2103 Project

2022-11-15

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
```

```
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, 0, -2, 0, -1, -1, 0, 0, 0, 2, -1, -2, -2,~
## $ V11 <int> -2, 0, 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, ~
```

```
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)
```

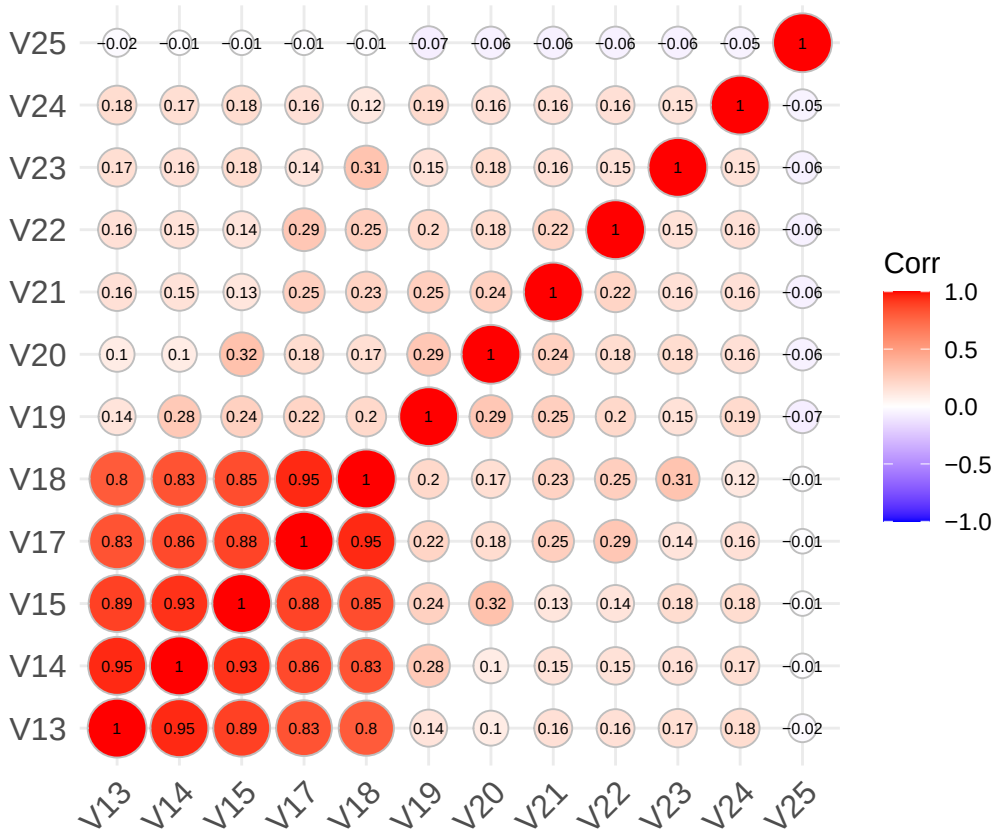
```
##
##     0     1     2     3     4     5     6
##    14 10585 14030  4917   123   280    51
```

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

