2103 Project

2022-11-15

```
## Warning: package 'knitr' was built under R version 4.1.3

## Warning: package 'dplyr' was built under R version 4.1.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Warning: package 'readxl' was built under R version 4.1.3

## Warning: package 'corrplot' was built under R version 4.1.3

## corrplot 0.92 loaded

## Warning: package 'Hmisc' was built under R version 4.1.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.1.3

## Loading required package: survival

## Warning: package 'survival' was built under R version 4.1.3

## Loading required package: Formula

## Warning: package 'Formula' was built under R version 4.1.1

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units

## Warning: package 'caret' was built under R version 4.1.3

##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster

## Warning: package 'leaps' was built under R version 4.1.3

## Warning: package 'e1071' was built under R version 4.1.3

##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
##     impute

## Warning: package 'ggcorrplot' was built under R version 4.1.3

## Warning: package 'ranger' was built under R version 4.1.3

## Warning: package 'data.table' was built under R version 4.1.3

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## Warning: package 'nnet' was built under R version 4.1.3

## Warning: package 'NeuralNetTools' was built under R version 4.1.3

## Warning: package 'smotefamily' was built under R version 4.1.3

## Warning: package 'ROSE' was built under R version 4.1.3

## Loaded ROSE 0.0-4

## Warning: package 'rpart' was built under R version 4.1.3
```

```
## Warning: package 'ROCR' was built under R version 4.1.3

## Warning: package 'pROC' was built under R version 4.1.1

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

## Warning: package 'caTools' was built under R version 4.1.3

## Warning: package 'partykit' was built under R version 4.1.3

## Loading required package: grid

## Loading required package: libcoin

## Warning: package 'libcoin' was built under R version 4.1.1

## Loading required package: mvtnorm

## Warning: package 'mvtnorm' was built under R version 4.1.1
```

Intro of Dataset

We take a look at the data to see what types of data we are given

```
head(data)
```

```
##   V1     V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12   V13   V14   V15   V16   V17
## 1  1  20000  2  2  1 24  2  2 -1  -1  -2  -2  3913  3102   689     0     0
## 2  2 120000  2  2  2 26 -1  2  0   0   0   2  2682  1725  2682  3272  3455
## 3  3  90000  2  2  2 34  0  0  0   0   0   0 29239 14027 13559 14331 14948
## 4  4  50000  2  2  1 37  0  0  0   0   0   0 46990 48233 49291 28314 28959
## 5  5  50000  1  2  1 57 -1  0 -1   0   0   0  8617  5670 35835 20940 19146
## 6  6  50000  1  1  2 37  0  0  0   0   0   0 64400 57069 57608 19394 19619
##     V18  V19   V20   V21  V22  V23  V24 V25
## 1     0    0   689     0    0    0    0   1
## 2  3261    0  1000  1000 1000    0 2000   1
## 3 15549 1518  1500  1000 1000 1000 5000   0
## 4 29547 2000  2019  1200 1100 1069 1000   0
## 5 19131 2000 36681 10000 9000  689  679   0
## 6 20024 2500  1815   657 1000 1000  800   0
```

```r
glimpse(data)
```

```
## Rows: 30,000
## Columns: 25
## $ V1  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,~
## $ V2  <int> 20000, 120000, 90000, 50000, 50000, 50000, 500000, 100000, 140000,~
## $ V3  <int> 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2, ~
## $ V4  <int> 2, 2, 2, 2, 2, 1, 1, 2, 3, 3, 3, 1, 2, 2, 1, 3, 1, 1, 1, 1, 3, 2, ~
## $ V5  <int> 1, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 3, 2, 1, 1, 2, 2, 1, ~
## $ V6  <int> 24, 26, 34, 37, 57, 37, 29, 23, 28, 35, 34, 51, 41, 30, 29, 23, 24~
## $ V7  <int> 2, -1, 0, 0, -1, 0, 0, 0, 0, -2, 0, -1, -1, 1, 0, 1, 0, 0, 1, 1, 0~
## $ V8  <int> 2, 2, 0, 0, 0, 0, 0, -1, 0, -2, 0, -1, 0, 2, 0, 2, 0, 0, -2, -2, 0~
## $ V9  <int> -1, 0, 0, 0, -1, 0, 0, -1, 2, -2, 2, -1, -1, 2, 0, 0, 2, 0, -2, -2~
## $ V10 <int> -1, 0, 0, 0, 0, 0, 0, 0, 0, -2, 0, -1, -1, 0, 0, 0, 2, -1, -2, -2,~
## $ V11 <int> -2, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1, -1, 0, 0, 0, 2, -1, -2, -2,~
## $ V12 <int> -2, 2, 0, 0, 0, 0, 0, -1, 0, -1, -1, 2, -1, 2, 0, 0, 2, -1, -2, -2~
## $ V13 <int> 3913, 2682, 29239, 46990, 8617, 64400, 367965, 11876, 11285, 0, 11~
## $ V14 <int> 3102, 1725, 14027, 48233, 5670, 57069, 412023, 380, 14096, 0, 9787~
## $ V15 <int> 689, 2682, 13559, 49291, 35835, 57608, 445007, 601, 12108, 0, 5535~
## $ V16 <int> 0, 3272, 14331, 28314, 20940, 19394, 542653, 221, 12211, 0, 2513, ~
## $ V17 <int> 0, 3455, 14948, 28959, 19146, 19619, 483003, -159, 11793, 13007, 1~
## $ V18 <int> 0, 3261, 15549, 29547, 19131, 20024, 473944, 567, 3719, 13912, 373~
## $ V19 <int> 0, 0, 1518, 2000, 2000, 2500, 55000, 380, 3329, 0, 2306, 21818, 10~
## $ V20 <int> 689, 1000, 1500, 2019, 36681, 1815, 40000, 601, 0, 0, 12, 9966, 65~
## $ V21 <int> 0, 1000, 1000, 1200, 10000, 657, 38000, 0, 432, 0, 50, 8583, 6500,~
## $ V22 <int> 0, 1000, 1000, 1100, 9000, 1000, 20239, 581, 1000, 13007, 300, 223~
## $ V23 <int> 0, 0, 1000, 1069, 689, 1000, 13750, 1687, 1000, 1122, 3738, 0, 287~
## $ V24 <int> 0, 2000, 5000, 1000, 679, 800, 13770, 1542, 1000, 0, 66, 3640, 0, ~
## $ V25 <int> 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, ~
```

```r
str(data)
```

```
## ’data.frame’:    30000 obs. of  25 variables:
##  $ V1 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ V2 : int  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
##  $ V3 : int  2 2 2 2 1 1 1 2 2 1 ...
##  $ V4 : int  2 2 2 2 2 1 1 2 3 3 ...
##  $ V5 : int  1 2 2 1 1 2 2 2 1 2 ...
##  $ V6 : int  24 26 34 37 57 37 29 23 28 35 ...
##  $ V7 : int  2 -1 0 0 -1 0 0 0 0 -2 ...
##  $ V8 : int  2 2 0 0 0 0 0 -1 0 -2 ...
##  $ V9 : int  -1 0 0 0 -1 0 0 -1 2 -2 ...
##  $ V10: int  -1 0 0 0 0 0 0 0 0 -2 ...
##  $ V11: int  -2 0 0 0 0 0 0 0 0 -1 ...
##  $ V12: int  -2 2 0 0 0 0 0 -1 0 -1 ...
##  $ V13: int  3913 2682 29239 46990 8617 64400 367965 11876 11285 0 ...
##  $ V14: int  3102 1725 14027 48233 5670 57069 412023 380 14096 0 ...
##  $ V15: int  689 2682 13559 49291 35835 57608 445007 601 12108 0 ...
##  $ V16: int  0 3272 14331 28314 20940 19394 542653 221 12211 0 ...
##  $ V17: int  0 3455 14948 28959 19146 19619 483003 -159 11793 13007 ...
##  $ V18: int  0 3261 15549 29547 19131 20024 473944 567 3719 13912 ...
##  $ V19: int  0 0 1518 2000 2000 2500 55000 380 3329 0 ...
##  $ V20: int  689 1000 1500 2019 36681 1815 40000 601 0 0 ...
```

```
##  $ V21: int  0 1000 1000 1200 10000 657 38000 0 432 0 ...
##  $ V22: int  0 1000 1000 1100 9000 1000 20239 581 1000 13007 ...
##  $ V23: int  0 0 1000 1069 689 1000 13750 1687 1000 1122 ...
##  $ V24: int  0 2000 5000 1000 679 800 13770 1542 1000 0 ...
##  $ V25: int  1 1 0 0 0 0 0 0 0 0 ...
```

