R - Decision Tree Tuning & Weighted Loss Matrix

Karthick Sharan

2/21/2022

R Markdown

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- readxl::read_excel('C:/Masters - Business Analytics/Data Mining/assignment 2/German Credit.xls')
names(df)[names(df) == "RADIO/TV"] <- "RADIO_TV"
names(df)[names(df) == "CO-APPLICANT"] <- "CO_APPLICANT"
```

Question (a)

```
# Displaying column names
col <- names(df)
print("Coulmn Names:")
```

```
## [1] "Coulmn Names:"
```

```
print(col)
```

```
##  [1] "OBS#"            "CHK_ACCT"        "DURATION"        "HISTORY"
##  [5] "NEW_CAR"         "USED_CAR"        "FURNITURE"       "RADIO_TV"
##  [9] "EDUCATION"       "RETRAINING"      "AMOUNT"          "SAV_ACCT"
## [13] "EMPLOYMENT"      "INSTALL_RATE"    "MALE_DIV"        "MALE_SINGLE"
## [17] "MALE_MAR_or_WID" "CO_APPLICANT"    "GUARANTOR"       "PRESENT_RESIDENT"
## [21] "REAL_ESTATE"     "PROP_UNKN_NONE"  "AGE"             "OTHER_INSTALL"
## [25] "RENT"            "OWN_RES"         "NUM_CREDITS"     "JOB"
## [29] "NUM_DEPENDENTS"  "TELEPHONE"       "FOREIGN"         "RESPONSE"
```

```r
# Selecting only Categorical columns (according to dataset definintions pdf)
c <- col[c(-3,-11,-14,-23,-27,-29)] # c has categorical column names
df[c] <- lapply(df[c], factor)
summary(df)
```

```
##       OBS#     CHK_ACCT    DURATION    HISTORY NEW_CAR USED_CAR FURNITURE
##  1      :  1   0:274    Min.   : 4.0   0: 40   0:766   0:897    0:819
##  2      :  1   1:269    1st Qu.:12.0   1: 49   1:234   1:103    1:181
##  3      :  1   2: 63    Median :18.0   2:530
##  4      :  1   3:394    Mean   :20.9   3: 88
##  5      :  1            3rd Qu.:24.0   4:293
##  6      :  1            Max.   :72.0
##  (Other):994
##  RADIO_TV EDUCATION RETRAINING     AMOUNT      SAV_ACCT EMPLOYMENT
##  0:720    0:950     0:903      Min.   :  250   0:603    0: 62
##  1:280    1: 50     1: 97      1st Qu.: 1366   1:103    1:172
##                                Median : 2320   2: 63    2:339
##                                Mean   : 3271   3: 48    3:174
##                                3rd Qu.: 3972   4:183    4:253
##                                Max.   :18424
##
##   INSTALL_RATE   MALE_DIV MALE_SINGLE MALE_MAR_or_WID CO_APPLICANT GUARANTOR
##  Min.   :1.000   0:950    0:452       0:908           0:959        0:948
##  1st Qu.:2.000   1: 50    1:548       1: 92           1: 41        1: 52
##  Median :3.000
##  Mean   :2.973
##  3rd Qu.:4.000
##  Max.   :4.000
##
##  PRESENT_RESIDENT REAL_ESTATE PROP_UNKN_NONE      AGE        OTHER_INSTALL
##  1:130             0:718       0:846          Min.   :19.00   0:814
##  2:308             1:282       1:154          1st Qu.:27.00   1:186
##  3:149                                        Median :33.00
##  4:413                                        Mean   :35.55
##                                               3rd Qu.:42.00
##                                               Max.   :75.00
##
##  RENT    OWN_RES  NUM_CREDITS    JOB     NUM_DEPENDENTS  TELEPHONE FOREIGN
##  0:821   0:287   Min.   :1.000   0: 22   Min.   :1.000   0:596     0:963
##  1:179   1:713   1st Qu.:1.000   1:200   1st Qu.:1.000   1:404     1: 37
##                  Median :1.000   2:630   Median :1.000
##                  Mean   :1.407   3:148   Mean   :1.155
##                  3rd Qu.:2.000           3rd Qu.:1.000
##                  Max.   :4.000           Max.   :2.000
##
##  RESPONSE
##  0:300
##  1:700
##
##
##
##
##
```

```r
head(df)
```

```
## # A tibble: 6 x 32
##   `OBS#` CHK_ACCT DURATION HISTORY NEW_CAR USED_CAR FURNITURE RADIO_TV EDUCATION
##   <fct>  <fct>       <dbl> <fct>   <fct>   <fct>    <fct>     <fct>    <fct>
## 1 1      0               6 4       0       0        0         1        0
## 2 2      1              48 2       0       0        0         1        0
## 3 3      3              12 4       0       0        0         0        1
## 4 4      0              42 2       0       0        1         0        0
## 5 5      0              24 3       1       0        0         0        0
## 6 6      3              36 2       0       0        0         0        1
## # ... with 23 more variables: RETRAINING <fct>, AMOUNT <dbl>, SAV_ACCT <fct>,
## #   EMPLOYMENT <fct>, INSTALL_RATE <dbl>, MALE_DIV <fct>, MALE_SINGLE <fct>,
## #   MALE_MAR_or_WID <fct>, CO_APPLICANT <fct>, GUARANTOR <fct>,
## #   PRESENT_RESIDENT <fct>, REAL_ESTATE <fct>, PROP_UNKN_NONE <fct>, AGE <dbl>,
## #   OTHER_INSTALL <fct>, RENT <fct>, OWN_RES <fct>, NUM_CREDITS <dbl>,
## #   JOB <fct>, NUM_DEPENDENTS <dbl>, TELEPHONE <fct>, FOREIGN <fct>,
## #   RESPONSE <fct>
```

```r
str(df)
```

```
## tibble [1,000 x 32] (S3: tbl_df/tbl/data.frame)
##  $ OBS#            : Factor w/ 1000 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ CHK_ACCT        : Factor w/ 4 levels "0","1","2","3": 1 2 4 1 1 4 4 2 4 2 ...
##  $ DURATION        : num [1:1000] 6 48 12 42 24 36 24 36 12 30 ...
##  $ HISTORY         : Factor w/ 5 levels "0","1","2","3",..: 5 3 5 3 4 3 3 3 3 5 ...
##  $ NEW_CAR         : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 2 ...
##  $ USED_CAR        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
##  $ FURNITURE       : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 2 1 1 1 ...
##  $ RADIO_TV        : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 2 1 ...
##  $ EDUCATION       : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 1 1 1 ...
##  $ RETRAINING      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AMOUNT          : num [1:1000] 1169 5951 2096 7882 4870 ...
##  $ SAV_ACCT        : Factor w/ 5 levels "0","1","2","3",..: 5 1 1 1 1 5 3 1 4 1 ...
##  $ EMPLOYMENT      : Factor w/ 5 levels "0","1","2","3",..: 5 3 4 4 3 3 5 3 4 1 ...
##  $ INSTALL_RATE    : num [1:1000] 4 2 2 2 3 2 3 2 2 4 ...
##  $ MALE_DIV        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ MALE_SINGLE     : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 1 1 ...
##  $ MALE_MAR_or_WID : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ CO_APPLICANT    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ GUARANTOR       : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ PRESENT_RESIDENT: Factor w/ 4 levels "1","2","3","4": 4 2 3 4 4 4 4 2 4 2 ...
##  $ REAL_ESTATE     : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 2 1 ...
##  $ PROP_UNKN_NONE  : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 1 1 ...
##  $ AGE             : num [1:1000] 67 22 49 45 53 35 53 35 61 28 ...
##  $ OTHER_INSTALL   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RENT            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...
##  $ OWN_RES         : Factor w/ 2 levels "0","1": 2 2 2 1 1 1 2 1 2 2 ...
##  $ NUM_CREDITS     : num [1:1000] 2 1 1 1 2 1 1 1 1 2 ...
##  $ JOB             : Factor w/ 4 levels "0","1","2","3": 3 3 2 3 3 2 3 4 2 4 ...
##  $ NUM_DEPENDENTS  : num [1:1000] 1 1 2 2 2 2 1 1 1 1 ...
##  $ TELEPHONE       : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 2 1 1 ...
##  $ FOREIGN         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ RESPONSE        : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 2 2 2 1 ...
```

```r
# Checking the category proportion of Target variable
df1 <- filter(df,df["RESPONSE"]==1)
df0 <- filter(df,df["RESPONSE"]==0)
print(paste("Count of Bad credit :",count(df0)))
```

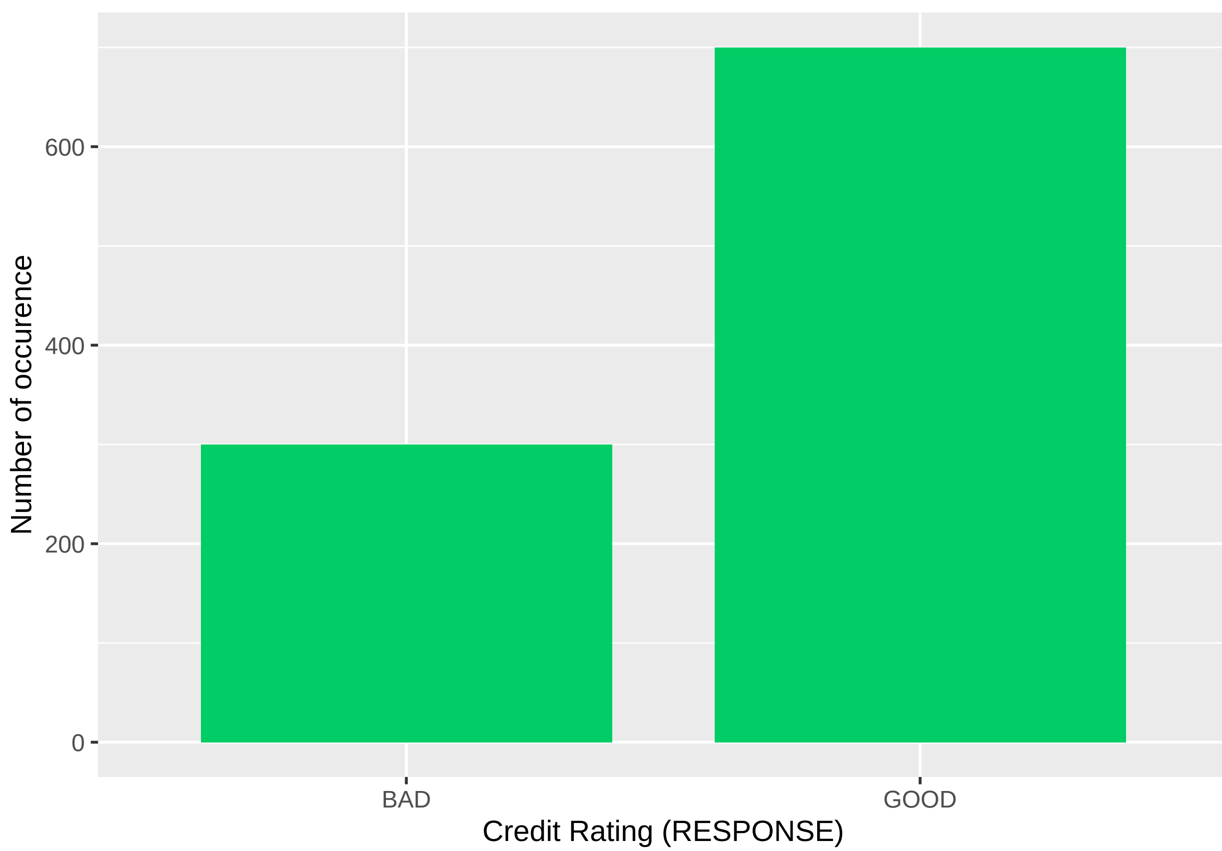
```
## [1] "Count of Bad credit : 300"
```

```r
print(paste("Count of Good credit :",count(df1)))
```

```
## [1] "Count of Good credit : 700"
```

```r
df2 <- df
df2$RESPONSE <- as.factor(ifelse(df2$RESPONSE == 0, "BAD", "GOOD"))
```