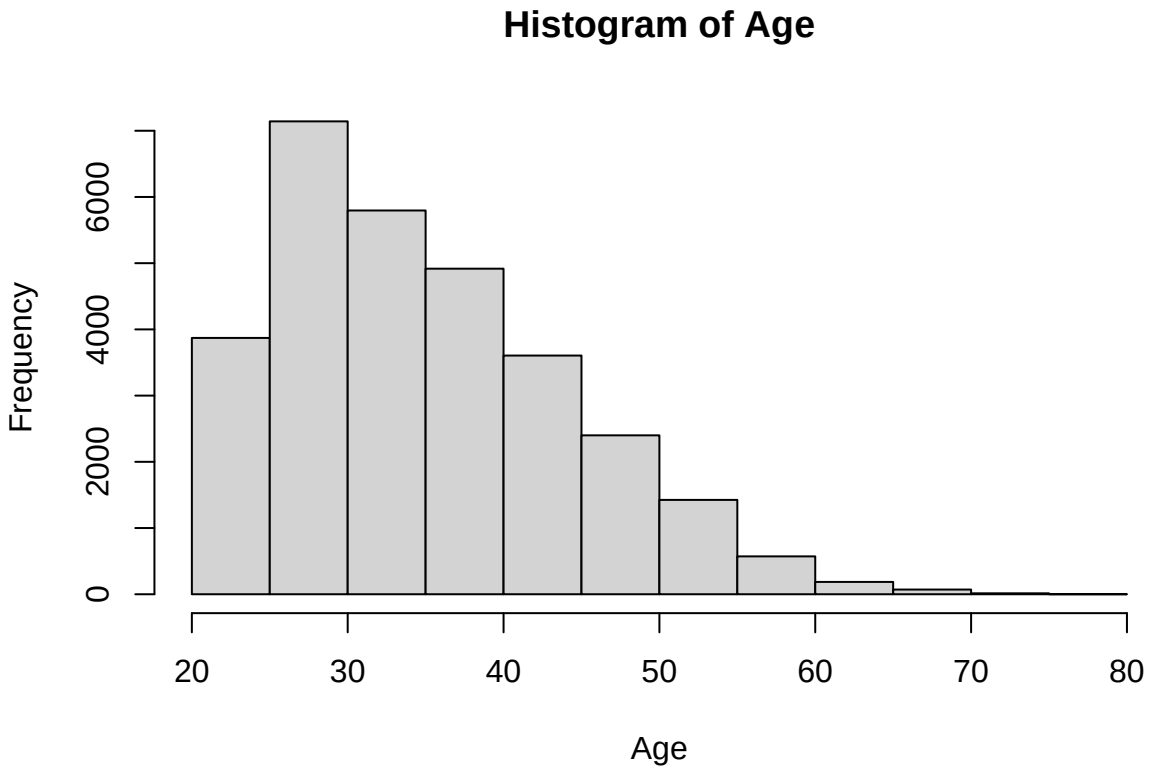
```
##
##     0     1     2     3
##    54 13659 15964   323
```

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.

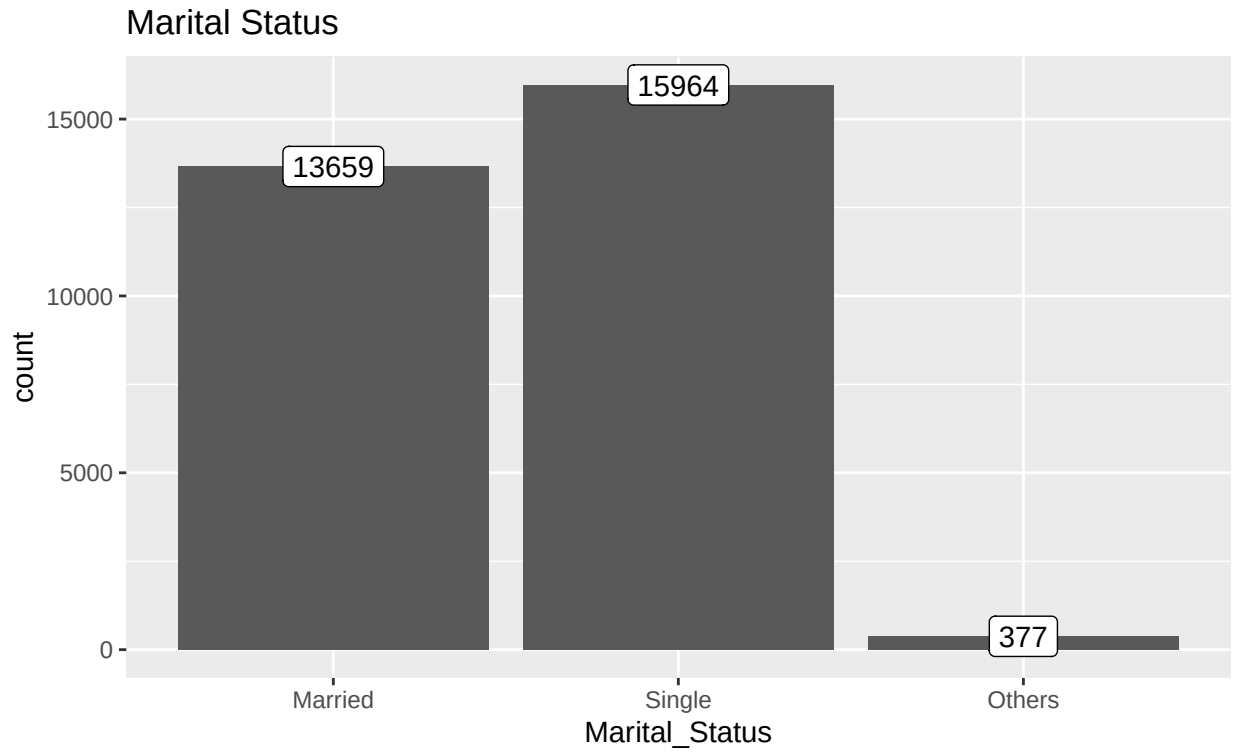
We check our continuous variables to check