We check for any NA values and look at a summary of the variables we have

```
#Checking for NA values
any(is.na(data))
```

```
## [1] FALSE
```

```
summary(data)
```

```
##        V1              V2                V3              V4
##  Min.   :    1   Min.   :  10000   Min.   :1.000   Min.   :0.000
##  1st Qu.: 7501   1st Qu.:  50000   1st Qu.:1.000   1st Qu.:1.000
##  Median :15000   Median : 140000   Median :2.000   Median :2.000
##  Mean   :15000   Mean   : 167484   Mean   :1.604   Mean   :1.853
##  3rd Qu.:22500   3rd Qu.: 240000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :30000   Max.   :1000000   Max.   :2.000   Max.   :6.000
##        V5              V6              V7               V8
##  Min.   :0.000   Min.   :21.00   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:1.000   1st Qu.:28.00   1st Qu.:-1.0000   1st Qu.:-1.0000
##  Median :2.000   Median :34.00   Median : 0.0000   Median : 0.0000
##  Mean   :1.552   Mean   :35.49   Mean   :-0.0167   Mean   :-0.1338
##  3rd Qu.:2.000   3rd Qu.:41.00   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :3.000   Max.   :79.00   Max.   : 8.0000   Max.   : 8.0000
##        V9               V10               V11               V12
##  Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000   Min.   :-2.0000
##  1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000   1st Qu.:-1.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   :-0.1662   Mean   :-0.2207   Mean   :-0.2662   Mean   :-0.2911
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000   Max.   : 8.0000
##       V13               V14               V15               V16
##  Min.   :-165580   Min.   :-69777   Min.   :-157264   Min.   :-170000
##  1st Qu.:   3559   1st Qu.:  2985   1st Qu.:   2666   1st Qu.:   2327
##  Median :  22382   Median : 21200   Median :  20089   Median :  19052
##  Mean   :  51223   Mean   : 49179   Mean   :  47013   Mean   :  43263
##  3rd Qu.:  67091   3rd Qu.: 64006   3rd Qu.:  60165   3rd Qu.:  54506
##  Max.   : 964511   Max.   :983931   Max.   :1664089   Max.   : 891586
##       V17              V18               V19              V20
##  Min.   :-81334   Min.   :-339603   Min.   :     0   Min.   :      0
##  1st Qu.:  1763   1st Qu.:   1256   1st Qu.:  1000   1st Qu.:    833
##  Median : 18105   Median :  17071   Median :  2100   Median :   2009
##  Mean   : 40311   Mean   :  38872   Mean   :  5664   Mean   :   5921
##  3rd Qu.: 50191   3rd Qu.:  49198   3rd Qu.:  5006   3rd Qu.:   5000
##  Max.   :927171   Max.   : 961664   Max.   :873552   Max.   :1684259
##       V21              V22              V23                V24
##  Min.   :     0   Min.   :     0   Min.   :     0.0   Min.   :     0.0
##  1st Qu.:   390   1st Qu.:   296   1st Qu.:   252.5   1st Qu.:   117.8
```

```
##  Median :  1800   Median :  1500   Median :  1500.0   Median :  1500.0
##  Mean   :  5226   Mean   :  4826   Mean   :  4799.4   Mean   :  5215.5
##  3rd Qu.:  4505   3rd Qu.:  4013   3rd Qu.:  4031.5   3rd Qu.:  4000.0
##  Max.   :896040   Max.   :621000   Max.   :426529.0   Max.   :528666.0
##       V25
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2212
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

Exploratory Data Analysis

We check our categorical variables Gender, Education and Marital Status

```
#Gender Table
table(data$V3)
```

```
##
##     1     2
## 11888 18112
```

```
#Education Table
table(data$V4)
```

```
##
##     0     1     2     3     4     5     6
##    14 10585 14030  4917   123   280    51
```

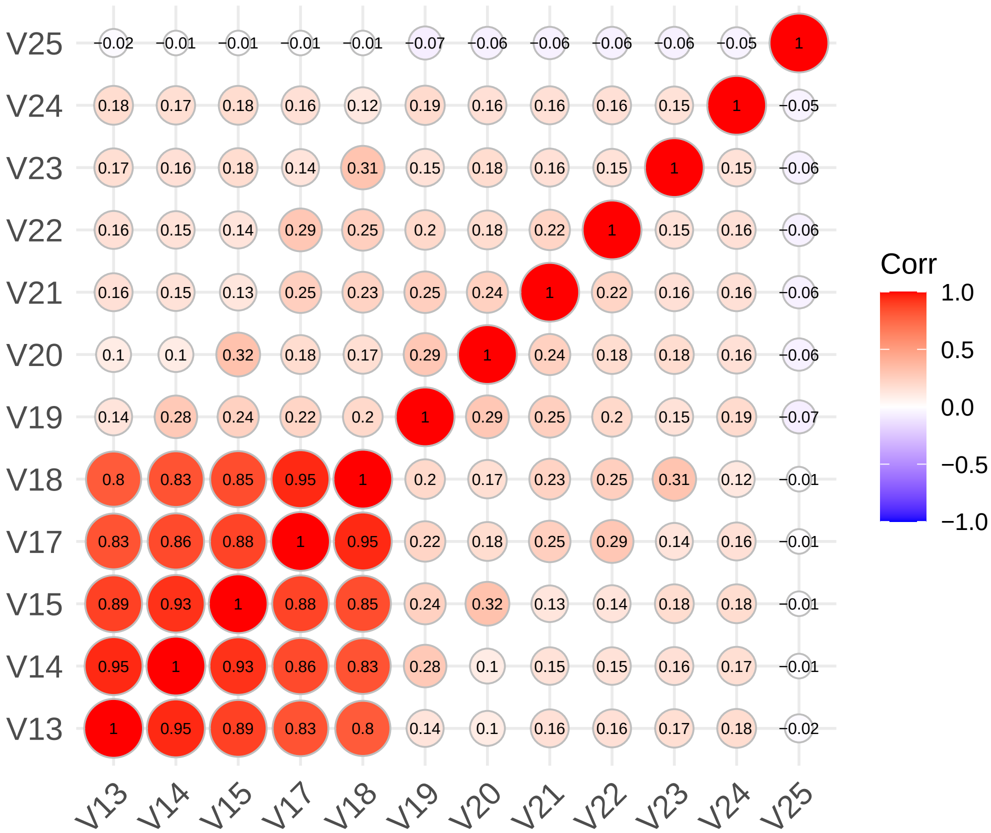
```
#Marital Status
table(data$V5)
```

```
##
##     0     1     2     3
##    54 13659 15964   323
```

We observe that Education has certain unknown observations that we can clump under the category "others". Similarly, we categorise certain unknown observations under "others" for Marital Status.

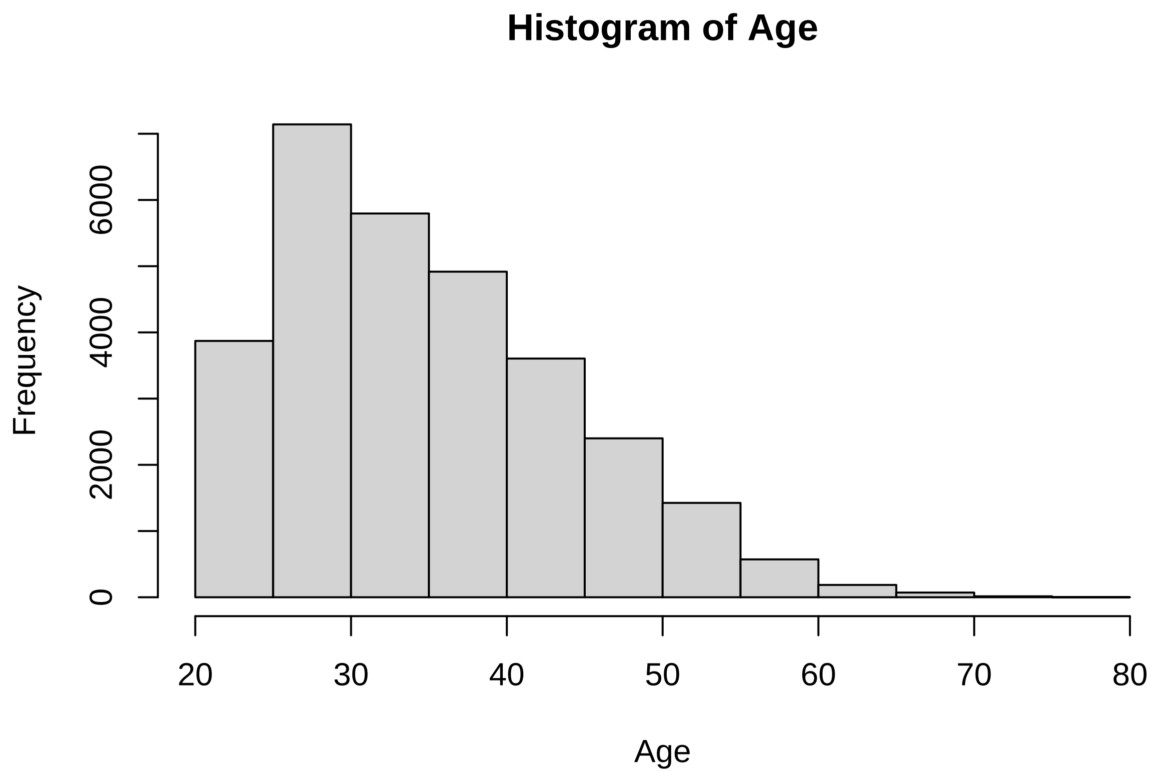
We check our continuous variables to check if any of them has strong explanatory power for our variable of interest V25 using a correlation matrix