```r
ggplot(df2, aes(x=factor(RESPONSE)))+ geom_bar(stat="count", width=0.8,fill='springgreen3')+
   xlab('Credit Rating (RESPONSE)') + ylab('Number of occurence')
```

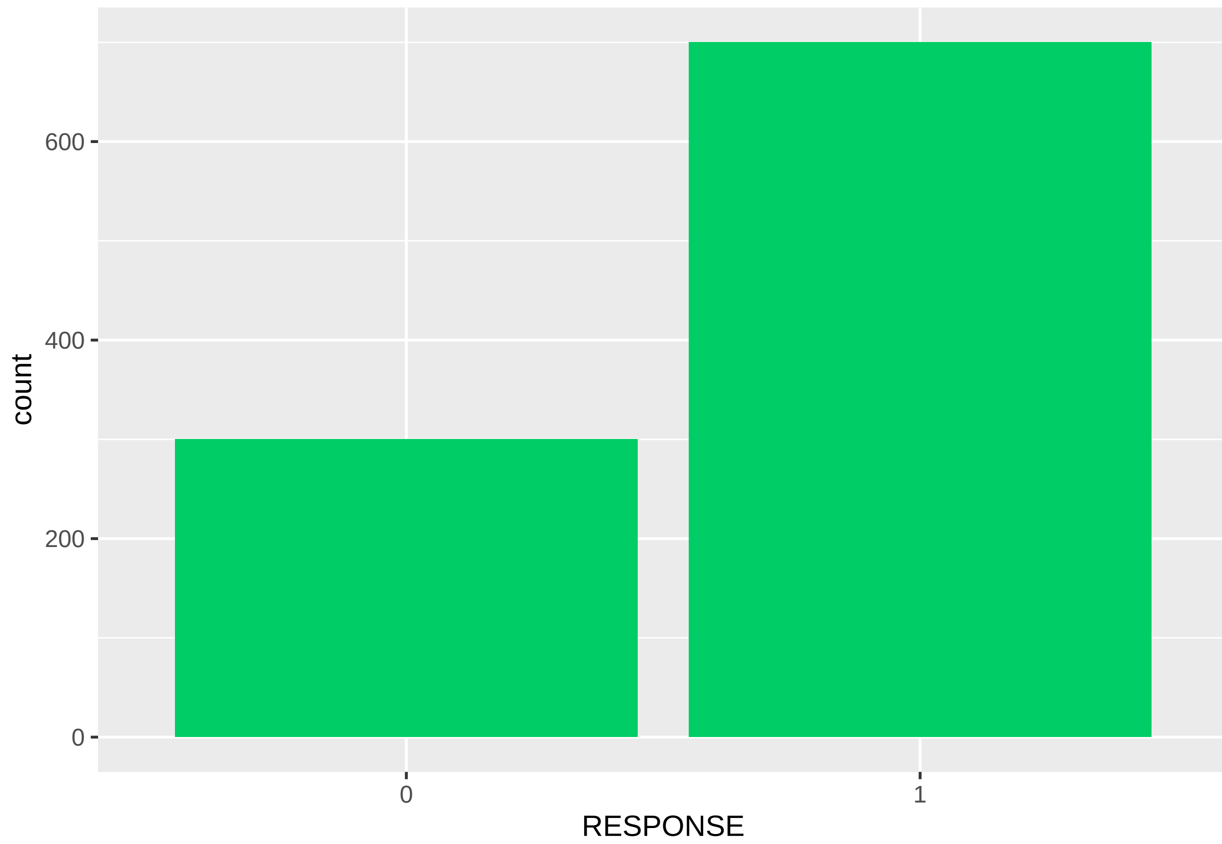



```r
  ggtitle("Distribution of RESPONSE variable (GOOD & BAD Credit)")
```

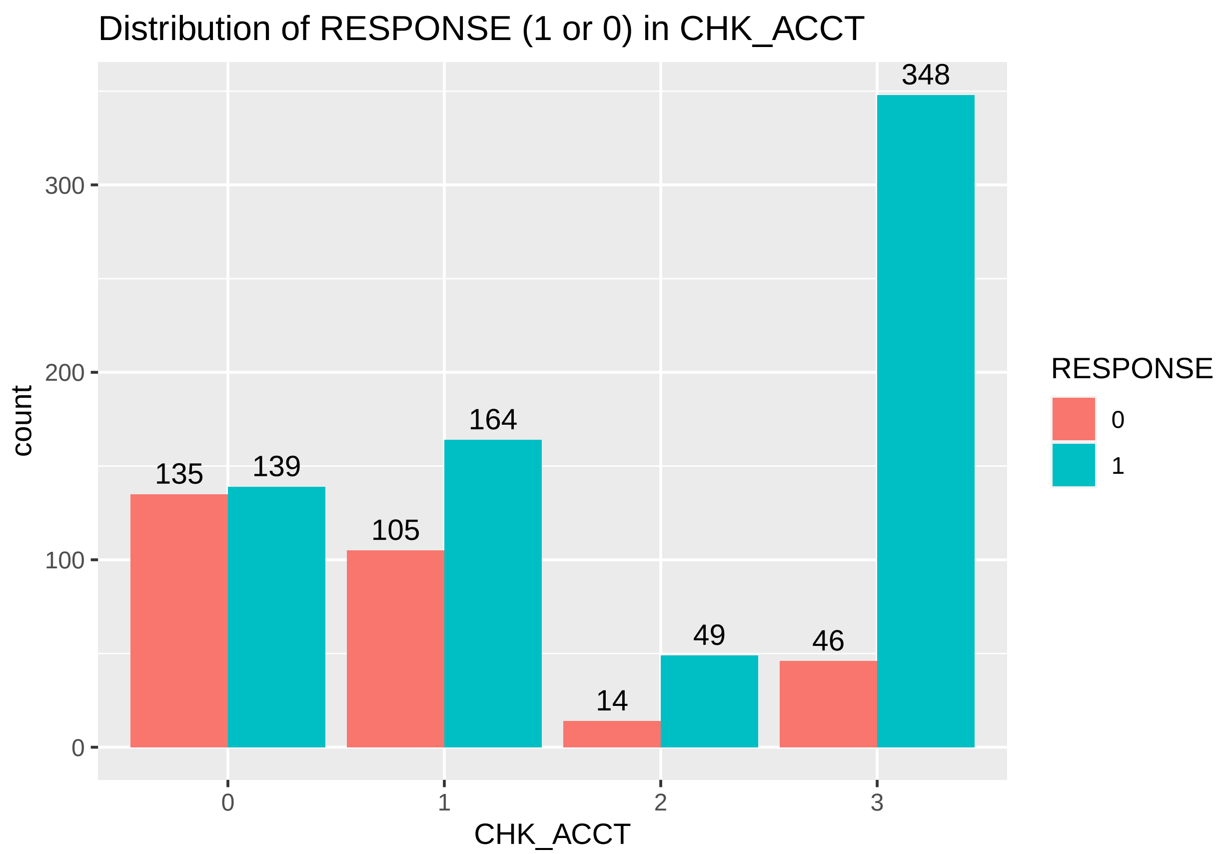
```
## $title
## [1] "Distribution of RESPONSE variable (GOOD & BAD Credit)"
```

```
##
## attr(,"class")
## [1] "labels"
```

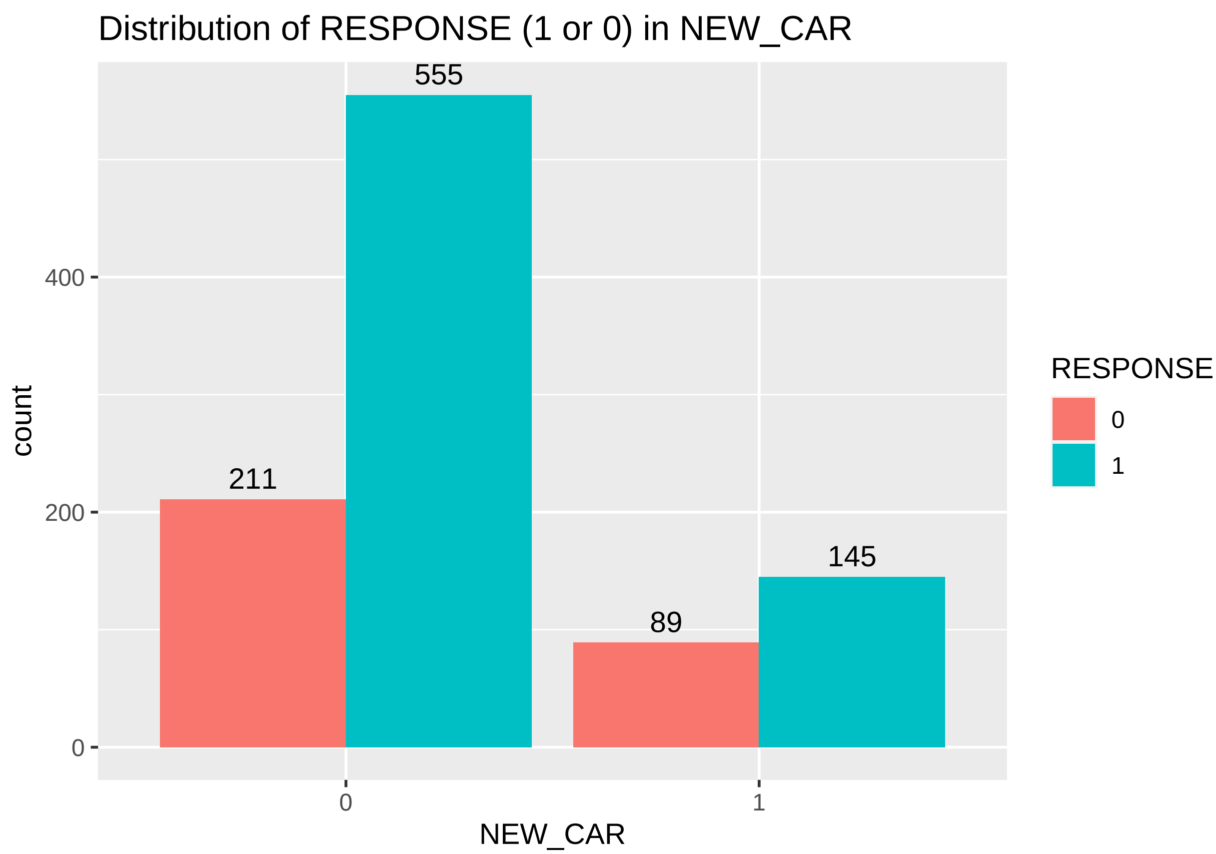
```
ggplot(df, aes(x=RESPONSE))+geom_bar(fill='springgreen3')
```

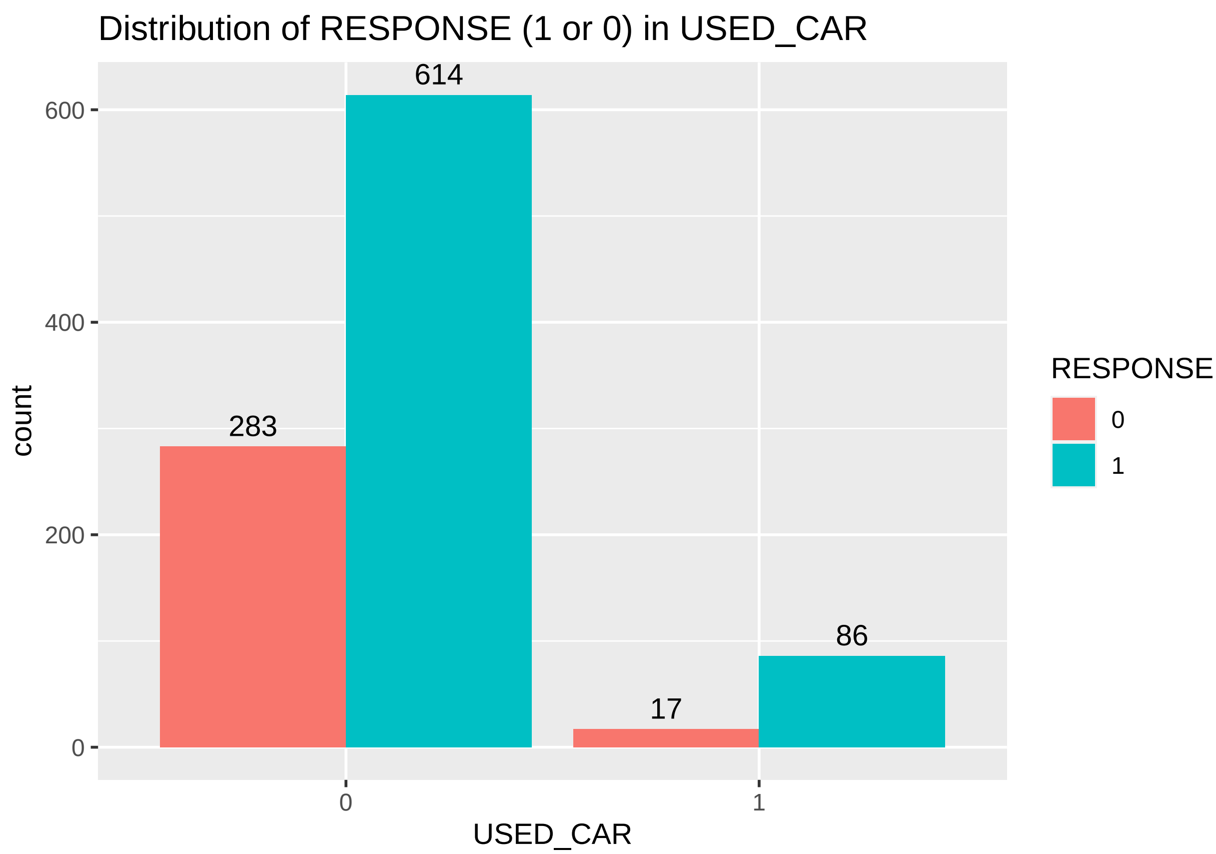


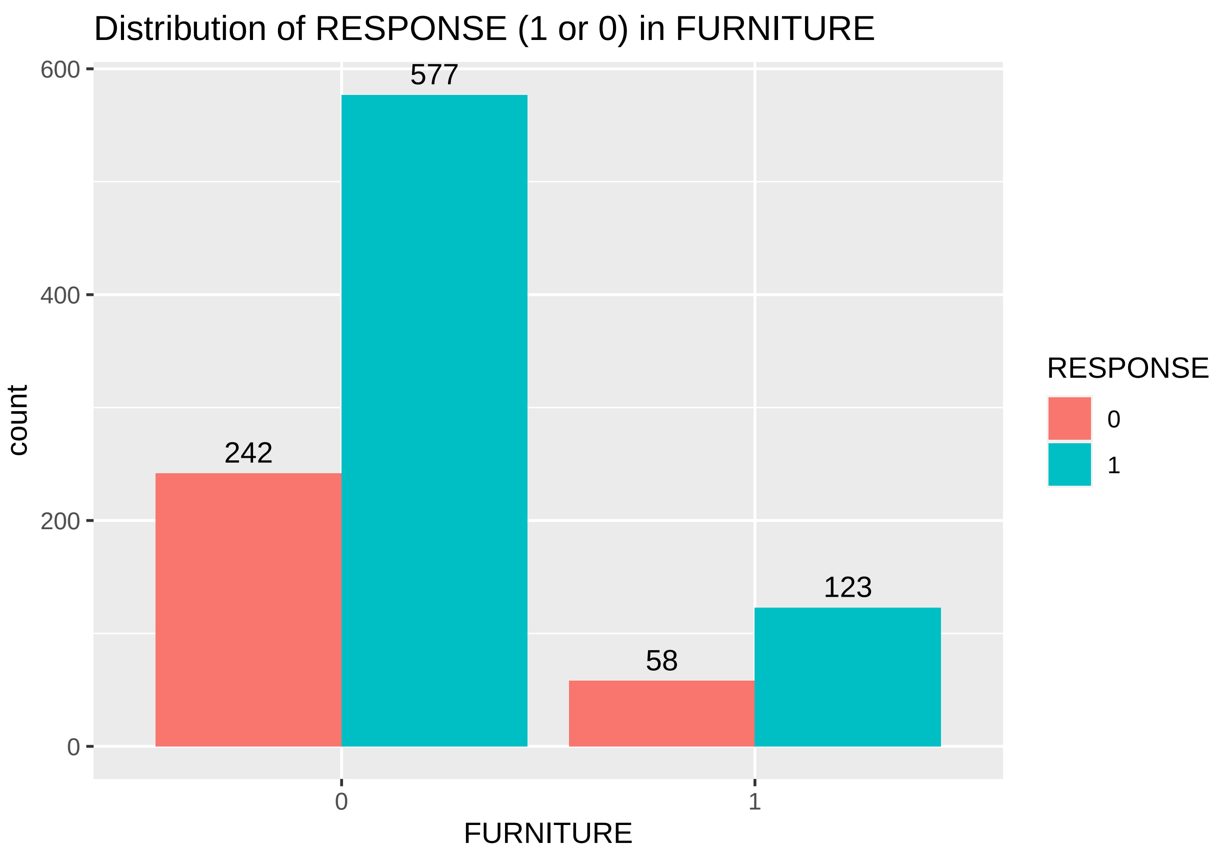