if any of them has strong explanatory power for our variable of interest V25 using a correlation matrix



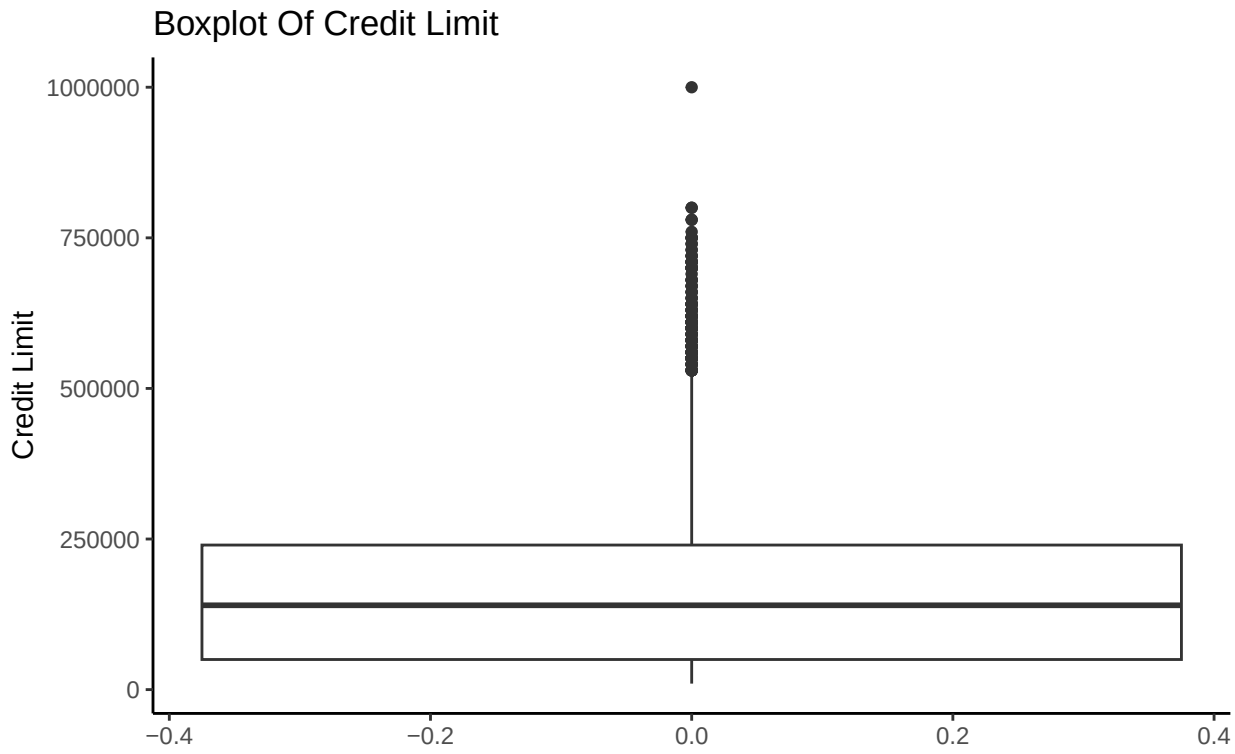
We take a look at the distribution of ages in our dataset



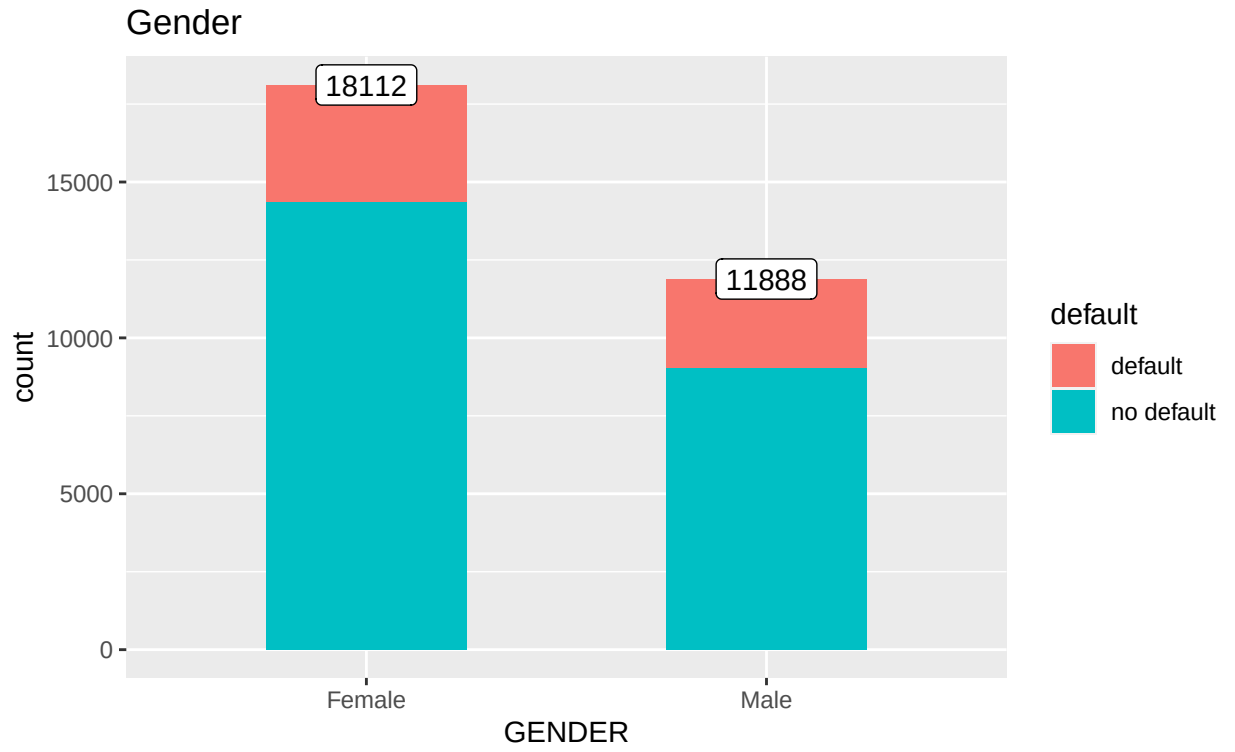