```
#Correlation matrix
data_onlycat <- subset(data, select = c(c(V13,V14,V15,V17,V18,V19,V20,V21,V22,V23,V24,V25)))
ggcorrplot(cor(data_onlycat), method = "circle", lab = T, lab_size = 2)
```



We take a look at the distribution of ages in our dataset

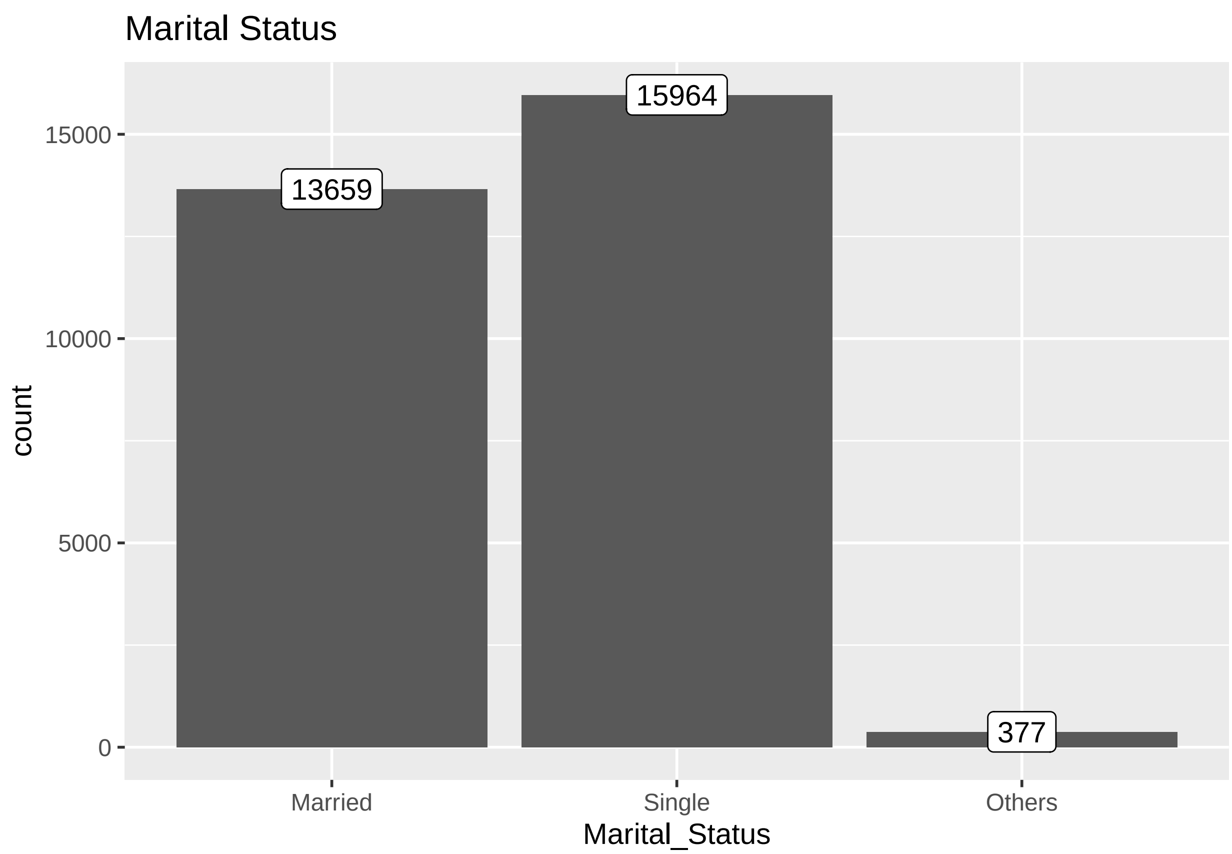
```
hist(data$V6, xlab="Age", main="Histogram of Age")
```



We also look at the distribution of marital statuses in our dataset

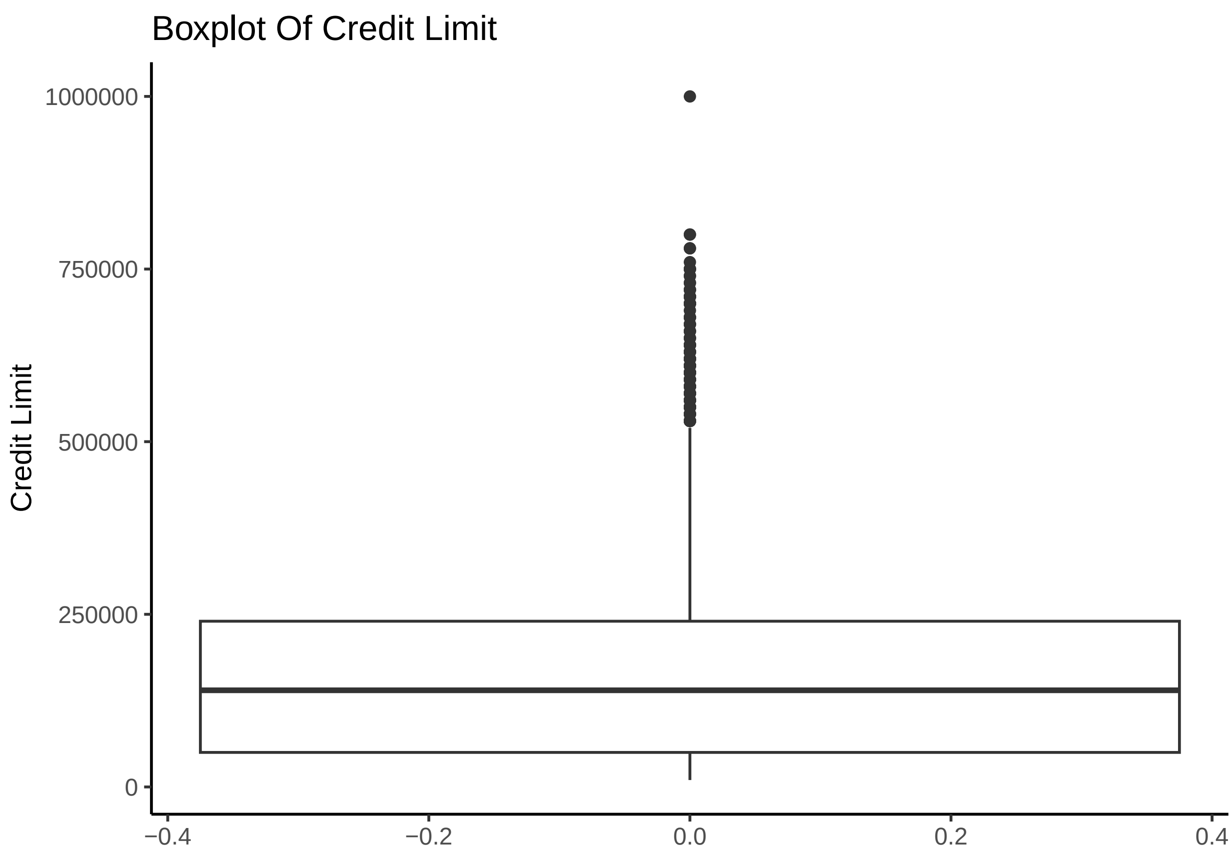
```
marital_status_plot <- ggplot(data_v, aes(x=Marital_Status))+
  geom_bar() +
  labs(title="Marital Status") +
  stat_count(aes(label = ..count..), geom = "label")

marital_status_plot
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
```



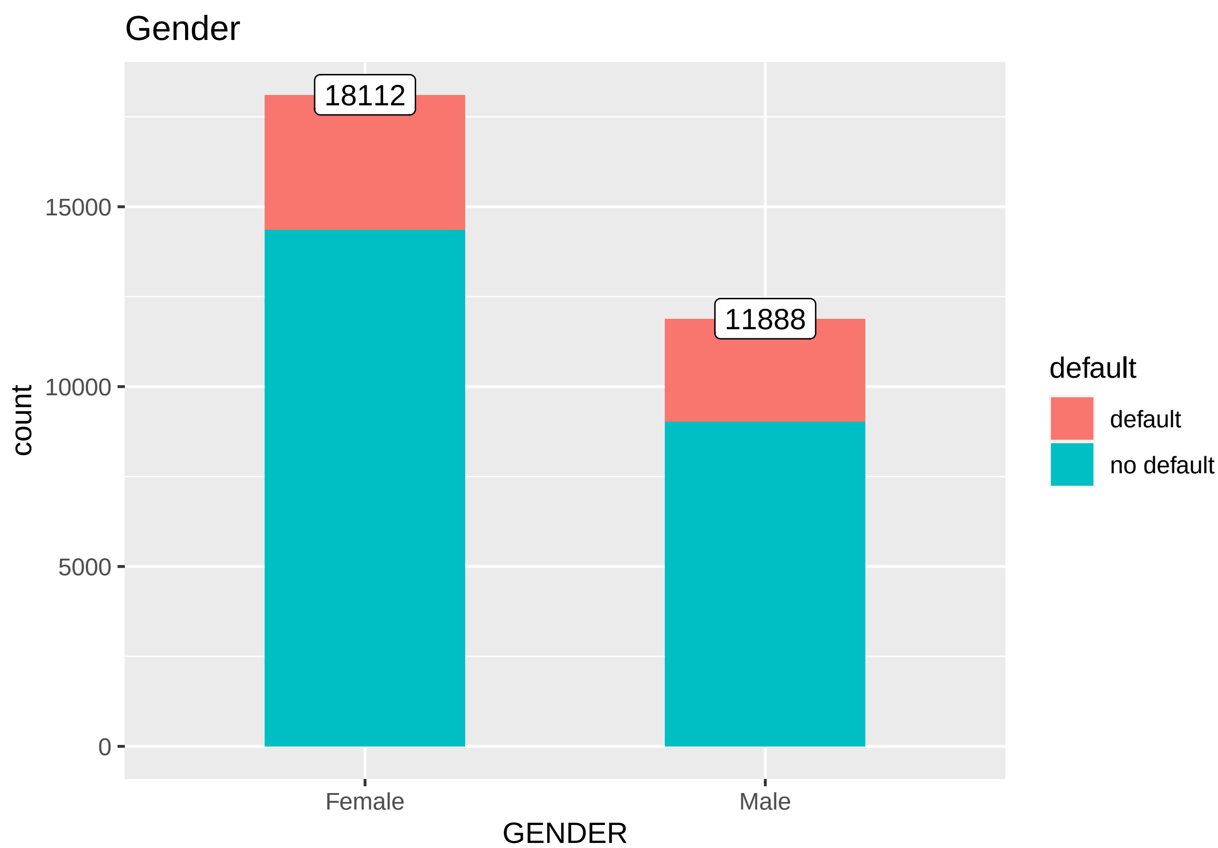
We check the boxplot of credit limit