```
# Plotting all categorical variables with respect to Target variable
for (i in c[2:25]){
  print(ggplot(df,aes_string(x=i, fill="RESPONSE"))+geom_bar(position="dodge")+
  geom_text(stat='count', aes(label=..count..),position = position_dodge(0.9), vjust=-0.5)+
    ggtitle(paste("Distribution of RESPONSE (1 or 0) in",i)))
}
```

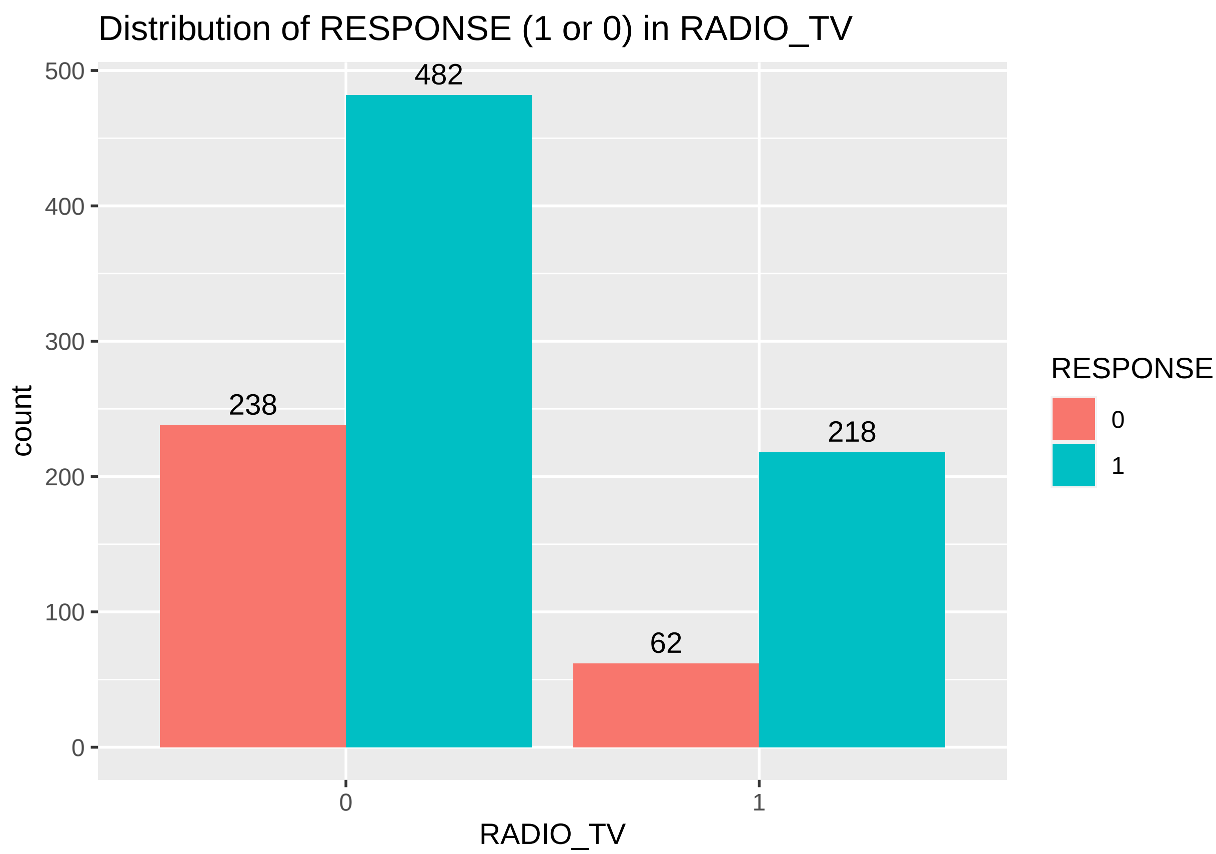

Distribution of RESPONSE (1 or 0) in CHK_ACCT
count
CHK_ACCT
RESPONSE
0
1
135
139
105
164
14
49
46
348
300
200
100
0
0
1
2
3

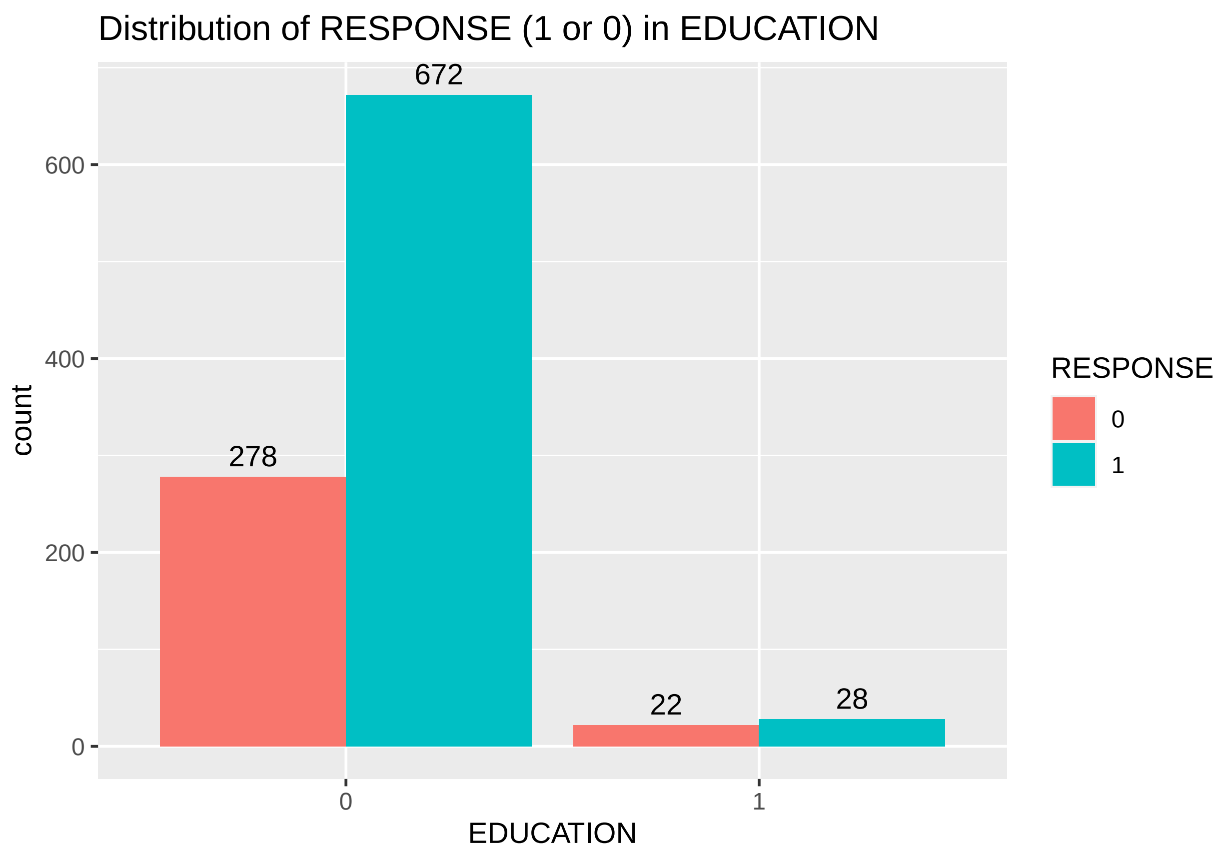

Distribution of RESPONSE (1 or 0) in HISTORY
count
300
200
100
0
25
15
28
21
169
361
28
60
50
243
0
1
2
3
4
HISTORY
RESPONSE
0
1


Distribution of RESPONSE (1 or 0) in NEW_CAR
555
211
89
145
400
200
0
count
0
1
NEW_CAR
RESPONSE
0
1

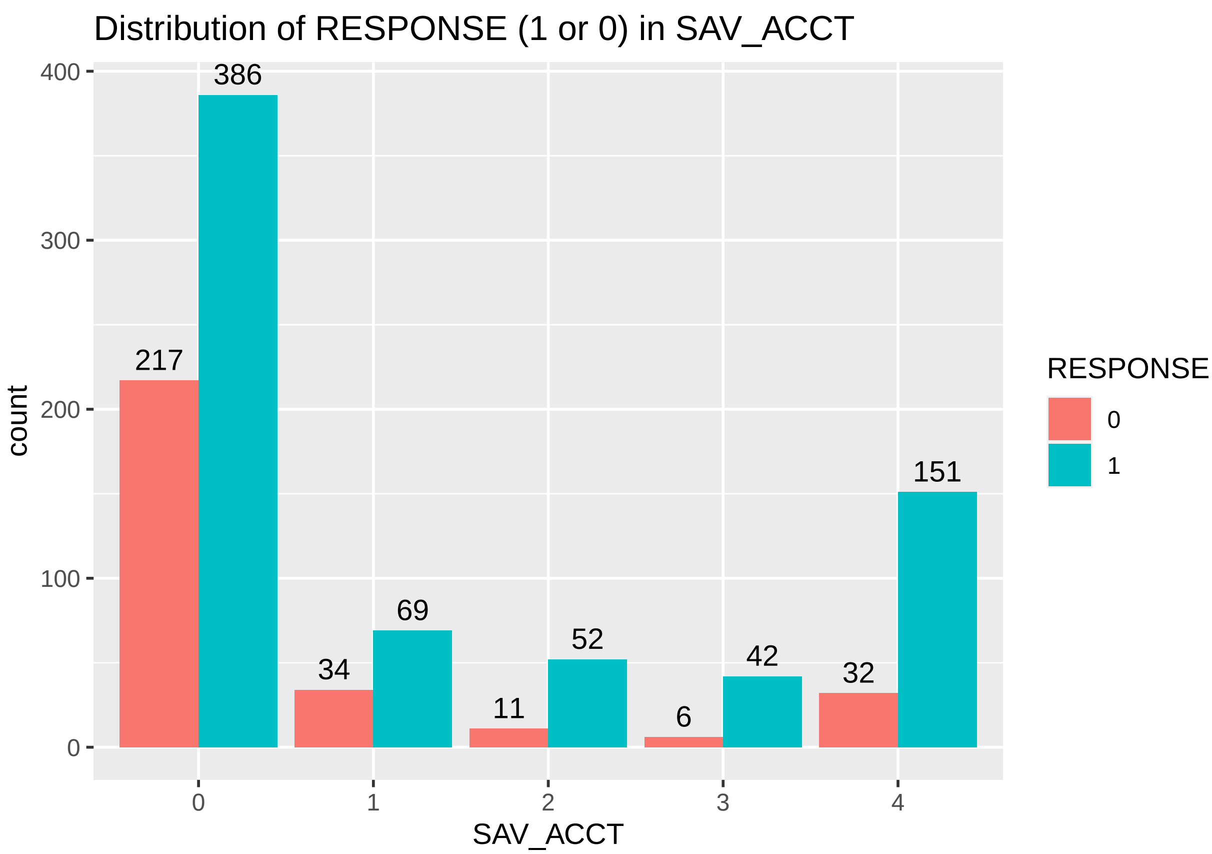