We also look at the distribution of marital statuses in our dataset



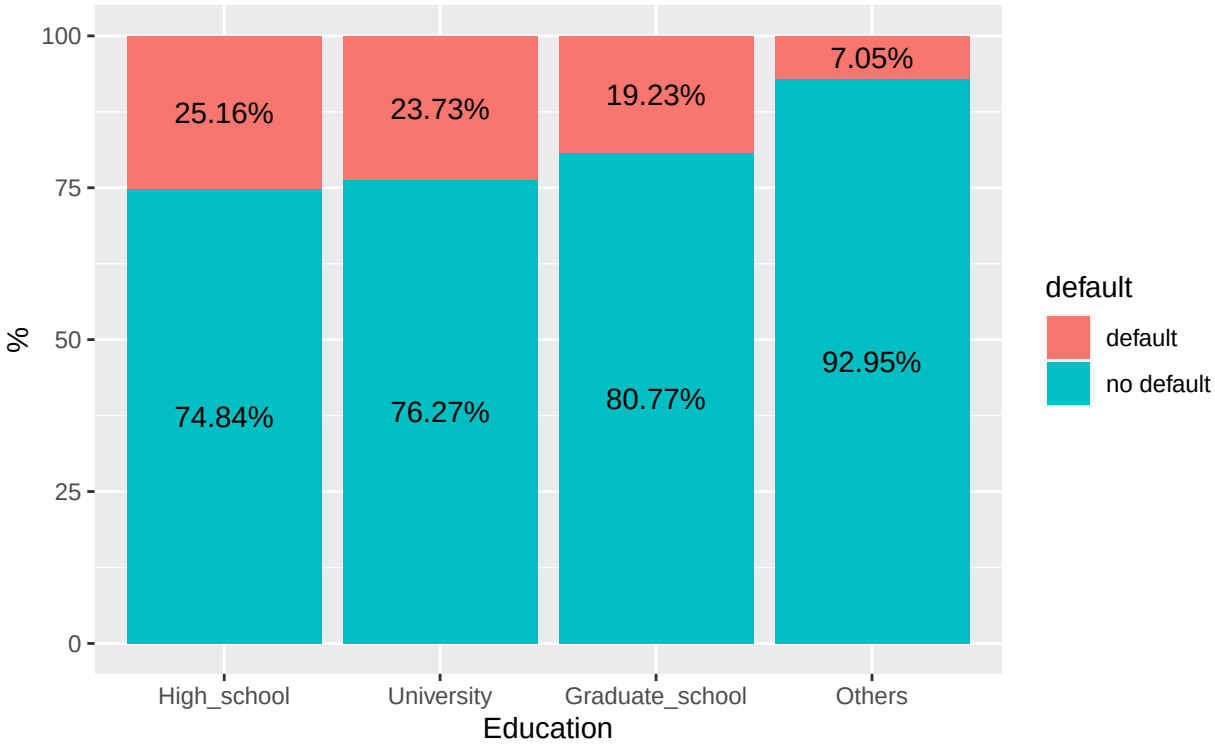
We check the boxplot of credit limit



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



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