```
credit_balance_plot <-ggplot(data_v, aes(x=V2), xlab="Credit Limit") + geom_boxplot() + coord_flip() + l
  theme_classic()

credit_balance_plot
```



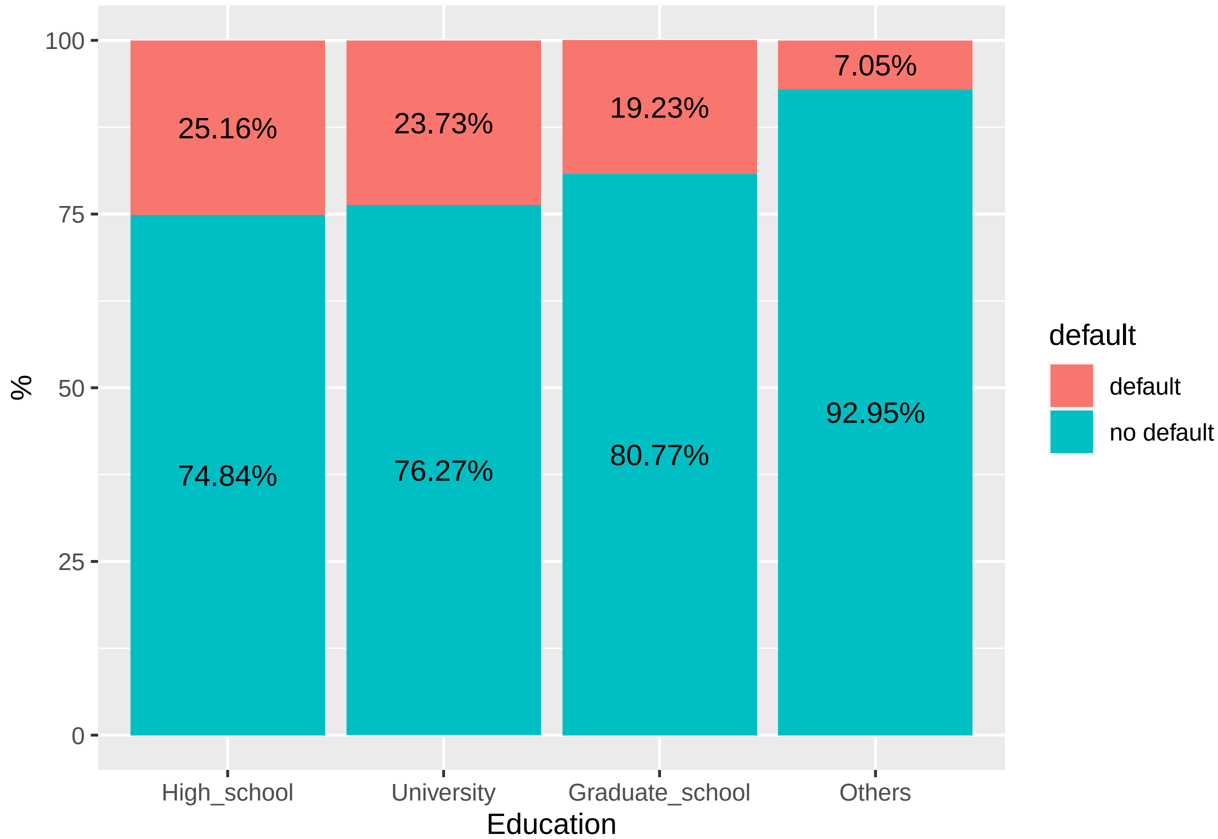
We check for our distribution of gender, as well as the proportion of those who defaulted.

```
data_v$default<- ifelse(data$V25 == 1, "default", "no default")

# Bar Graph for gender
gender_plot<- ggplot(data_v, aes(GENDER))+
  geom_bar(aes(fill=default), width = 0.5) +
  labs(title="Gender") +
  stat_count(aes(label = ..count..), geom = "label")

gender_plot
```



Similarly, we check for the distribution of Education level, and the proportion of those who defaulted

```
# Bar graph for Education
education_plot <- data_v %>%
  count(EDUCATION, default) %>%
    group_by(EDUCATION) %>%
    mutate(n = n/sum(n) * 100) %>%
    ggplot() +
    aes(factor(EDUCATION,
               levels = c("High_school", "University", "Graduate_school","Others")), n,
        fill = default, label = paste0(round(n, 2), "%")) +
    geom_col() +
    geom_text(position=position_stack(0.5))+
    xlab("Education")+
    ylab("%")

education_plot
```




```
set.seed(1234)
n = length(data$V1)
index <- 1:nrow(data)
testindex <- sample(index, trunc(n)/4)
test.data <- data[testindex,]
train.data <- data[-testindex,]
```

```
chistat <- matrix(0, 10, 2)
col  <- ncol(data) - 1
pred <- as.factor(data[,25])
for (i in 1:10) {
  x <- as.factor(data[,2+i])
  tbl <- table(x, pred)
  chi2res <- chisq.test(tbl)
  print(chi2res)
  chistat[i, 1] <- chi2res$statistic
  chistat[i, 2] <- chi2res$p.value
}
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tbl
## X-squared = 47.709, df = 1, p-value = 4.945e-12
##
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 160.41, df = 3, p-value < 2.2e-16
##
##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 28.13, df = 2, p-value = 7.791e-07

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 158.55, df = 55, p-value = 5.643e-12

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 5366, df = 10, p-value < 2.2e-16

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 3474.5, df = 10, p-value < 2.2e-16

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 2622.5, df = 10, p-value < 2.2e-16

## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 2341.5, df = 10, p-value < 2.2e-16
```

```
## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 2197.7, df = 9, p-value < 2.2e-16
```

```
## Warning in chisq.test(tbl): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl
## X-squared = 1886.8, df = 9, p-value < 2.2e-16
```

```
chistat <- chistat[-4,]
df <- data.frame(chistat[,1:2])
names(df) <- c("chi2 stat", "p-value")
df
```

```
##    chi2 stat      p-value
## 1   47.70880 4.944679e-12
## 2  160.40995 1.495065e-34
## 3   28.13032 7.790720e-07
## 4 5365.96498 0.000000e+00
## 5 3474.46679 0.000000e+00
## 6 2622.46213 0.000000e+00
## 7 2341.46995 0.000000e+00
## 8 2197.69490 0.000000e+00
## 9 1886.83531 0.000000e+00
```

```
log_model <- glm(V25 ~., data = train.data, family = "binomial")
summary(log_model)
```

```
##
## Call:
## glm(formula = V25 ~ ., family = "binomial", data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1585  -0.7014  -0.5425  -0.2797   3.7000
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.431e-01  1.039e-01  -9.077  < 2e-16 ***
## V1          -9.386e-07  2.025e-06  -0.464 0.642990
## V2          -6.510e-07  1.828e-07  -3.562 0.000368 ***
## V32         -1.095e-01  3.553e-02  -3.082 0.002055 **
## V42         -8.476e-02  4.119e-02  -2.058 0.039597 *
## V43         -8.364e-02  5.482e-02  -1.526 0.127067
## V44         -1.096e+00  2.132e-01  -5.141 2.73e-07 ***
```

```
## V52         -1.980e-01  3.994e-02  -4.958 7.12e-07 ***
## V53         -2.344e-01  1.602e-01  -1.464 0.143286
## V6           4.330e-03  2.144e-03   2.020 0.043386 *
## V7           5.886e-01  2.053e-02  28.664  < 2e-16 ***
## V8           8.608e-02  2.338e-02   3.683 0.000231 ***
## V9           6.624e-02  2.617e-02   2.531 0.011369 *
## V10          1.777e-02  2.896e-02   0.614 0.539517
## V11          3.619e-02  3.096e-02   1.169 0.242504
## V12          1.700e-02  2.578e-02   0.660 0.509489
## V13         -5.262e-06  1.304e-06  -4.037 5.41e-05 ***
## V14          2.637e-06  1.711e-06   1.542 0.123163
## V15          1.101e-06  1.539e-06   0.715 0.474445
## V16          7.538e-07  1.564e-06   0.482 0.629845
## V17          5.545e-07  1.701e-06   0.326 0.744433
## V18         -9.087e-07  1.340e-06  -0.678 0.497812
## V19         -1.175e-05  2.522e-06  -4.660 3.17e-06 ***
## V20         -1.418e-05  2.806e-06  -5.055 4.31e-07 ***
## V21         -1.535e-06  1.904e-06  -0.806 0.420092
## V22         -2.530e-06  1.941e-06  -1.303 0.192496
## V23         -3.186e-06  2.054e-06  -1.551 0.120863
## V24         -3.303e-06  1.602e-06  -2.062 0.039172 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 23756  on 22499  degrees of freedom
## Residual deviance: 20809  on 22472  degrees of freedom
## AIC: 20865
##
## Number of Fisher Scoring iterations: 6
```

```r
#V1, V10-12, V14-18, V21-23
better_log_model <- glm(V25~V2+V3+V4+V5+V6+V7+V8+V9+V13+V19+V20+V24, data = train.data, family = "binomi
summary(better_log_model)
```

```
##
## Call:
## glm(formula = V25 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V13 +
##     V19 + V20 + V24, family = "binomial", data = train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1834  -0.6977  -0.5444  -0.2897   3.5810
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.759e-01  9.962e-02  -9.797  < 2e-16 ***
## V2          -8.007e-07  1.782e-07  -4.495 6.97e-06 ***
## V32         -1.063e-01  3.547e-02  -2.997  0.00273 **
## V42         -8.674e-02  4.111e-02  -2.110  0.03485 *
## V43         -8.768e-02  5.470e-02  -1.603  0.10896
## V44         -1.109e+00  2.129e-01  -5.210 1.89e-07 ***
## V52         -1.998e-01  3.989e-02  -5.009 5.47e-07 ***
```