Distribution of RESPONSE (1 or 0) in USED_CAR
614
283
17
86
600
400
200
0
count
0
1
USED_CAR
RESPONSE
0
1

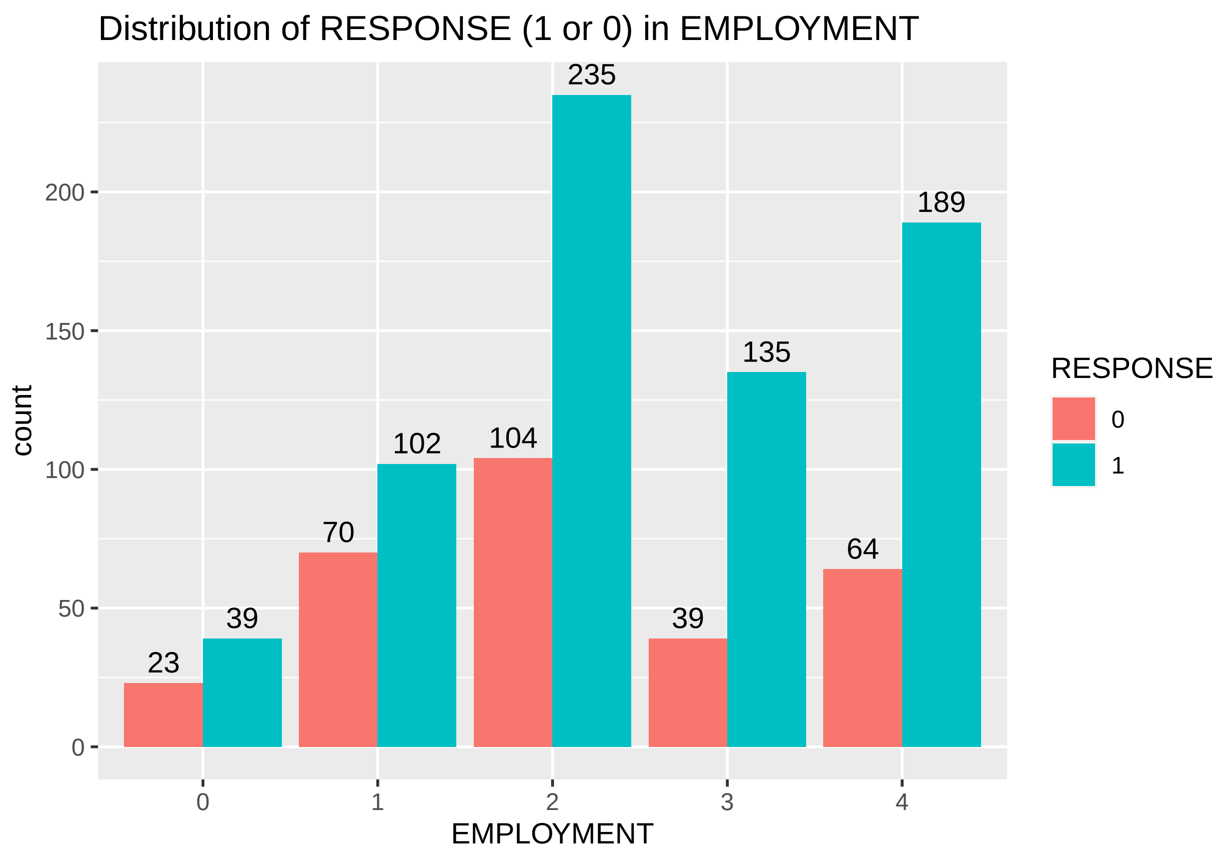

Distribution of RESPONSE (1 or 0) in FURNITURE
600
400
200
0
count
577
242
123
58
0
1
FURNITURE
RESPONSE
0
1


Distribution of RESPONSE (1 or 0) in RADIO_TV
count
500
400
300
200
100
0
482
238
218
62
0
1
RADIO_TV
RESPONSE
0
1

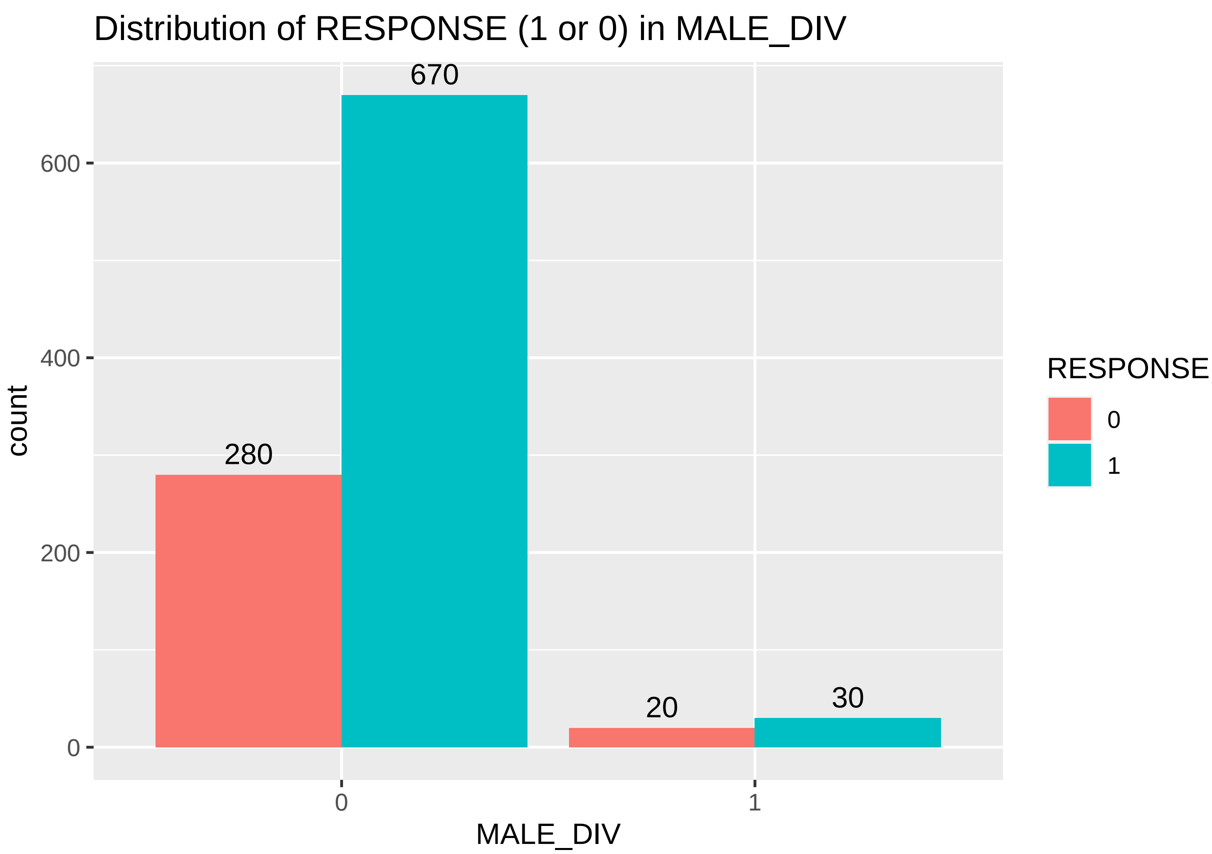

Distribution of RESPONSE (1 or 0) in EDUCATION
672
278
22
28
600
400
200
0
count
0
1
EDUCATION
RESPONSE
0
1

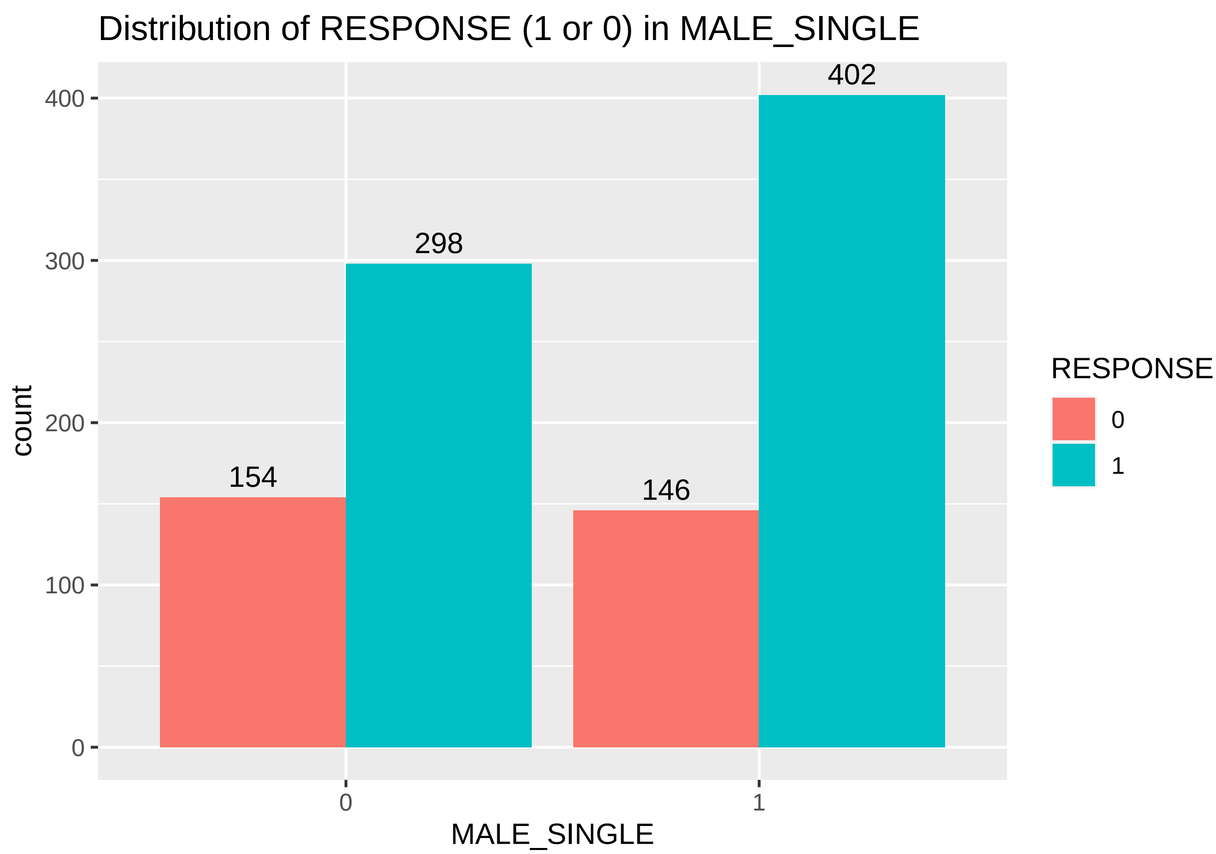

Distribution of RESPONSE (1 or 0) in RETRAINING
637
600
400
266
200
34
63
0
count
0
1
RETRAINING
RESPONSE
0
1

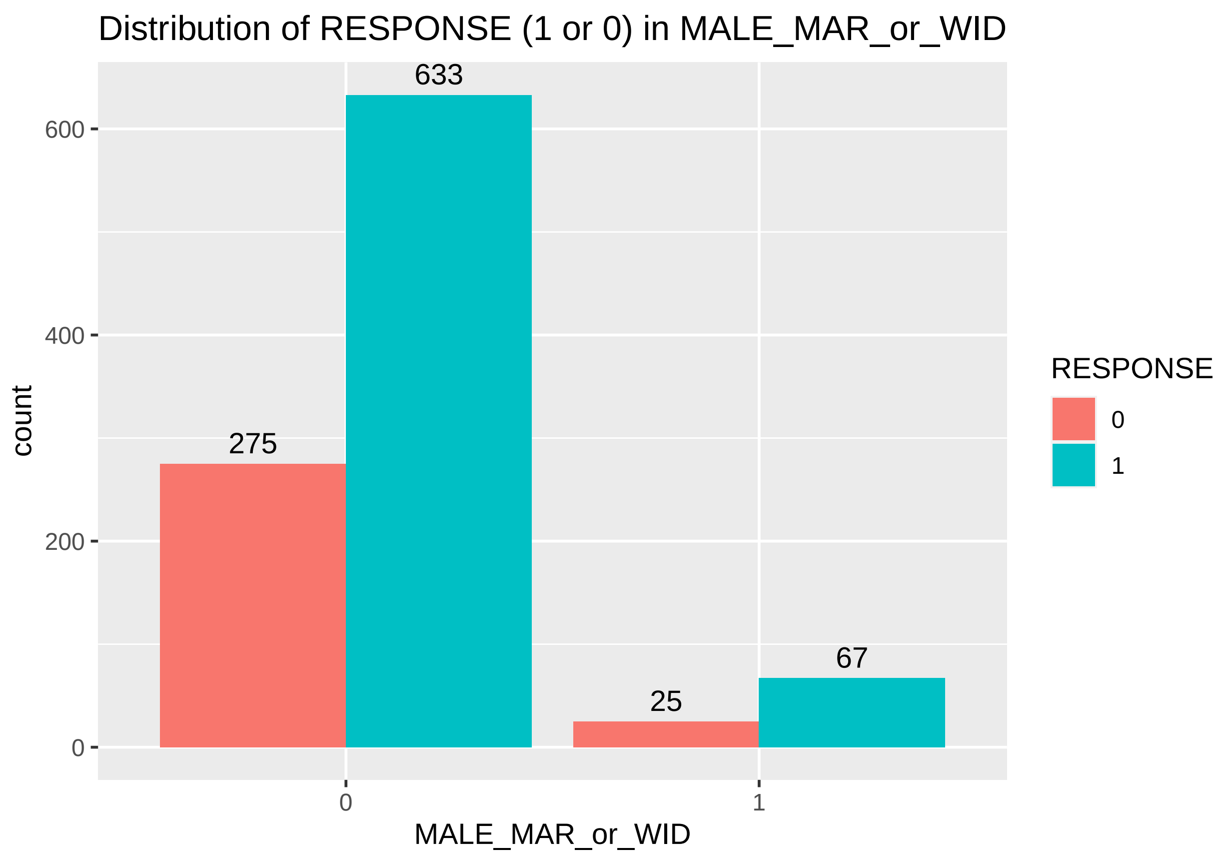

Distribution of RESPONSE (1 or 0) in SAV_ACCT
count
SAV_ACCT
RESPONSE