```
##
##  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

##    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
```

```
##
## 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
```

```
##
## 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
```

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

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

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

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

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

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

```
##
## 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
```

```
##
## 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
```

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

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

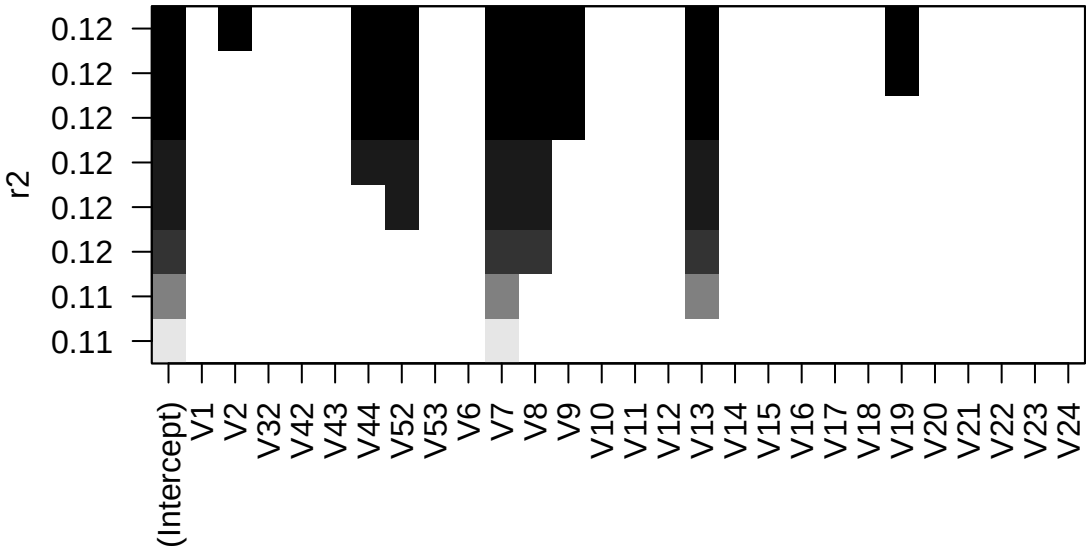
```
## [1] 0.455545
```

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

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

```
## 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 ) " " " " " " " " "*" " " " " " " " " "
```




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

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

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

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

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

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

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

```
## # 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
```

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

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

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

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

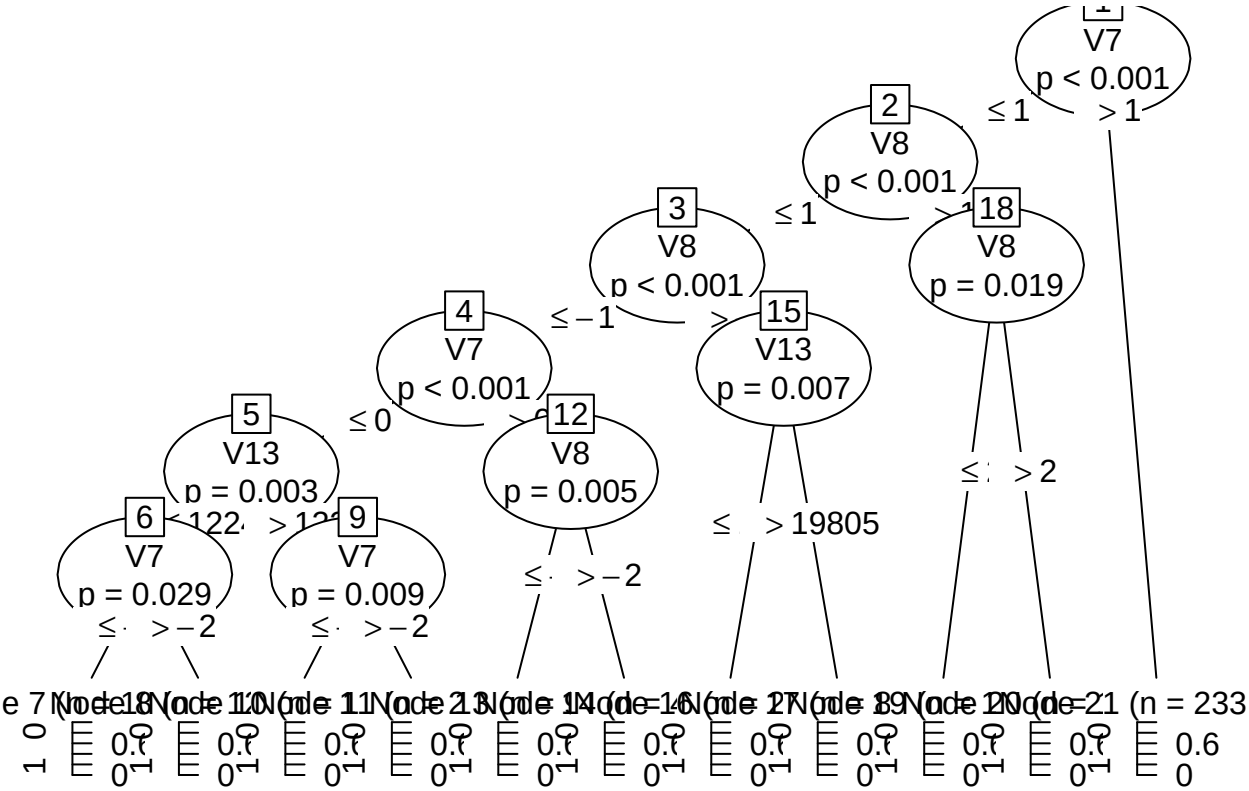
```
## Setting direction: controls < cases
```

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

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

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

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


1
V7
p < 0.001
≤ 1
> 1
2
V8
p < 0.001
≤ 1
18
V8
p = 0.019
3
V8
p < 0.001
≤ −1
15
V13
p = 0.007
4
V7
p < 0.001
≤ 0
12
V8
p = 0.005
5
V13
p = 0.003
6
V7
p = 0.029
9
V7
p = 0.009
≤ 2
> 2
> −2
≤ 19805
> 19805
Node 7 (n = 18)
Node 10 (n = 11)
Node 11 (n = 13)
Node 13 (n = 14)
Node 14 (n = 16)
Node 17 (n = 89)
Node 20 (n = 21)
Node 21 (n = 233
1
0
0.6
0