```
## V53         -2.503e-01  1.599e-01  -1.565  0.11757
## V6           4.427e-03  2.142e-03   2.067  0.03872 *
## V7           6.033e-01  2.032e-02  29.690  < 2e-16 ***
## V8           8.918e-02  2.306e-02   3.867  0.00011 ***
## V9           1.127e-01  2.129e-02   5.291 1.22e-07 ***
## V13         -1.654e-06  3.066e-07  -5.396 6.82e-08 ***
## V19         -9.420e-06  2.250e-06  -4.187 2.83e-05 ***
## V20         -1.265e-05  2.476e-06  -5.110 3.22e-07 ***
## V24         -3.864e-06  1.589e-06  -2.432  0.01502 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 23756  on 22499  degrees of freedom
## Residual deviance: 20844  on 22484  degrees of freedom
## AIC: 20876
##
## Number of Fisher Scoring iterations: 6
```

```r
predict <- predict(better_log_model, test.data, type = "response")
predict <- ifelse(predict >0.5, 1, 0)
table <- table(test.data$V25, predict)
table
```

```
##    predict
##        0    1
##   0 5639  193
##   1 1270  398
```

```r
class_acc1 <- table[1,1]/(table[1,1] + table[1,2])
class_acc2 <- table[2,1]/(table[2,1] + table[2,2])
(class_acc1 + class_acc2)/2 #average class accuracy
```

```
## [1] 0.8641488
```

```r
1/((1/class_acc1)+(1/class_acc2)) #harmonic mean
```

```
## [1] 0.4259648
```

```r
acc <- mean(predict == test.data$V25) #accuracy
roc <- auc(test.data$V25, predict) #area under roc curve
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
specificity <- table[2,2]/(table[2,1] + table[2,2]) #specificity
recall <- table[1,1]/(table[1,1] + table[1,2])
precision <- table[1,1]/(table[1,1] + table[2,1])
```

```r
F1 <- 2*recall*precision/(recall + precision) #F1 statistic
log_metrics <- c(acc, specificity, roc, F1)
metrics <- t(as.data.frame(log_metrics))
`colnames<-`(metrics, c("Accuracy", "Specificity", "Area under ROC Curve", "F1 Statistic"))
```

```
##              Accuracy Specificity Area under ROC Curve F1 Statistic
## log_metrics 0.8049333   0.2386091            0.6027579    0.8851738
```

```r
model <- lm(V25~., data = train.data)
summary(model)
```

```
##
## Call:
## lm(formula = V25 ~ ., data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29260 -0.24154 -0.16041  0.03751  1.29635
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.763e-01  1.566e-02  17.649  < 2e-16 ***
## V1           4.589e-09  3.009e-07   0.015 0.987829
## V2          -6.974e-08  2.519e-08  -2.768 0.005641 **
## V32         -1.442e-02  5.352e-03  -2.695 0.007038 **
## V42         -1.364e-02  6.068e-03  -2.249 0.024547 *
## V43         -1.265e-02  8.270e-03  -1.529 0.126165
## V44         -1.099e-01  2.107e-02  -5.213 1.87e-07 ***
## V52         -3.082e-02  5.968e-03  -5.164 2.44e-07 ***
## V53         -4.109e-02  2.432e-02  -1.689 0.091144 .
## V6           9.214e-04  3.289e-04   2.801 0.005097 **
## V7           9.748e-02  3.201e-03  30.454  < 2e-16 ***
## V8           1.924e-02  3.852e-03   4.996 5.89e-07 ***
## V9           1.211e-02  4.119e-03   2.941 0.003270 **
## V10          5.582e-04  4.573e-03   0.122 0.902837
## V11          6.200e-03  4.953e-03   1.252 0.210673
## V12          2.638e-03  4.094e-03   0.644 0.519314
## V13         -6.395e-07  1.331e-07  -4.806 1.55e-06 ***
## V14          1.730e-07  1.837e-07   0.942 0.346425
## V15          1.194e-07  1.747e-07   0.683 0.494419
## V16         -2.437e-08  1.819e-07  -0.134 0.893385
## V17         -2.594e-08  2.112e-07  -0.123 0.902273
## V18         -2.932e-08  1.668e-07  -0.176 0.860426
## V19         -7.821e-07  2.078e-07  -3.765 0.000167 ***
## V20         -3.795e-07  1.681e-07  -2.257 0.024003 *
## V21          3.193e-08  1.946e-07   0.164 0.869689
## V22         -2.202e-07  2.083e-07  -1.057 0.290544
## V23         -2.795e-07  2.196e-07  -1.272 0.203268
## V24         -1.961e-07  1.603e-07  -1.224 0.221061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.3879 on 22472 degrees of freedom
## Multiple R-squared:  0.1264, Adjusted R-squared:  0.1253
## F-statistic: 120.4 on 27 and 22472 DF,  p-value: < 2.2e-16
```

```r
#V1, V10-12, V14-18, V21-24
better_model <- lm(V25~V2+V3+V4+V5+V6+V7+V8+V9+V13+V19+V20, data = train.data)
summary(better_model)
```

```
##
## Call:
## lm(formula = V25 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V13 +
##     V19 + V20, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3019 -0.2405 -0.1598  0.0360  1.2455
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.751e-01  1.505e-02  18.281  < 2e-16 ***
## V2          -8.997e-08  2.436e-08  -3.694 0.000222 ***
## V32         -1.421e-02  5.347e-03  -2.658 0.007871 **
## V42         -1.366e-02  6.064e-03  -2.253 0.024293 *
## V43         -1.277e-02  8.264e-03  -1.545 0.122435
## V44         -1.107e-01  2.103e-02  -5.262 1.44e-07 ***
## V52         -3.091e-02  5.966e-03  -5.182 2.22e-07 ***
## V53         -4.229e-02  2.432e-02  -1.739 0.082036 .
## V6           9.265e-04  3.289e-04   2.817 0.004846 **
## V7           9.903e-02  3.166e-03  31.278  < 2e-16 ***
## V8           2.012e-02  3.813e-03   5.277 1.33e-07 ***
## V9           1.708e-02  3.444e-03   4.958 7.19e-07 ***
## V13         -4.473e-07  3.968e-08 -11.275  < 2e-16 ***
## V19         -6.925e-07  1.729e-07  -4.005 6.22e-05 ***
## V20         -3.427e-07  1.164e-07  -2.943 0.003255 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.388 on 22485 degrees of freedom
## Multiple R-squared:  0.1257, Adjusted R-squared:  0.1251
## F-statistic: 230.8 on 14 and 22485 DF,  p-value: < 2.2e-16
```

```r
predict <- predict(better_model, test.data, type = "response")
predict <- ifelse(predict >0.5, 1, 0)
table <- table(test.data$V25, predict)
table
```

```
##    predict
##        0    1
##   0 5724  108
##   1 1418  250
```

```r
class_acc1 <- table[1,1]/(table[1,1] + table[1,2])
class_acc2 <- table[2,1]/(table[2,1] + table[2,2])
(class_acc1 + class_acc2)/2 #average class accuracy
```

```
## [1] 0.9158007
```

```r
1/((1/class_acc1)+(1/class_acc2)) #harmonic mean
```

```
## [1] 0.455545
```

```r
acc <- mean(predict == test.data$V25) #accuracy
roc <- auc(test.data$V25, predict) #area under roc curve
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