0
1
400
300
200
100
0
1
2
3
4
217
386
34
69
11
52
6
42
32
151

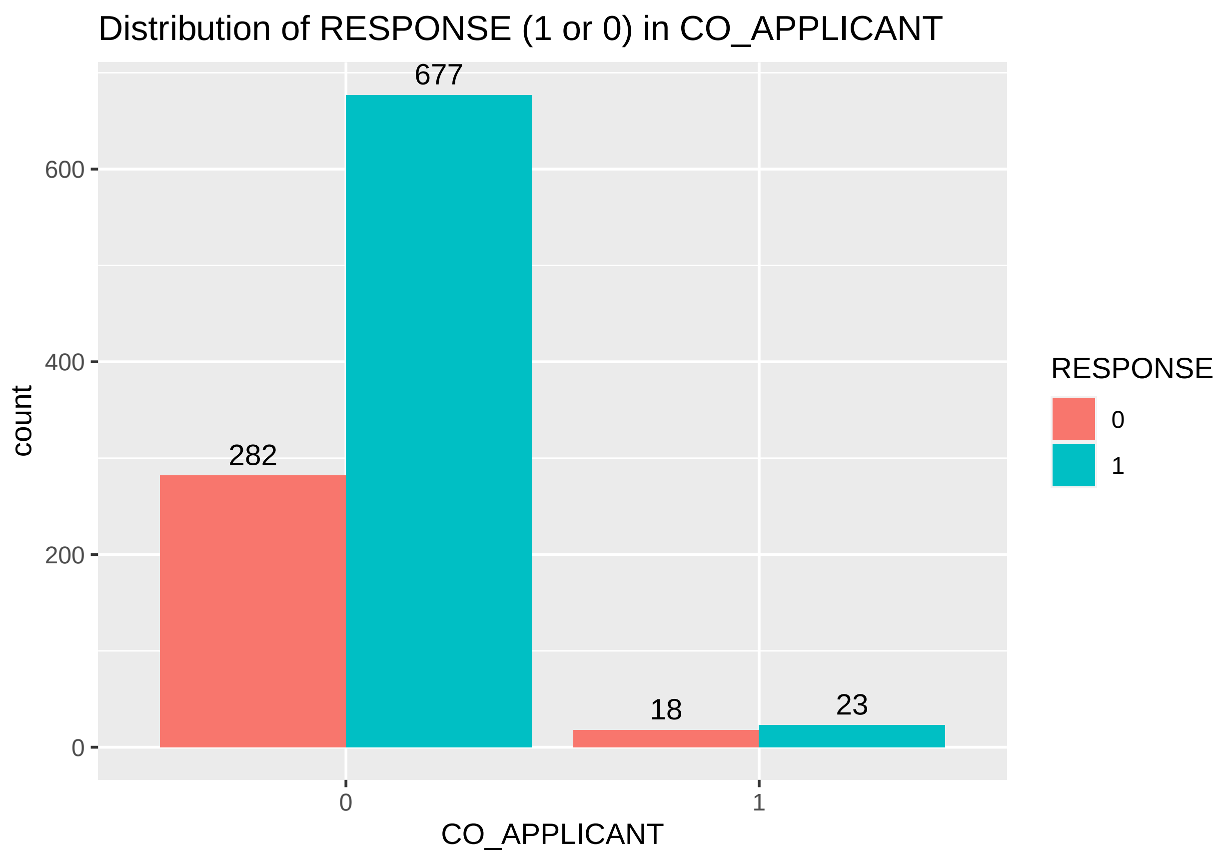


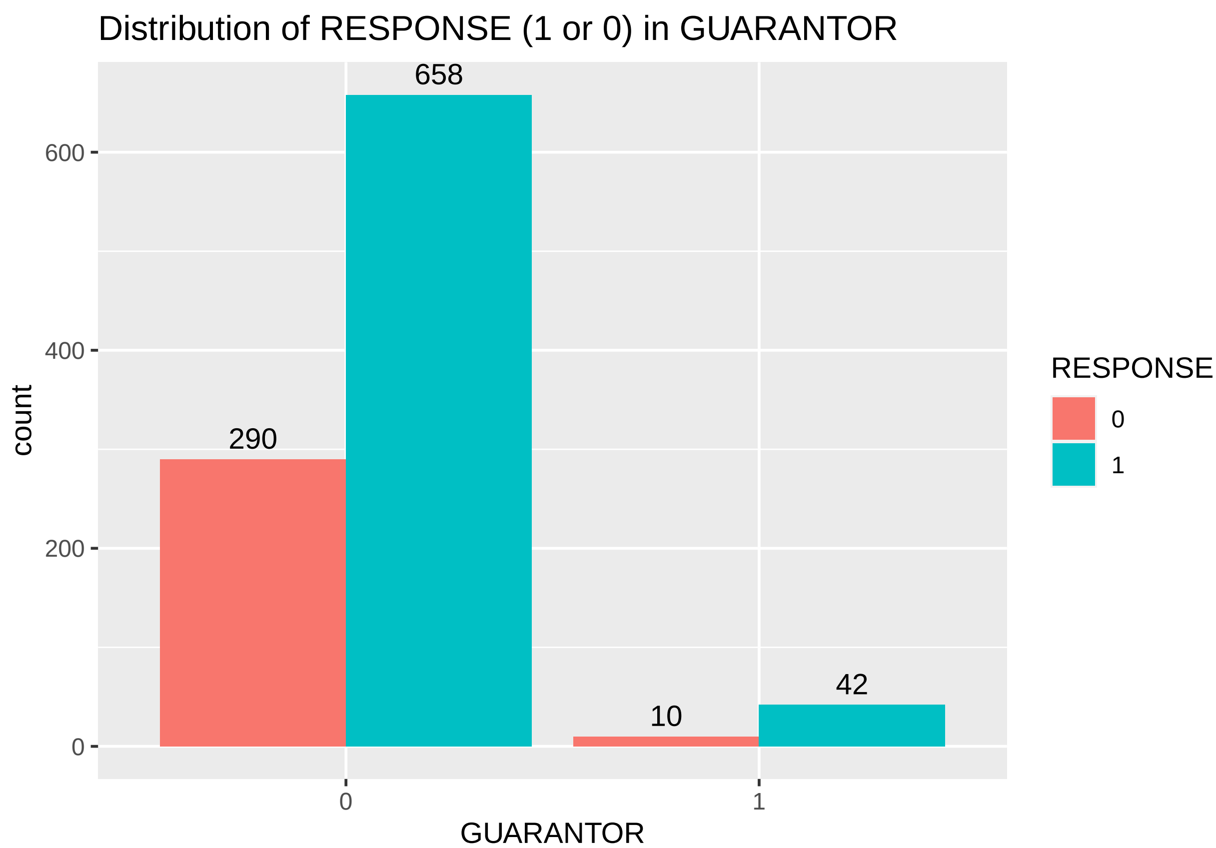
Distribution of RESPONSE (1 or 0) in EMPLOYMENT
count
0
50
100
150
200
23
39
70
102
104
235
39
135
64
189
0
1
2
3
4
EMPLOYMENT
RESPONSE
0
1

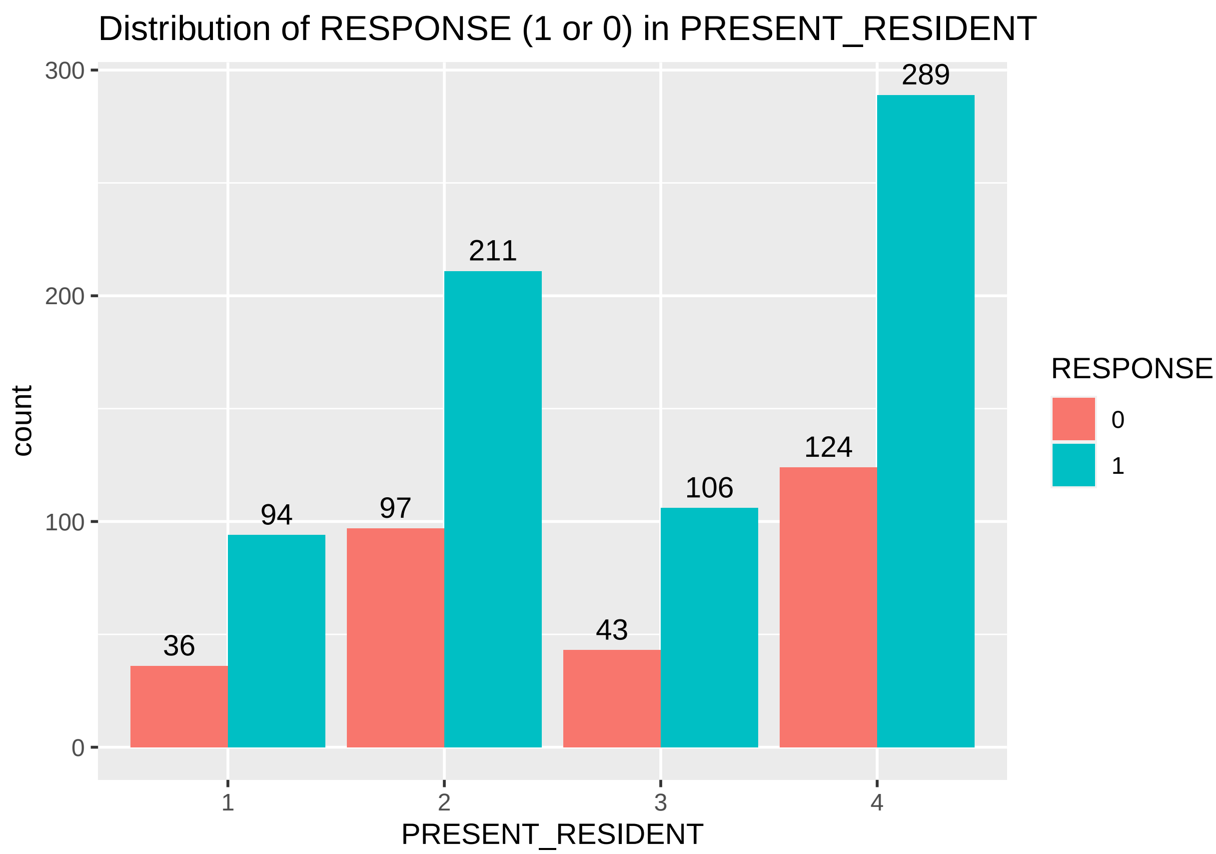

Distribution of RESPONSE (1 or 0) in MALE_DIV
670
280
20
30
600
400
200
0
count
0
1
MALE_DIV
RESPONSE
0
1


Distribution of RESPONSE (1 or 0) in MALE_SINGLE
400
300
200
100
0
count
154
298
146
402
0
1
MALE_SINGLE
RESPONSE
0
1

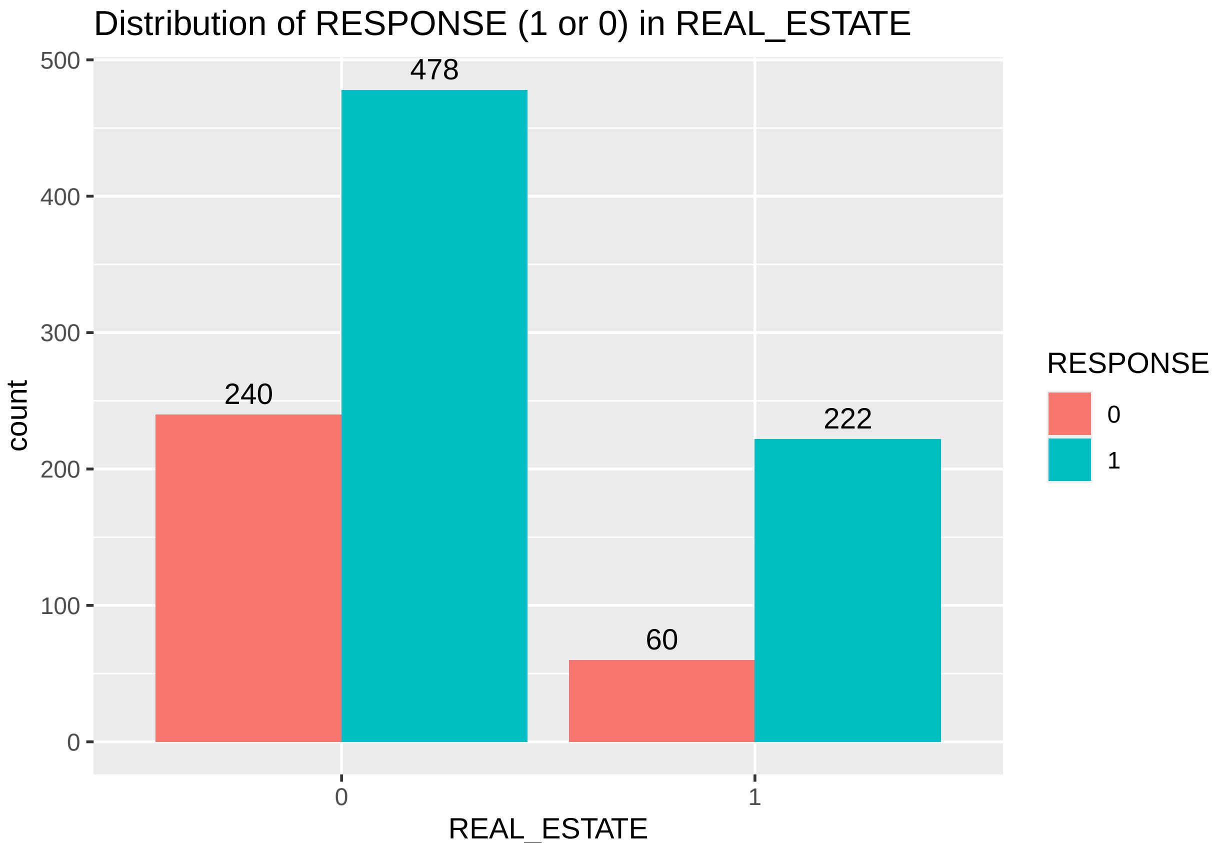

Distribution of RESPONSE (1 or 0) in MALE_MAR_or_WID
633
275
25
67
600
400
200
0
count
0
1
MALE_MAR_or_WID
RESPONSE
0
1

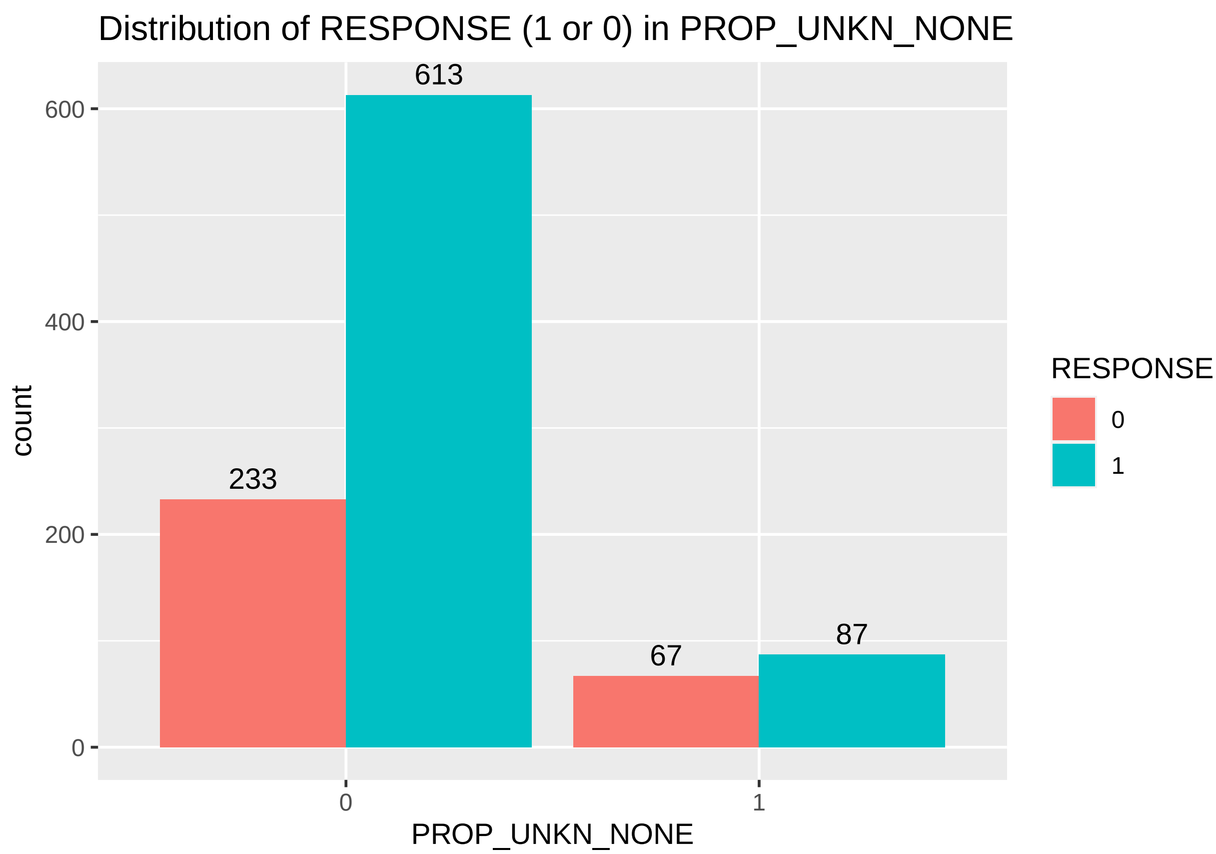

Distribution of RESPONSE (1 or 0) in CO_APPLICANT
