rm(i, auc, accuracy, sensitivity, specificity, mean_error, lab1, lab2)

kable(caption = "Models Performance Summary",align = "lrrrrr", model_summary, format =
↪  "markdown")
```

Table 1: Models Performance Summary

Model	AUC	Accuracy	Sensitivity	Specificity	MeanError
Logistic Regression: Normal Sampling	0.8539	0.9944	0.0000	0.9998	0.0056
Logistic Regression: Over Sampling	0.8539	0.8792	0.7081	0.8802	0.1208

Model	AUC	Accuracy	Sensitivity	Specificity	MeanError
Logistic Regression: Under Sampling	0.8981	0.8717	0.7135	0.8725	0.1283
Logistic Regression: Mixed Sampling	0.9004	0.8784	0.8793	0.7081	0.1216
Logistic Regression: ROSE Sampling	0.9024	0.8651	0.7162	0.8659	0.1349
Logistic Regression: Cross Validation using Normal Sampling	0.8539	0.9944	0.0000	0.9998	0.0056
Logistic Regression: Cross Validation using Over Sampling	0.9001	0.8806	0.7081	0.8816	0.1194
Logistic Regression: Cross Validation using Under Sampling	0.8977	0.8684	0.7108	0.8692	0.1316
Logistic Regression: Cross Validation using SMOTE Sampling	0.9021	0.8653	0.7189	0.8661	0.1347
Logistic Regression: Cross Validation using ROSE Sampling	0.8985	0.9109	0.7108	0.9120	0.0891
Decision Tree: Normal Sampling	0.8347	0.9962	0.5135	0.9988	0.0038
Decision Tree: Over Sampling	0.9441	0.9350	0.9162	0.9351	0.0650
Decision Tree: Under Sampling	0.9461	0.9326	0.9216	0.9326	0.0674
Decision Tree: Mixed Sampling	0.9448	0.9362	0.9162	0.9363	0.0638
Decision Tree: ROSE Sampling	0.8357	0.9485	0.7216	0.9497	0.0515
Random Forests: Normal Sampling	0.9567	0.9967	0.4838	0.9995	0.0033
Random Forests: Over Sampling	0.9768	0.9970	0.6243	0.9990	0.0030
Random Forests: Under Sampling	0.9817	0.9558	0.9135	0.9560	0.0442
Random Forests: Mixed Sampling	0.9833	0.9962	0.6838	0.9979	0.0038
Random Forests: ROSE Sampling	0.9438	0.9749	0.7459	0.9762	0.0251
XG Booster: Logistic Objective	0.9906	0.9970	0.5649	0.9993	0.0030
XG Booster: Logistic Objective and GBTree Booster	0.9914	0.9970	0.5622	0.9994	0.0030

```
cat("Best Models Based on different performance metrics",
    "\nAUC:         ", model_summary[which.max(model_summary$AUC),1],
    ↪  "(",max(model_summary$AUC), ")",
    "\nAccuracy:    ", model_summary[which.max(model_summary$Accuracy),1], "
    ↪  (",max(model_summary$Accuracy), " )",
    "\nMean Error:  ", model_summary[which.min(model_summary$MeanError),1], "
    ↪  (",min(model_summary$MeanError), " )")
```

```
## Best Models Based on different performance metrics
## AUC:          XG Booster: Logistic Objective and GBTree Booster ( 0.9914 )
## Accuracy:     Random Forests: Over Sampling                     ( 0.997  )
## Mean Error:   Random Forests: Over Sampling                     ( 0.003  )
```

XG Booster: Logistic Objective and GBTree Booster:

```
xgb.m1.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 67738   161
##          1    46   209
```

```
##
##                Accuracy : 0.997
##                  95% CI : (0.9965, 0.9974)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6673
##
##  Mcnemar's Test P-Value : 2.308e-15
##
##             Sensitivity : 0.564865
##             Specificity : 0.999321
##          Pos Pred Value : 0.819608
##          Neg Pred Value : 0.997629
##              Prevalence : 0.005429
##          Detection Rate : 0.003067
##    Detection Prevalence : 0.003742
##       Balanced Accuracy : 0.782093
##
##        'Positive' Class : 1
##
```

Random Forests: Over Sampling:

```
rf.m2.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 67718   139
##          1    66   231
##
##                Accuracy : 0.997
##                  95% CI : (0.9966, 0.9974)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6912
##
##  Mcnemar's Test P-Value : 4.938e-07
##
##             Sensitivity : 0.624324
##             Specificity : 0.999026
##          Pos Pred Value : 0.777778
##          Neg Pred Value : 0.997952
##              Prevalence : 0.005429
##          Detection Rate : 0.003389
##    Detection Prevalence : 0.004358
##       Balanced Accuracy : 0.811675
##
##        'Positive' Class : 1
##
```

Important Variables in the best models

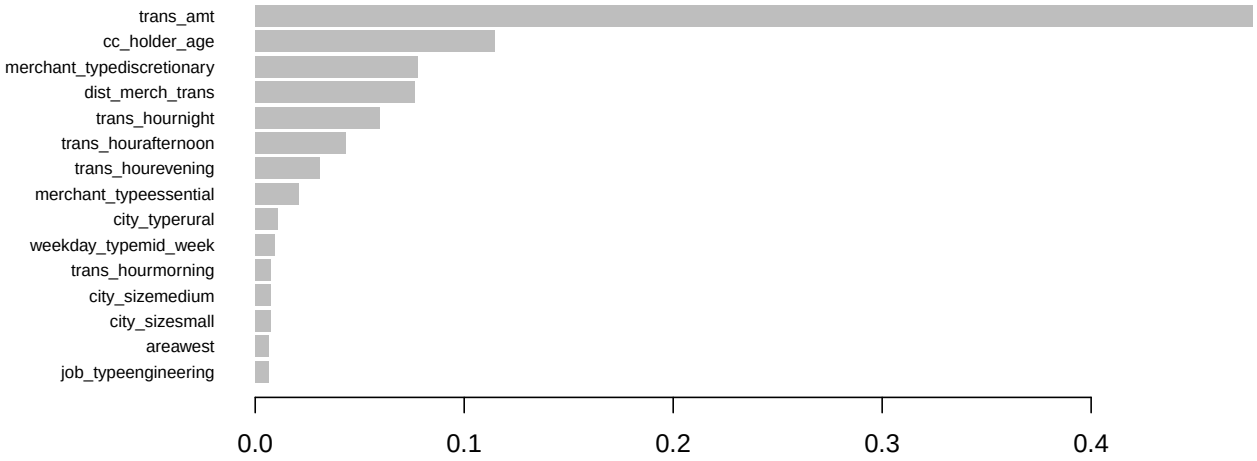
```
imp_vars_auc <- xgb.importance(model = xgb.m2)