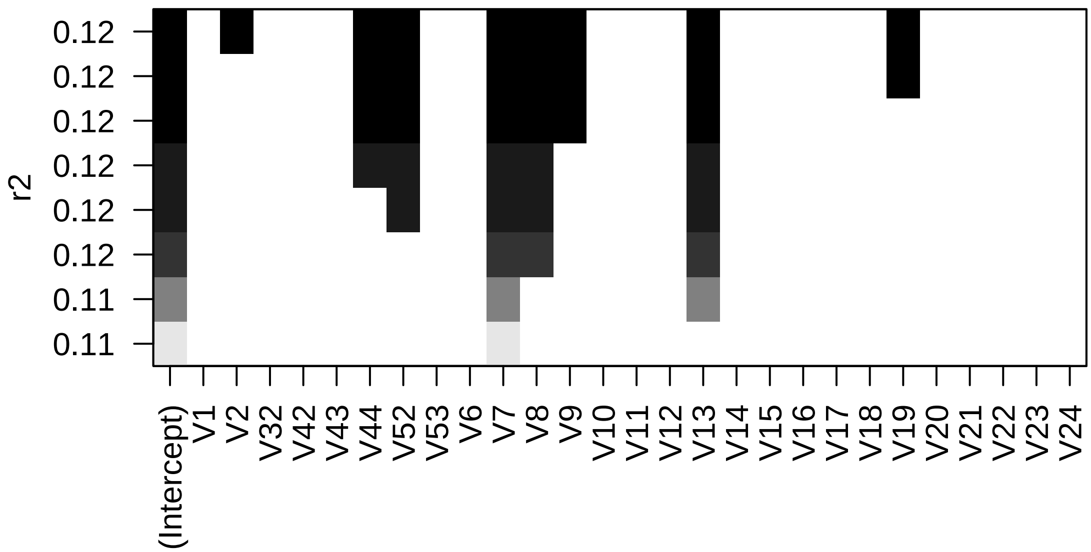
```r
specificity <- table[2,2]/(table[2,1] + table[2,2]) #specificity
recall <- table[1,1]/(table[1,1] + table[1,2])
precision <- table[1,1]/(table[1,1] + table[2,1])
F1 <- 2*recall*precision/(recall + precision) #F1 statistic
lm_metrics <- c(acc, specificity, roc, F1)
metrics <- rbind(metrics, lm_metrics)
```

```r
outforward <- regsubsets(data$V25 ~., data = data, method = "forward")
summary(outforward)
```

```
## Subset selection object
## Call: regsubsets.formula(data$V25 ~ ., data = data, method = "forward")
## 27 Variables  (and intercept)
##     Forced in Forced out
## V1      FALSE      FALSE
## V2      FALSE      FALSE
## V32     FALSE      FALSE
## V42     FALSE      FALSE
## V43     FALSE      FALSE
## V44     FALSE      FALSE
## V52     FALSE      FALSE
## V53     FALSE      FALSE
## V6      FALSE      FALSE
## V7      FALSE      FALSE
## V8      FALSE      FALSE
## V9      FALSE      FALSE
## V10     FALSE      FALSE
## V11     FALSE      FALSE
## V12     FALSE      FALSE
## V13     FALSE      FALSE
## V14     FALSE      FALSE
## V15     FALSE      FALSE
## V16     FALSE      FALSE
```

```
## V17     FALSE      FALSE
## V18     FALSE      FALSE
## V19     FALSE      FALSE
## V20     FALSE      FALSE
## V21     FALSE      FALSE
## V22     FALSE      FALSE
## V23     FALSE      FALSE
## V24     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          V1  V2  V32 V42 V43 V44 V52 V53 V6  V7  V8  V9  V10 V11 V12 V13 V14
## 1  ( 1 ) " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " "
## 2  ( 1 ) " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " "*" " "
## 3  ( 1 ) " " " " " " " " " " " " " " " " " " "*" "*" " " " " " " " " "*" " "
## 4  ( 1 ) " " " " " " " " " " " " "*" " " " " "*" "*" " " " " " " " " "*" " "
## 5  ( 1 ) " " " " " " " " " " "*" "*" " " " " "*" "*" " " " " " " " " "*" " "
## 6  ( 1 ) " " " " " " " " " " "*" "*" " " " " "*" "*" "*" " " " " " " "*" " "
## 7  ( 1 ) " " " " " " " " " " "*" "*" " " " " "*" "*" "*" " " " " " " "*" " "
## 8  ( 1 ) " " "*" " " " " " " "*" "*" " " " " "*" "*" "*" " " " " " " "*" " "
##          V15 V16 V17 V18 V19 V20 V21 V22 V23 V24
## 1  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 2  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 3  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 4  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 5  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 6  ( 1 ) " " " " " " " " " " " " " " " " " " " "
## 7  ( 1 ) " " " " " " " " "*" " " " " " " " " " "
## 8  ( 1 ) " " " " " " " " "*" " " " " " " " " " "
```

```
plot(outforward, scale="r2")
```




```
# inbackward <- regsubsets(data$V25 ~., data = data, method="backward")
# summary(inbackward)
# plot(inbackward, scale="r2")

#V7, V13, V8, V6, V9, V19, V5, V2
```

```
svm.model.feature.selection <- svm(V25 ~ V7 + V13 + V8,
                                   data = train.data, type="C-classification",
                                   kernel="linear", cost=0.2,
                                   class.weights=c("0"=0.185, "1"=0.815))
#result_train_feature_selection <- predict(svm.model.feature.selection, train.data[,-25])
result_test_feature_selection <- predict(svm.model.feature.selection, test.data)
#table(pred=result_train_feature_selection, actual=train.data$V25)
table(pred=result_test_feature_selection, actual=test.data$V25)
```

```
##     actual
## pred    0    1
##    0 4972  813
##    1  860  855
```

```
#mean(result_train_feature_selection == train.data$V25)
mean(result_test_feature_selection == test.data$V25)
```

```
## [1] 0.7769333
```

```r
tablesvm <- table(test.data$V25, result_test_feature_selection)
tablesvm
```

```
##    result_test_feature_selection
##        0    1
##   0 4972  860
##   1  813  855
```

```r
class_accsvm1 <- tablesvm[1,1]/(tablesvm[1,1] + tablesvm[1,2])
class_accsvm2 <- tablesvm[2,1]/(tablesvm[2,1] + tablesvm[2,2])
(class_accsvm1 + class_accsvm2)/2 #average class accuracy
```

```
## [1] 0.6699739
```

```r
1/((1/class_accsvm1)+(1/class_accsvm2)) #harmonic mean
```

```
## [1] 0.3101132
```

```r
accsvm <- mean(result_test_feature_selection == test.data$V25) #accuracy
rocsvm <- auc(test.data$V25, as.numeric(result_test_feature_selection)) #area under roc curve
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
specificitysvm <- tablesvm[2,2]/(tablesvm[2,1] + tablesvm[2,2]) #specificity
recallsvm <- tablesvm[1,1]/(tablesvm[1,1] + tablesvm[1,2])
precisionsvm <- tablesvm[1,1]/(tablesvm[1,1] + tablesvm[2,1])
F1svm <- 2*recallsvm*precisionsvm/(recallsvm + precisionsvm) #F1 statistic
svm_metrics <- c(accsvm, specificitysvm, rocsvm, F1svm)
metrics <- rbind(metrics, svm_metrics)
#V2,3,4,5,6,7,8,9,13,19,20,22
# With cost=0.2, (0.2,0.8) (test)
#     actual
# pred    0    1
#    0 5065  758
#    1  835  842
# 0.7876
```

```r
#BEFORE BALANCING
#TRAIN
nn <- nnet(V25 ~ V7 + V13 + V8, train.data,maxit=1000,size=8,entropy=TRUE, decay = 0.01)
```

```
## # weights:  41
## initial  value 19093.396067
## iter  10 value 11888.695171
## iter  20 value 11658.638695
## iter  30 value 10402.481801
## iter  40 value 10180.270955
## iter  50 value 10154.841735
```

```
## iter  60 value 10136.501960
## iter  70 value 10123.554681
## iter  80 value 10122.559034
## iter  90 value 10109.253948
## iter 100 value 10039.985075
## iter 110 value 10000.072144
## iter 120 value 9988.831389
## iter 130 value 9984.403142
## iter 140 value 9980.029598
## iter 150 value 9976.220718
## iter 150 value 9976.220659
## final  value 9976.220659
## converged
```

```r
#pred <- predict(nn, data = train.data)
#train.binpred <- predict(nn, train.data[,1:24], type = c("class"))
#mean(train.data$V25 == train.binpred)
#auc(train.data$V25, pred)

#TEST
#predtest <- predict(nn, data = test.data)
test.binpred <- predict(nn, test.data, type = c("class"))
#mean(test.data$V25 == test.binpred)
#auc(test.data$V25, predtest)

tablenn <- table(test.data$V25, test.binpred)
tablenn
```

```
##    test.binpred
##        0    1
##   0 5529  303
##   1 1087  581
```

```r
class_accnn1 <- tablenn[1,1]/(tablenn[1,1] + tablenn[1,2])
class_accnn2 <- tablenn[2,1]/(tablenn[2,1] + tablenn[2,2])
(class_accnn1 + class_accnn2)/2 #average class accuracy
```

```
## [1] 0.799862
```

```r
1/((1/class_accnn1)+(1/class_accnn2)) #harmonic mean
```

```
## [1] 0.3862047
```

```r
accnn <- mean(test.binpred == test.data$V25) #accuracy
rocnn <- auc(test.data$V25, as.numeric(test.binpred)) #area under roc curve
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
specificitynn <- tablenn[2,2]/(tablenn[2,1] + tablenn[2,2]) #specificity
recallnn <- tablenn[1,1]/(tablenn[1,1] + tablenn[1,2])
precisionnn <- tablenn[1,1]/(tablenn[1,1] + tablenn[2,1])
F1nn <- 2*recallnn*precisionnn/(recallnn + precisionnn) #F1 statistic
nn_metrics <- c(accnn, specificitynn, rocnn, F1nn)
metrics <- rbind(metrics, nn_metrics)
```

```r
#BEFORE BALANCING
tree.model<- ctree(as.factor(V25) ~ V7 + V13 + V8, train.data)
tree.predict_test <-predict(tree.model, test.data)

tabletree <- table(test.data$V25, tree.predict_test)
class_acctree1 <- tabletree[1,1]/(tabletree[1,1] + tabletree[1,2])
class_acctree2 <- tabletree[2,1]/(tabletree[2,1] + tabletree[2,2])
(class_acctree1 + class_acctree2)/2 #average class accuracy
```

```
## [1] 0.8066294
```