count
600
400
200
0
677
282
18
23
0
1
CO_APPLICANT
RESPONSE
0
1

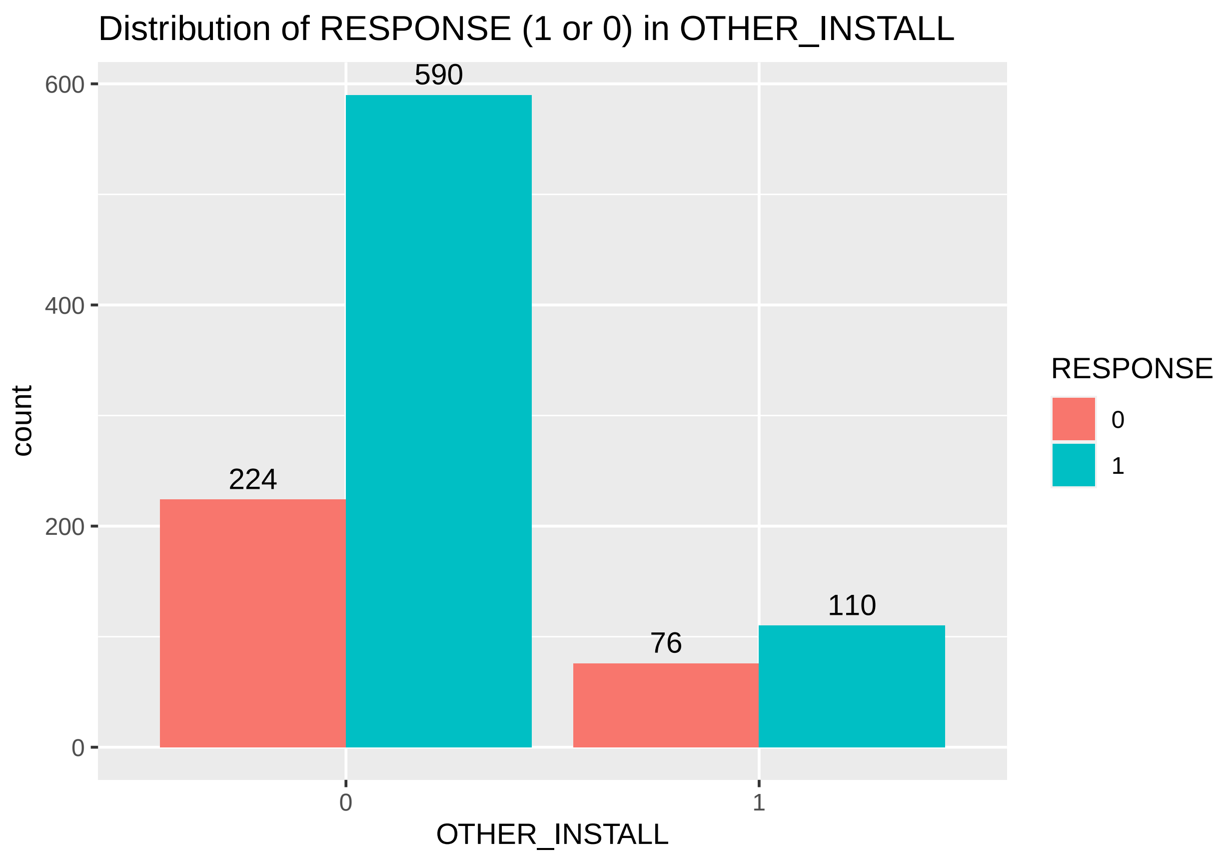

Distribution of RESPONSE (1 or 0) in GUARANTOR
658
290
10
42
600
400
200
0
count
0
1
GUARANTOR
RESPONSE
0
1



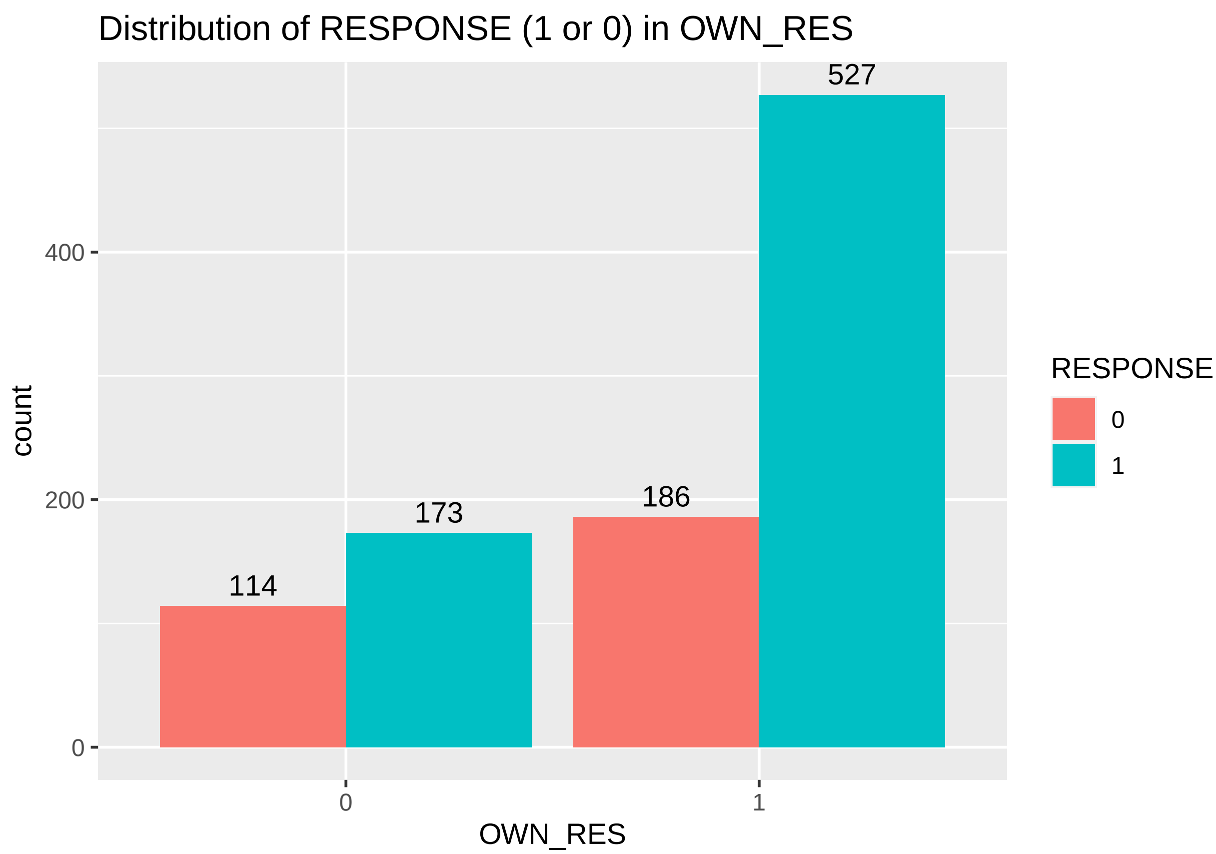
Distribution of RESPONSE (1 or 0) in PRESENT_RESIDENT
300
200
100
0
count
36
94
97
211
43
106
124
289
1
2
3
4
PRESENT_RESIDENT
RESPONSE
0
1

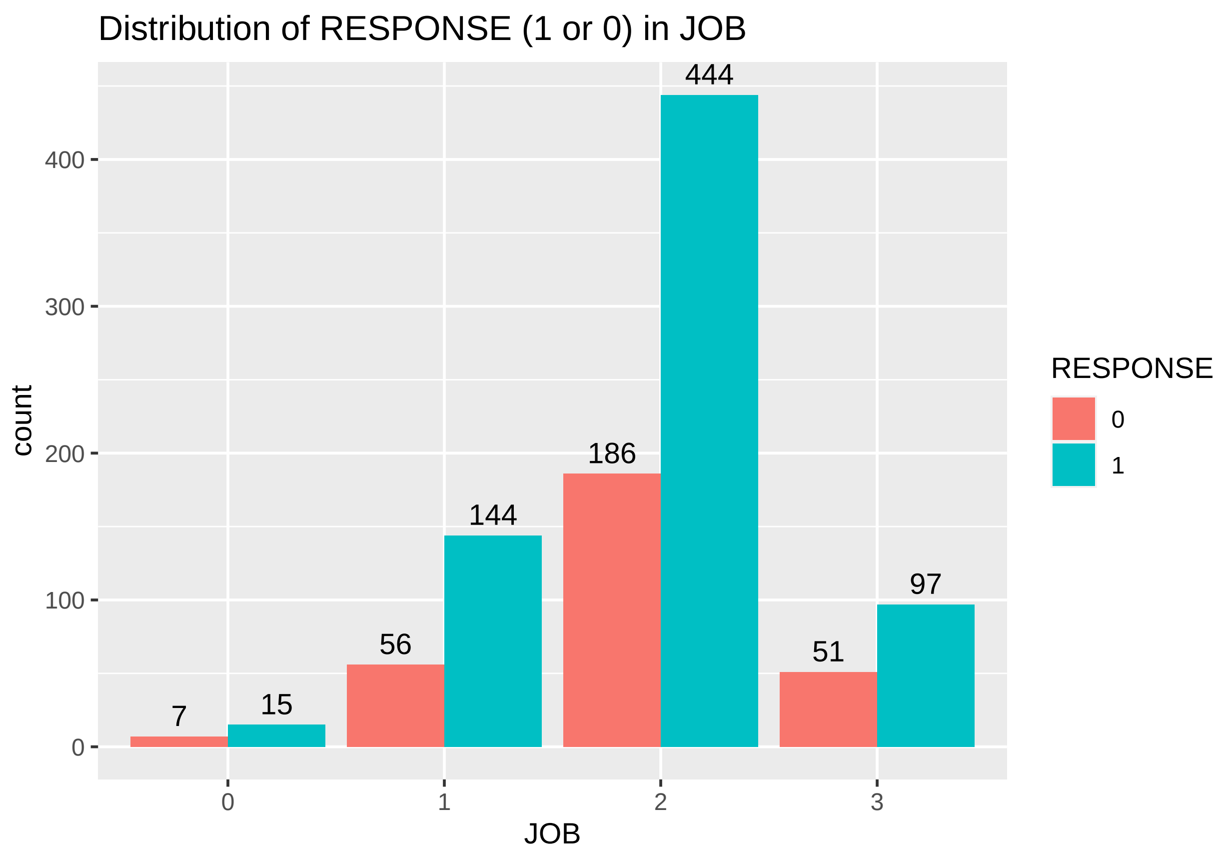

Distribution of RESPONSE (1 or 0) in REAL_ESTATE
count
REAL_ESTATE
RESPONSE
0
1
478
240
222
60
0
100
200
300
400
500


Distribution of RESPONSE (1 or 0) in PROP_UNKN_NONE
613
233
67
87
600
400
200
0
count
0
1
PROP_UNKN_NONE
RESPONSE
0
1

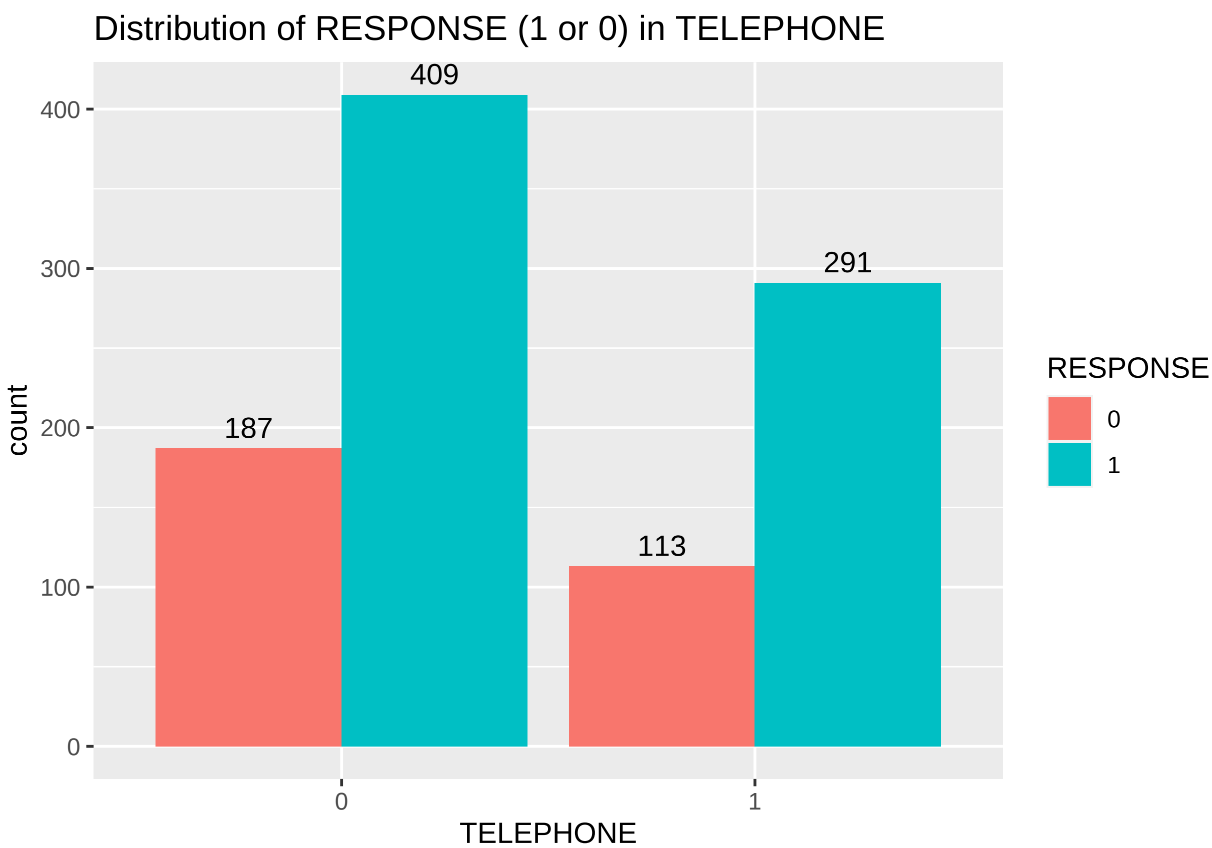

Distribution of RESPONSE (1 or 0) in OTHER_INSTALL
count
600
400
200
0
590
224