kable(caption = "Important Variables: XG Booster",
      align = "lrrrrr", imp_vars_auc, format = "markdown")
```

Important Variables from XG Booster with Logistic Objective and GBTree Booster Model

Table 2: Important Variables: XG Booster

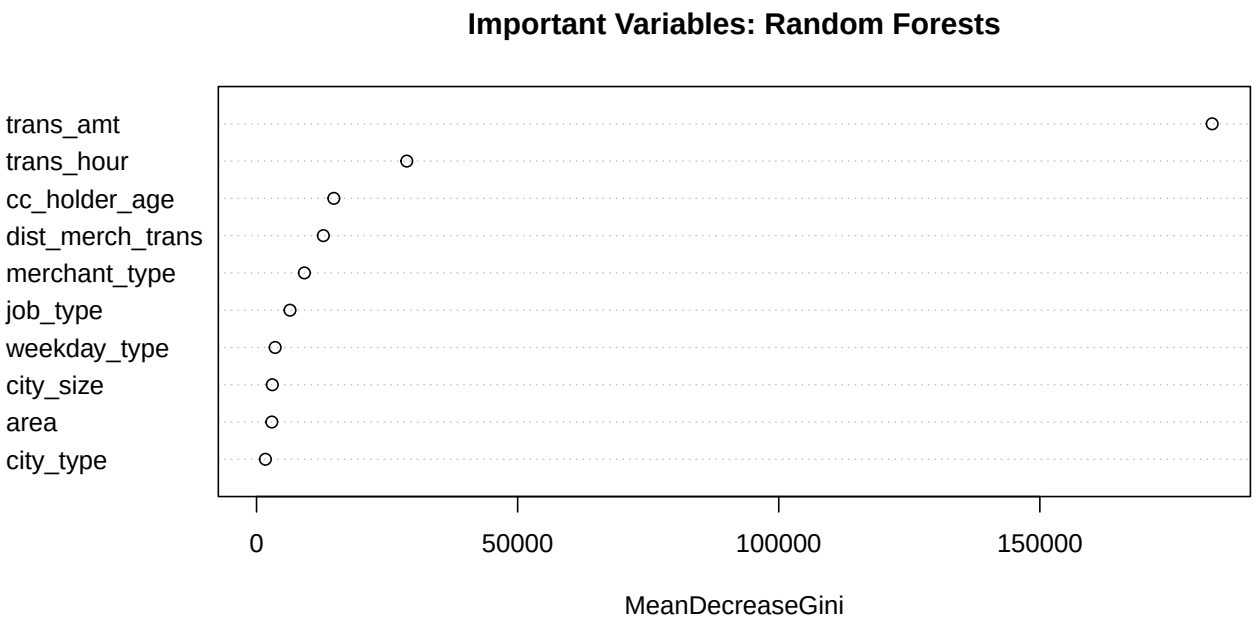
Feature	Gain	Cover	Frequency
trans_amt	0.4772136	0.7304742	0.3030215
cc_holder_age	0.1145622	0.0594703	0.1792562
merchant_typediscretionary	0.0779130	0.0389602	0.0316676
dist_merch_trans	0.0765009	0.0517313	0.1955259
trans_hournight	0.0594519	0.0196452	0.0255665
trans_hourafternoon	0.0431347	0.0297932	0.0133643
trans_hourevening	0.0309851	0.0038417	0.0188844
merchant_typeessential	0.0208816	0.0077985	0.0072632
city_typerural	0.0106473	0.0032727	0.0217897
weekday_typemid_week	0.0091162	0.0050736	0.0220802
trans_hourmorning	0.0072865	0.0166868	0.0055200
city_sizemedium	0.0072819	0.0027842	0.0133643
city_sizesmall	0.0072166	0.0038894	0.0168507
areawest	0.0064890	0.0018486	0.0177223
job_typeengineering	0.0064067	0.0024569	0.0142359
job_typehealthcare	0.0057718	0.0029661	0.0130738
weekday_typeweek_start	0.0047724	0.0028363	0.0116212
areasouthwest	0.0047471	0.0018483	0.0133643
job_typeeducation	0.0047246	0.0021072	0.0130738
job_typebusiness	0.0044991	0.0021049	0.0119117
areamidwest	0.0043719	0.0017810	0.0119117
city_sizelarge	0.0040292	0.0012454	0.0090064
weekday_typeweek_end	0.0039162	0.0016144	0.0107496
job_typecreative	0.0030974	0.0017216	0.0072632
job_typelegal	0.0022278	0.0015407	0.0052295
city_typeurban	0.0019834	0.0003724	0.0046485
city_sizemega	0.0007720	0.0021347	0.0020337

```
xgb.plot.importance(imp_vars_auc[1:15])
```



Important Variables from Random Forests with Over Sampling Model

```
varImpPlot(rf.m2, main = "Important Variables: Random Forests")
```



External Factors Importance in the Fraud Detection

```
prefixes <- c("weekday_type", "area", "city_type", "city_size", "dist_merch_trans")

external_vars <- imp_vars_auc[grep(paste(prefixes, collapse="|"), imp_vars_auc$Feature),
↪  ]

rm(prefixes)

kable(caption = "External factors importance in the Fraud Detection", align = "lrrrrr",
↪  external_vars, format = "markdown")
```

Table 3: External factors importance in the Fraud Detection

Feature	Gain	Cover	Frequency
dist_merch_trans	0.0765009	0.0517313	0.1955259
city_typerural	0.0106473	0.0032727	0.0217897
weekday_typemid_week	0.0091162	0.0050736	0.0220802
city_sizemedium	0.0072819	0.0027842	0.0133643
city_sizesmall	0.0072166	0.0038894	0.0168507
areawest	0.0064890	0.0018486	0.0177223
weekday_typeweek_start	0.0047724	0.0028363	0.0116212
areasouthwest	0.0047471	0.0018483	0.0133643
areamidwest	0.0043719	0.0017810	0.0119117
city_sizelarge	0.0040292	0.0012454	0.0090064
weekday_typeweek_end	0.0039162	0.0016144	0.0107496
city_typeurban	0.0019834	0.0003724	0.0046485
city_sizemega	0.0007720	0.0021347	0.0020337

```
xgb.plot.importance(external_vars)
```