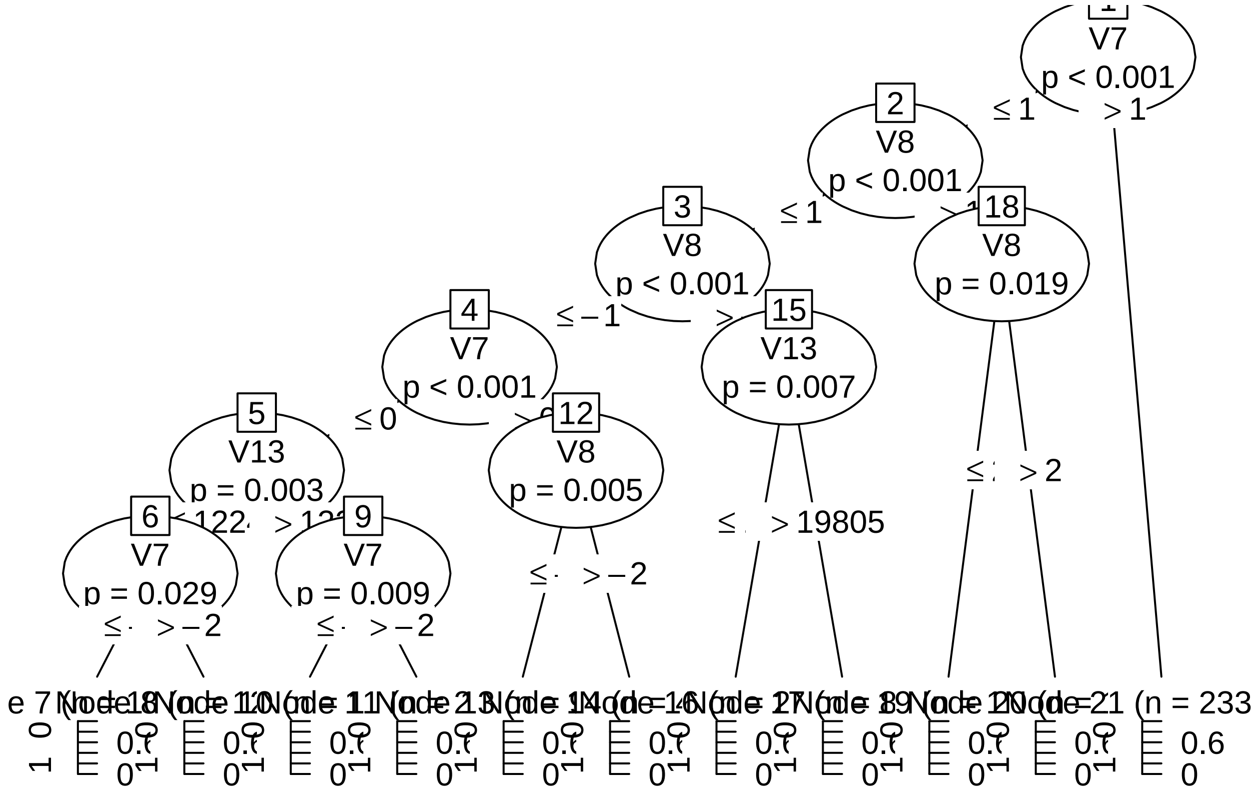
```r
1/((1/class_acctree1)+(1/class_acctree2)) #harmonic mean
```

```
## [1] 0.3904327
```

```r
acctree <- mean(tree.predict_test == test.data$V25) #accuracy
roctree <- auc(test.data$V25, as.numeric(tree.predict_test)) #area under roc curve
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
specificitytree <- tabletree[2,2]/(tabletree[2,1] + tabletree[2,2]) #specificity
recalltree <- tabletree[1,1]/(tabletree[1,1] + tabletree[1,2])
precisiontree <- tabletree[1,1]/(tabletree[1,1] + tabletree[2,1])
F1tree <- 2*recalltree*precisiontree/(recalltree + precisiontree) #F1 statistic
tree_metrics <- c(acctree, specificitytree, roctree, F1tree)
metrics <- rbind(metrics, tree_metrics)
plot(tree.model)
```




```r
row.names(metrics) <- c("Linear Regression", "Logistic Regression", "Support Vector Machine","Neural Ne

colnames(metrics) <- c("Accuracy", "Specificity", "Area under ROC Curve", "F1 Statistic")
```

```r
#OVERSAMPLING
oversampled_train_data <- ovun.sample(V25 ~ ., data = train.data, method = "over",
                                      N = 2*nrow(subset(train.data, train.data$V25 == 0)))$data

table(oversampled_train_data$V25)
```

```
##
##     0     1
## 17532 17532
```

```r
svm.model.feature.selection.balanced <- svm(as.factor(V25) ~ V7 + V13 + V8,
                                    data = oversampled_train_data, type="C-classification",
                                    kernel="linear", cost=0.2)
result_train_feature_selection_balanced <- predict(svm.model.feature.selection.balanced, oversampled_tr
result_test_feature_selection_balanced <- predict(svm.model.feature.selection.balanced, test.data[,-25]
table(pred=result_train_feature_selection_balanced, actual=oversampled_train_data$V25)
```

```
##     actual
## pred     0     1
##    0 15065  8561
##    1  2467  8971
```

```r
table(pred=result_test_feature_selection_balanced, actual=test.data$V25)
```

```
##       actual
## pred      0    1
##      0 4978  823
##      1  854  845
```

```r
mean(result_train_feature_selection_balanced == oversampled_train_data$V25)
```

```
## [1] 0.6854894
```

```r
mean(result_test_feature_selection_balanced == test.data$V25)
```

```
## [1] 0.7764
```

```r
log_model_balanced <- glm(V25 ~., data = oversampled_train_data, family = "binomial")
summary(log_model_balanced)
```

```
##
## Call:
## glm(formula = V25 ~ ., family = "binomial", data = oversampled_train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3515  -1.0718   0.0404   1.0517   3.5526
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.120e-01  7.002e-02   3.028 0.002465 **
## V1          -6.108e-07  1.363e-06  -0.448 0.654152
## V2          -5.953e-07  1.199e-07  -4.964 6.92e-07 ***
## V32         -1.193e-01  2.390e-02  -4.993 5.96e-07 ***
## V42         -6.925e-02  2.745e-02  -2.523 0.011645 *
## V43         -7.779e-02  3.685e-02  -2.111 0.034750 *
## V44         -9.922e-01  1.247e-01  -7.959 1.73e-15 ***
## V52         -1.839e-01  2.692e-02  -6.830 8.50e-12 ***
## V53         -2.419e-01  1.076e-01  -2.249 0.024490 *
## V6           6.613e-03  1.457e-03   4.540 5.62e-06 ***
## V7           5.245e-01  1.331e-02  39.410  < 2e-16 ***
## V8           7.101e-02  1.557e-02   4.561 5.09e-06 ***
## V9           7.101e-02  1.709e-02   4.154 3.26e-05 ***
## V10          2.403e-02  1.885e-02   1.275 0.202428
## V11          2.673e-02  2.053e-02   1.302 0.192813
## V12         -6.718e-03  1.711e-02  -0.393 0.694564
## V13         -7.005e-06  8.224e-07  -8.518  < 2e-16 ***
## V14          5.143e-06  1.050e-06   4.899 9.62e-07 ***
## V15         -4.970e-08  9.486e-07  -0.052 0.958212
## V16          1.344e-06  9.811e-07   1.370 0.170811
## V17          3.396e-07  1.057e-06   0.321 0.747939
## V18         -9.930e-07  8.349e-07  -1.189 0.234282
## V19         -1.475e-05  1.522e-06  -9.687  < 2e-16 ***
```

```
## V20         -9.886e-06  1.424e-06  -6.941 3.88e-12 ***
## V21         -2.321e-06  1.121e-06  -2.070 0.038415 *
## V22         -1.316e-06  1.118e-06  -1.178 0.238961
## V23         -3.637e-06  1.246e-06  -2.920 0.003502 **
## V24         -3.239e-06  9.592e-07  -3.377 0.000733 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 48609  on 35063  degrees of freedom
## Residual deviance: 42708  on 35036  degrees of freedom
## AIC: 42764
##
## Number of Fisher Scoring iterations: 5
```

```r
predictlr_balanced <- predict(log_model_balanced, test.data, type = "response")
predictlr_balanced <- ifelse(predictlr_balanced >0.5, 1, 0)
tablelr <- table(test.data$V25, predictlr_balanced)
tablelr
```

```
##    predictlr_balanced
##        0    1
##   0 4108 1724
##   1  603 1065
```

```r
class_acc1 <- tablelr[1,1]/(tablelr[1,1] + tablelr[1,2])
class_acc2 <- tablelr[2,1]/(tablelr[2,1] + tablelr[2,2])
(class_acc1 + class_acc2)/2 #average class accuracy
```

```
## [1] 0.5329502
```

```r
1/((1/class_acc1)+(1/class_acc2)) #harmonic mean
```

```
## [1] 0.2389008
```

```r
mean(predictlr_balanced == test.data$V25) #accuracy
```

```
## [1] 0.6897333
```

```r
auc(test.data$V25, predictlr_balanced) #area under roc curve
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Area under the curve: 0.6714
```

```r
tablelr[2,2]/(tablelr[2,1] + tablelr[2,2]) #specificity
```

```
## [1] 0.6384892
```

```r
recalllr <- tablelr[1,1]/(tablelr[1,1] + tablelr[1,2])
precisionlr <- tablelr[1,1]/(tablelr[1,1] + tablelr[2,1])
F1lr <- 2*recalllr*precisionlr/(recalllr + precisionlr) #F1 statistic
```

```r
better_model_balanced <- lm(V25~V2+V3+V4+V5+V6+V7+V8+V9+V13+V19+V20, data = oversampled_train_data)
summary(better_model_balanced)
```

```
##
## Call:
## lm(formula = V25 ~ V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V13 +
##     V19 + V20, data = oversampled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.66376 -0.44291 -0.05205  0.44338  1.60198
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.381e-01  1.425e-02  37.754  < 2e-16 ***
## V2          -2.151e-07  2.439e-08  -8.822  < 2e-16 ***
## V32         -2.375e-02  5.067e-03  -4.688 2.77e-06 ***
## V42         -1.405e-02  5.834e-03  -2.409   0.0160 *
## V43         -1.725e-02  7.808e-03  -2.209   0.0272 *
## V44         -2.026e-01  2.283e-02  -8.872  < 2e-16 ***
## V52         -3.999e-02  5.697e-03  -7.018 2.29e-12 ***
## V53         -5.496e-02  2.304e-02  -2.386   0.0171 *
## V6           1.412e-03  3.075e-04   4.592 4.41e-06 ***
## V7           1.133e-01  2.690e-03  42.116  < 2e-16 ***
## V8           1.298e-02  3.286e-03   3.949 7.86e-05 ***
## V9           2.091e-02  2.966e-03   7.052 1.80e-12 ***
## V13         -4.496e-07  3.813e-08 -11.792  < 2e-16 ***
## V19         -1.672e-06  1.891e-07  -8.844  < 2e-16 ***
## V20         -8.187e-07  1.313e-07  -6.238 4.49e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4616 on 35049 degrees of freedom
## Multiple R-squared:  0.1481, Adjusted R-squared:  0.1477
## F-statistic: 435.1 on 14 and 35049 DF,  p-value: < 2.2e-16
```

```r
predict_balanced <- predict(better_model_balanced, test.data, type = "response")
predict_balanced <- ifelse(predict_balanced >0.5, 1, 0)
table(test.data$V25, predict_balanced)
```

```
##    predict_balanced
##        0    1
##   0 4231 1601
##   1  614 1054
```

```
mean(predict_balanced == test.data$V25)
```

```
## [1] 0.7046667
```

```
predictlm <- predict(better_model_balanced, oversampled_train_data, type = "response")
predictlm <- ifelse(predictlm >0.5, 1, 0)
table(oversampled_train_data$V25, predictlm)
```

```
##    predictlm
##         0     1
##   0 12551  4981
##   1  6391 11141
```

```
mean(predictlm == oversampled_train_data$V25)
```

```
## [1] 0.6756788
```

```
#AFTER BALANCING OVERSAMPLING
#TRAIN
tree.model_oversample<- ctree(as.factor(V25) ~ V7 + V13 + V8, oversampled_train_data)
tree.binpredict_oversample <-predict(tree.model_oversample, test.data)

tabletreebal <- table(test.data$V25, tree.binpredict_oversample)
class_acctreebal1 <- tabletreebal[1,1]/(tabletreebal[1,1] + tabletreebal[1,2])
class_acctreebal2 <- tabletreebal[2,1]/(tabletreebal[2,1] + tabletreebal[2,2])
(class_acctreebal1 + class_acctreebal2)/2 #average class accuracy
```

```
## [1] 0.6046379
```

```
1/((1/class_acctreebal1)+(1/class_acctreebal2)) #harmonic mean
```

```
## [1] 0.2732866
```

```
mean(tree.binpredict_oversample == test.data$V25) #accuracy
```

```
## [1] 0.7454667
```

```
auc(test.data$V25, as.numeric(tree.binpredict_oversample)) #area under roc curve
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## Area under the curve: 0.6874
```

```
tabletreebal[2,2]/(tabletreebal[2,1] + tabletreebal[2,2]) #specificity
```

```
## [1] 0.5827338
```

```
recalltreebal <- tabletreebal[1,1]/(tabletreebal[1,1] + tabletreebal[1,2])
recalltreebal
```

```
## [1] 0.7920096
```