76
110
0
1
OTHER_INSTALL
RESPONSE
0
1

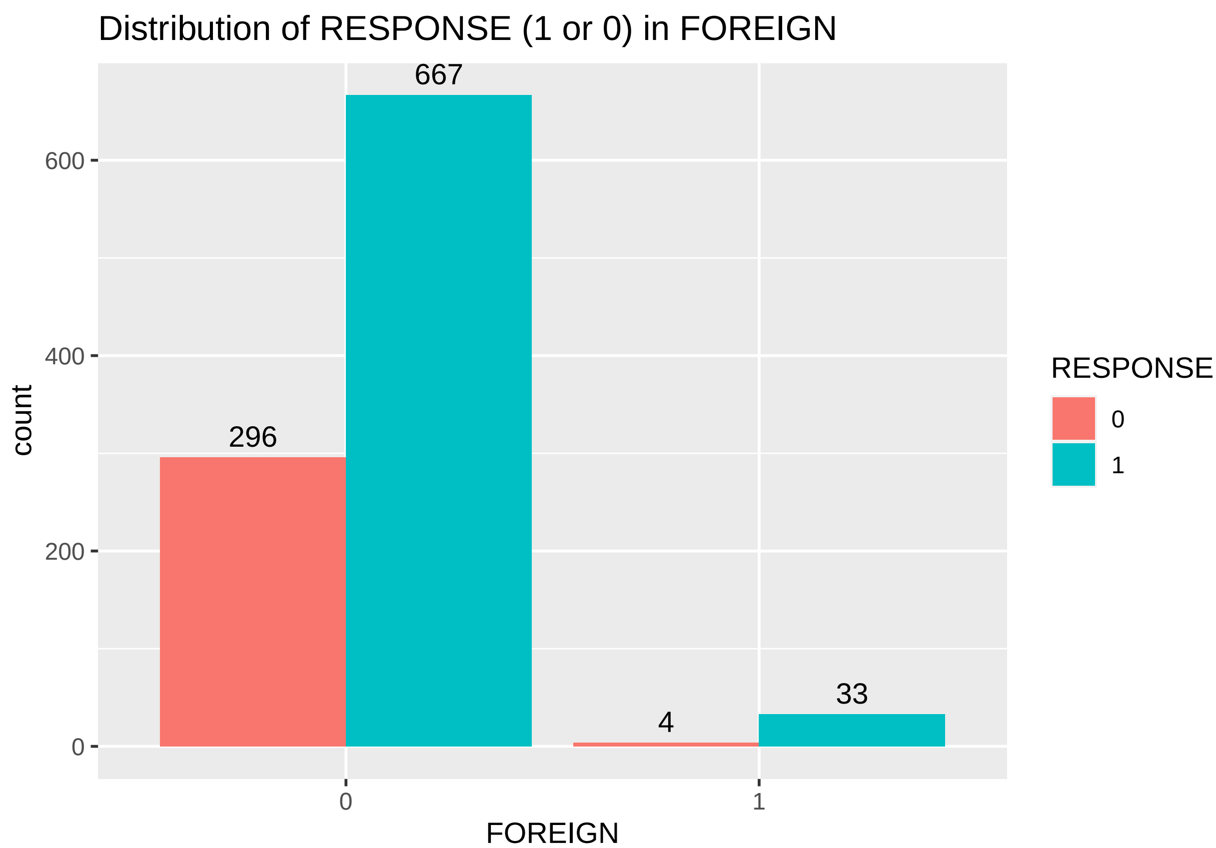

Distribution of RESPONSE (1 or 0) in RENT
591
230
70
109
600
400
200
0
count
0
1
RENT
RESPONSE
0
1


Distribution of RESPONSE (1 or 0) in OWN_RES
count
400
200
0
114
173
186
527
0
1
OWN_RES
RESPONSE
0
1



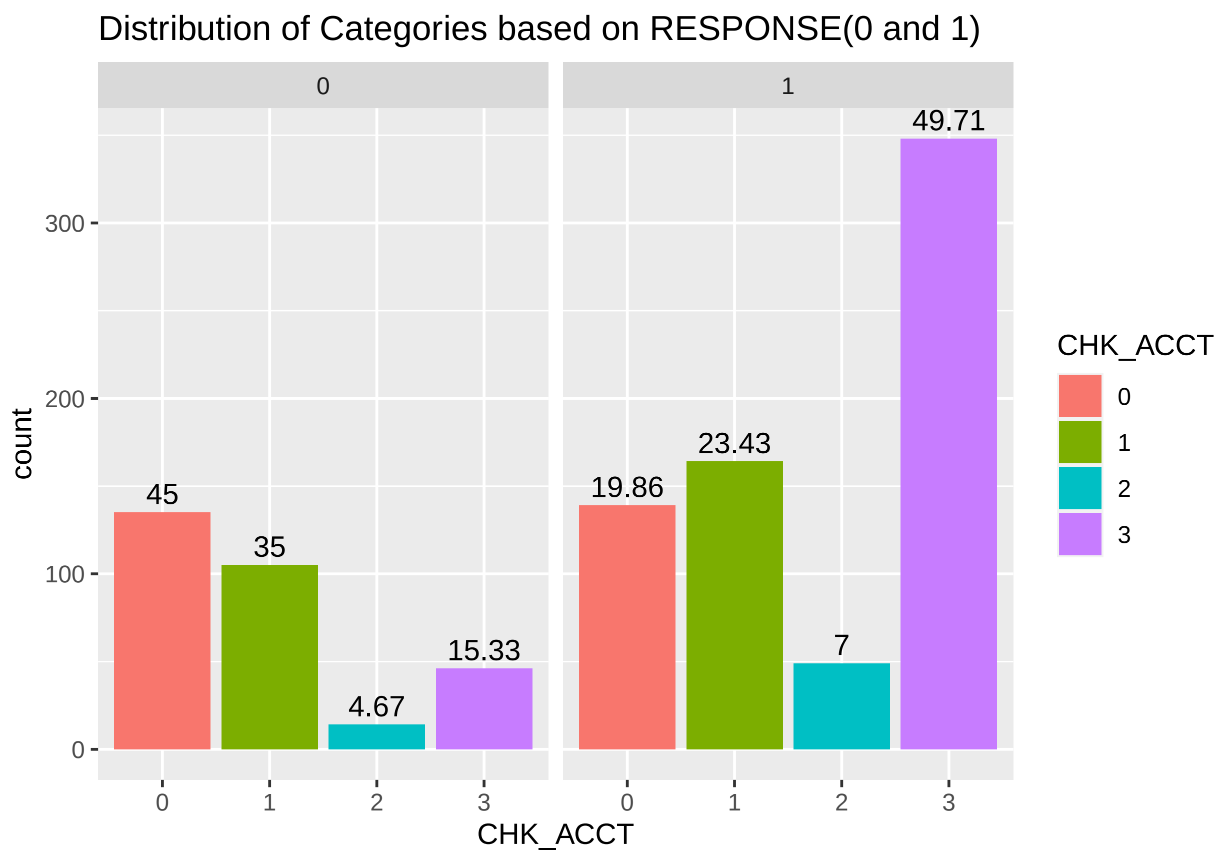
Distribution of RESPONSE (1 or 0) in JOB
count
400
300
200
100
0
7
15
56
144
186
444
51
97
0
1
2
3
JOB
RESPONSE
0
1


Distribution of RESPONSE (1 or 0) in TELEPHONE
409
291
187
113
400
300
200
100
0
count
0
1
TELEPHONE
RESPONSE
0
1

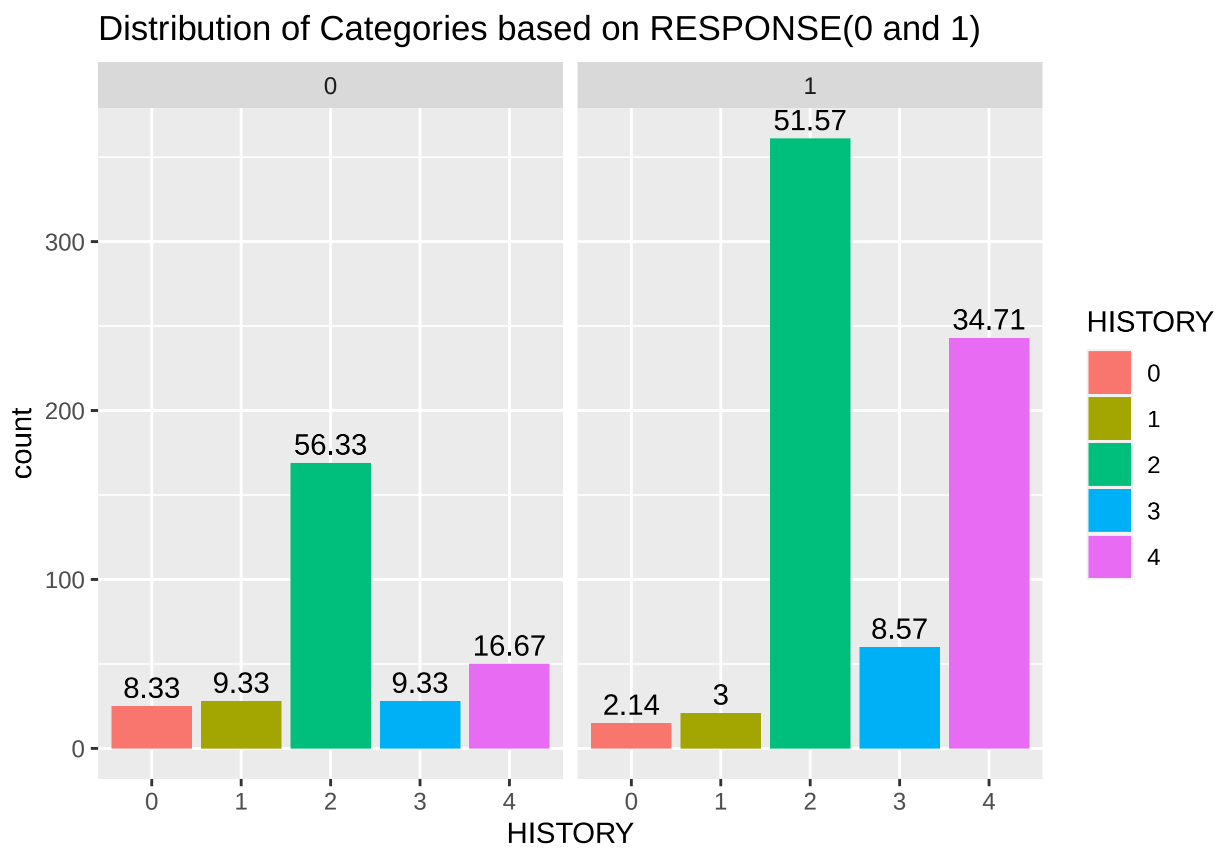


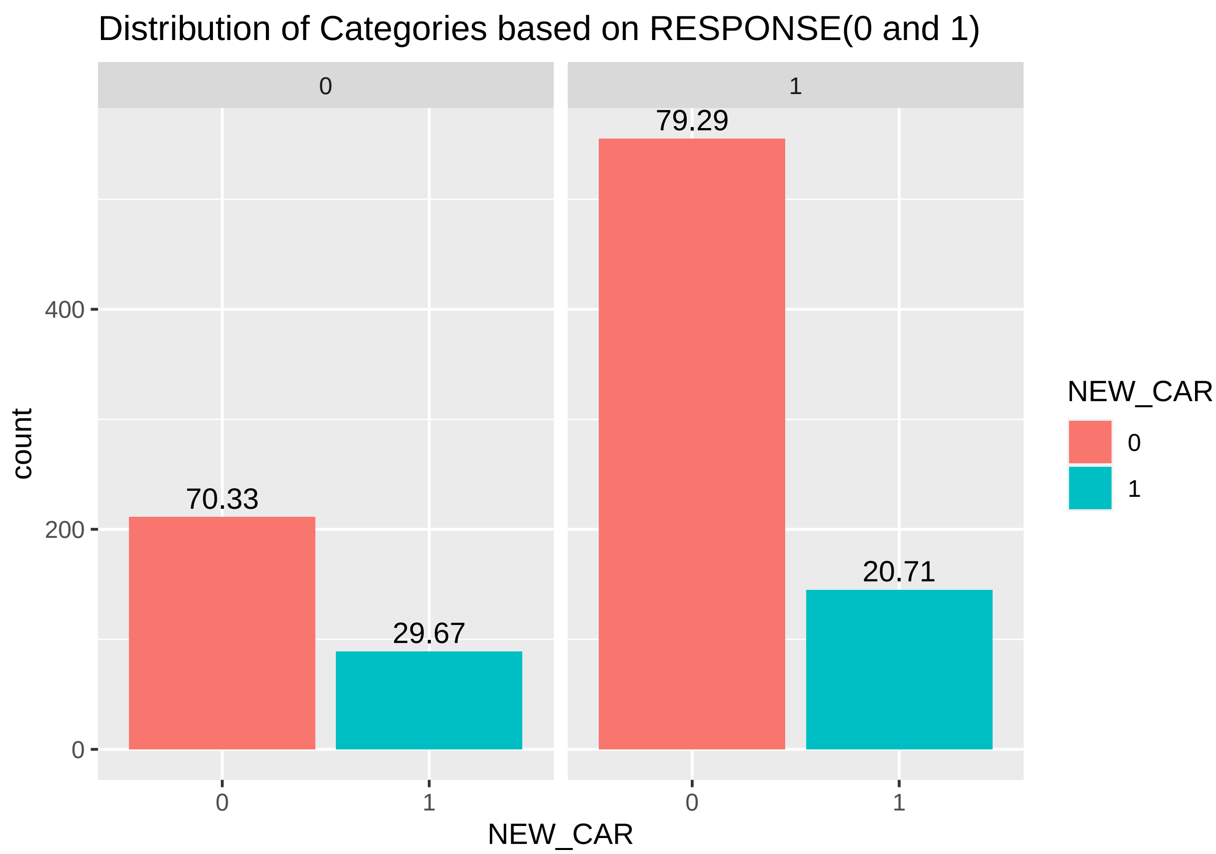
From the above plots we can check the proportion of 0 and 1 of Target in each category of the input variable. If the proportions are same across all categories