```
{r balancing} # #OVERSAMPLING # oversampled_train_data <- ovun.sample(V2
~ ., data = train.data, method = "over", #
= 2*nrow(subset(train.data, train.data$V25 == 0)))$data #  #
table(oversampled_train_data$V25) #
```

```
{r svm AFTER BALANCING} #  # svm.model.feature.selection.balanced
<- svm(as.factor(V25) ~ V7 + V13 + V8, #
= oversampled_train_data, type="C-classification", #
cost=0.2) # result_train_feature_selection_balanced <- predict(svm.model
oversampled_train_data[,-25]) # result_test_feature_selection_balanced
<- predict(svm.model.feature.selection.balanced, test.data[,-25])
# table(pred=result_train_feature_selection_balanced, actual=oversampled
# table(pred=result_test_feature_selection_balanced, actual=test.data$V2
# mean(result_train_feature_selection_balanced == oversampled_train_data
# mean(result_test_feature_selection_balanced == test.data$V25)
#  #  #
```

```
{r logistic regression AFTER BALANCING} # log_model_balanced
<- glm(V25 ~., data = oversampled_train_data, family = "binomial")
# summary(log_model_balanced) # predictlr_balanced <- predict(log_model_
test.data, type = "response") # predictlr_balanced <- ifelse(predictlr_b
>0.5, 1, 0) # tablelr <- table(test.data$V25, predictlr_balanced)
# tablelr # class_acc1 <- tablelr[1,1]/(tablelr[1,1] + tablelr[1,2])
# class_acc2 <- tablelr[2,1]/(tablelr[2,1] + tablelr[2,2]) #
(class_acc1 + class_acc2)/2 #average class accuracy # 1/((1/class_acc1)+
#harmonic mean # mean(predictlr_balanced == test.data$V25)
#accuracy # auc(test.data$V25, predictlr_balanced) #area under
roc curve # tablelr[2,2]/(tablelr[2,1] + tablelr[2,2]) #specificity
# 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 lm after balance} #  # better_model_balanced <- lm(V25~V2+V3+V4+V5+V6
data = oversampled_train_data) # summary(better_model_balanced)
```