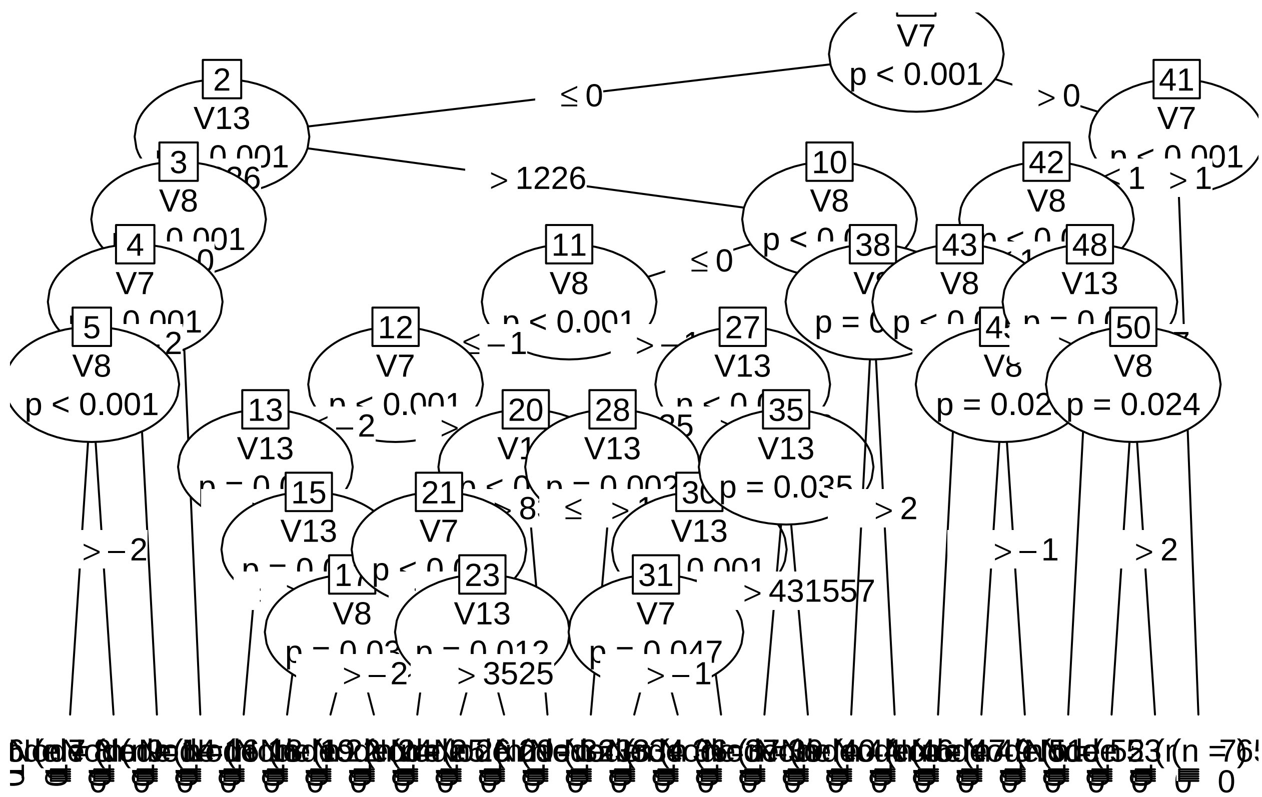
```
precisiontreebal <- tabletreebal[1,1]/(tabletreebal[1,1] + tabletreebal[2,1])
precisiontreebal
```

```
## [1] 0.8690499
```

```
F1treebal <- 2*recalltreebal*precisiontreebal/(recalltreebal + precisiontreebal) #F1 statistic
F1treebal
```

```
## [1] 0.8287432
```

```
plot(tree.model_oversample)
```



```r
#OVERSAMPLE
#TRAIN
nn_oversample <- nnet(oversampled_train_data$V25~., oversampled_train_data[,1:24],maxit=2000,size=20,en
```

```
## # weights:  581
## initial  value 25909.469891
## iter  10 value 24059.410038
## iter  20 value 23970.951888
## iter  30 value 23892.400356
## iter  40 value 23861.121547
## iter  50 value 23843.126543
## iter  60 value 23837.443167
## iter  70 value 23821.329034
## iter  80 value 23710.809241
## iter  90 value 23701.626307
## iter 100 value 23695.664388
## iter 110 value 23673.992605
## iter 120 value 23668.072709
## iter 130 value 23663.996065
## iter 140 value 23504.827082
## iter 150 value 23493.850532
## iter 160 value 23489.451884
## iter 170 value 23485.778114
## iter 180 value 23482.473128
## iter 190 value 23473.452997
## iter 200 value 23466.276861
## iter 210 value 23461.167045
## iter 220 value 23451.242944
## iter 230 value 23449.327687
## iter 240 value 23447.477591
## iter 250 value 23443.566335
## iter 260 value 23442.615155
## iter 270 value 23440.323979
## iter 280 value 23439.705291
## iter 290 value 23438.207417
## iter 300 value 23437.650162
## iter 310 value 23436.647547
## iter 320 value 23435.819333
## final  value 23435.475059
## converged
```

```r
predover <- predict(nn_oversample, data = oversampled_train_data)
train.binpredover <- predict(nn_oversample, oversampled_train_data, type = c("class"))
mean(oversampled_train_data$V25 == train.binpredover)
```

```
## [1] 0.591661
```

```r
auc(oversampled_train_data$V25, predover)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response, predictor, auc = TRUE, ...): Deprecated use a
```

```
## matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.

## Setting direction: controls < cases

## Area under the curve: 0.6168
```

```r
#TEST
predovertest <- predict(nn_oversample, newdata = test.data)
testover.binpred <- predict(nn_oversample, test.data, type = c("class"))
mean(test.data$V25 == testover.binpred)
```

```
## [1] 0.4765333
```

```r
auc(test.data$V25, predovertest)
```

```
## Setting levels: control = 0, case = 1

## Warning in roc.default(response, predictor, auc = TRUE, ...): Deprecated use a
## matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.

## Setting direction: controls < cases

## Area under the curve: 0.6109
```

```r
tablenn_balanced <- table(test.data$V25, testover.binpred)
tablenn_balanced
```

```
##    testover.binpred
##        0    1
##   0 2302 3530
##   1  396 1272
```

```r
class_accnnbal1 <- tablenn_balanced[1,1]/(tablenn_balanced[1,1] + tablenn_balanced[1,2])
class_accnnbal2 <- tablenn_balanced[2,1]/(tablenn_balanced[2,1] + tablenn_balanced[2,2])
(class_accnnbal1 + class_accnnbal2)/2 #average class accuracy
```

```
## [1] 0.3160644
```

```r
1/((1/class_accnnbal1)+(1/class_accnnbal2)) #harmonic mean
```

```
## [1] 0.1482454
```

```r
mean(testover.binpred == test.data$V25) #accuracy
```

```
## [1] 0.4765333
```

```r
auc(test.data$V25, predovertest) #area under roc curve
```

```
## Setting levels: control = 0, case = 1

## Warning in roc.default(response, predictor, auc = TRUE, ...): Deprecated use a
## matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.

## Setting direction: controls < cases

## Area under the curve: 0.6109
```

```r
tablenn_balanced[2,2]/(tablenn_balanced[2,1] + tablenn_balanced[2,2]) #specificity
```

```
## [1] 0.7625899
```

```r
recallnn_balanced <- tablenn_balanced[1,1]/(tablenn_balanced[1,1] + tablenn_balanced[1,2])
recallnn_balanced
```

```
## [1] 0.3947188
```

```r
precisionnnbal <- tablenn_balanced[1,1]/(tablenn_balanced[1,1] + tablenn_balanced[2,1])
precisionnnbal
```

```
## [1] 0.8532246
```

```r
F1nnbal <- 2*recallnn_balanced*precisionnnbal/(recallnn_balanced + precisionnnbal) #F1 statistic
F1nnbal
```

```
## [1] 0.5397421
```