then the input variable is not much significant in predicting the Target. In other words it won't affect our Target variable to a significant level.

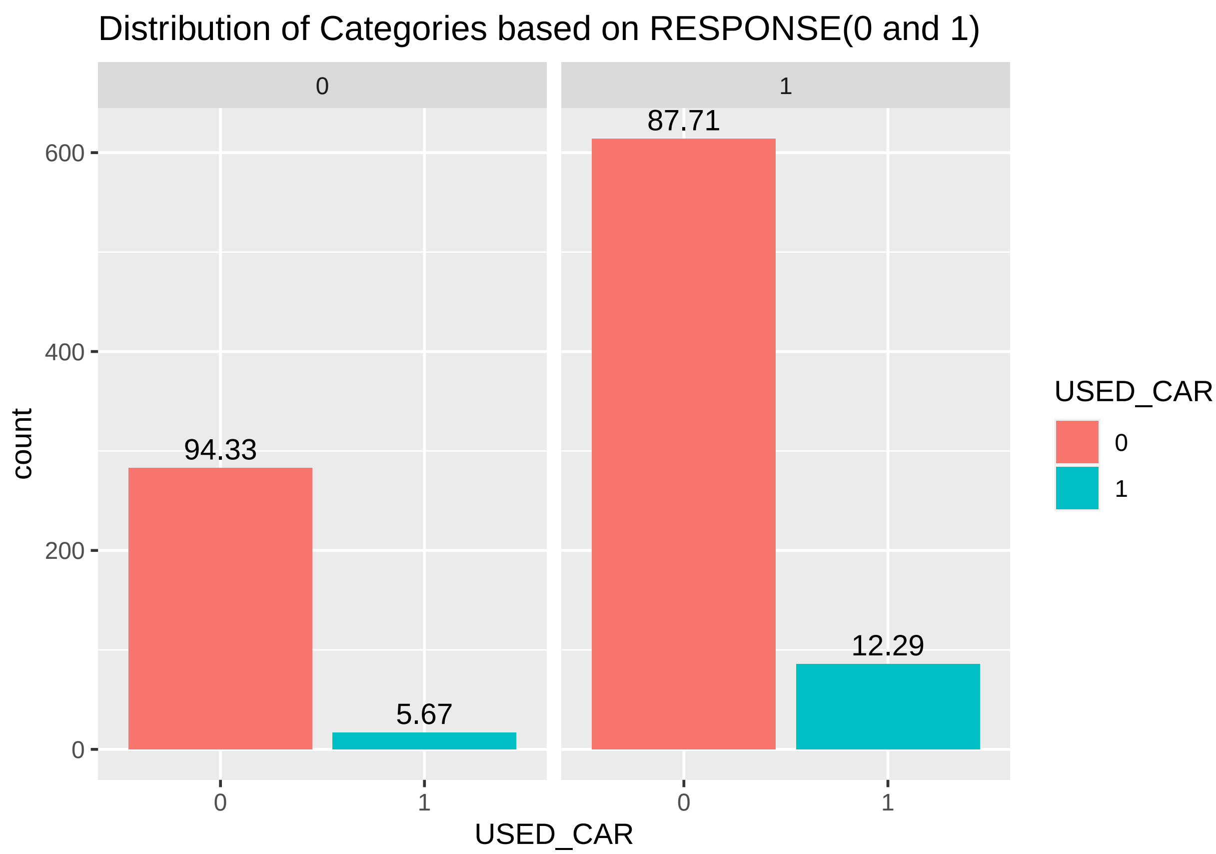
```
# Distribution of input variable categories in each Target category
for (i in c[2:25]){
  print(ggplot(df,aes_string(x=i, fill=i))+geom_bar(position="dodge")+
        geom_text(aes(label=round(after_stat(prop*100),2), group=1),stat='count',
         size=4,position = position_dodge(0.9), vjust=-0.4) + facet_wrap('RESPONSE')
        + ggtitle("Distribution of Categories based on RESPONSE(0 and 1)"))
}
```

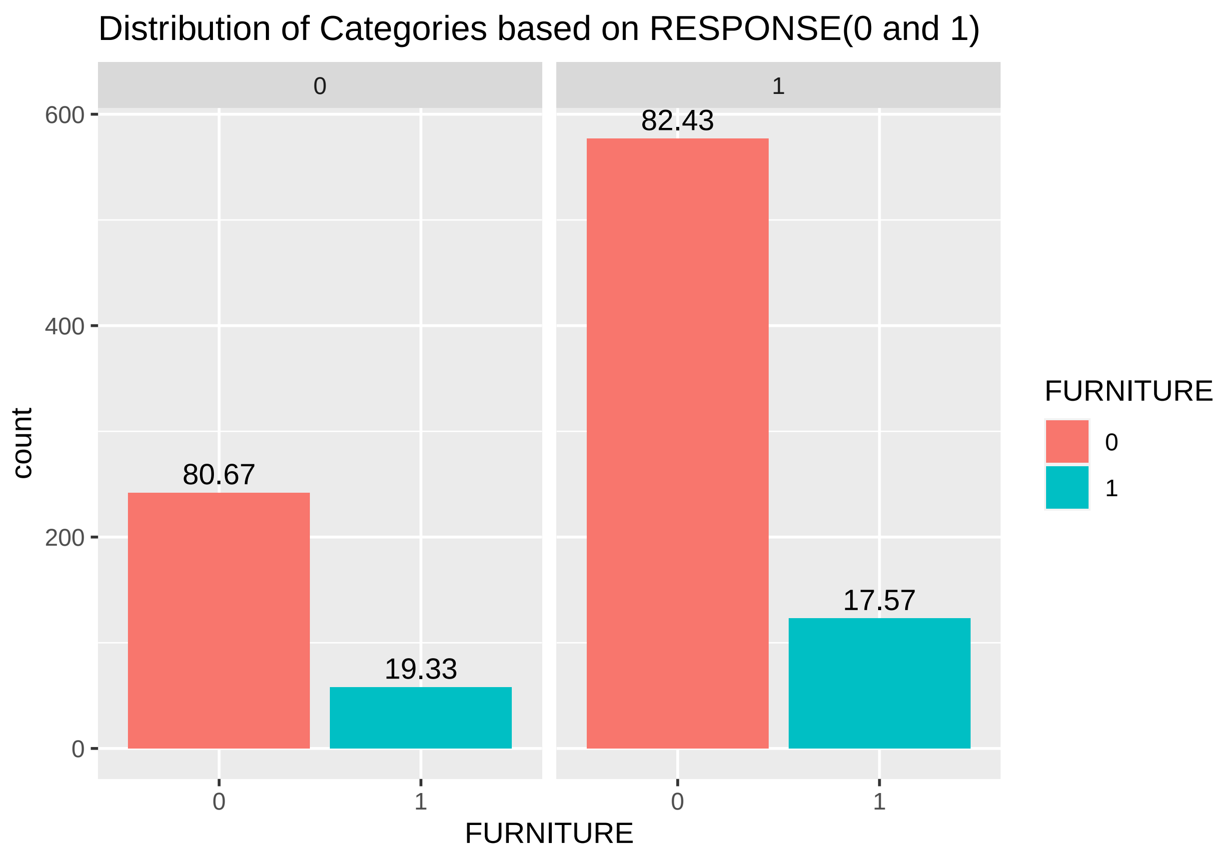


Distribution of Categories based on RESPONSE(0 and 1)
0
1
45
35
4.67
15.33
19.86
23.43
7
49.71
count
300
200
100
0
0
1
2
3
0
1
2
3
CHK_ACCT
CHK_ACCT
0
1
2
3

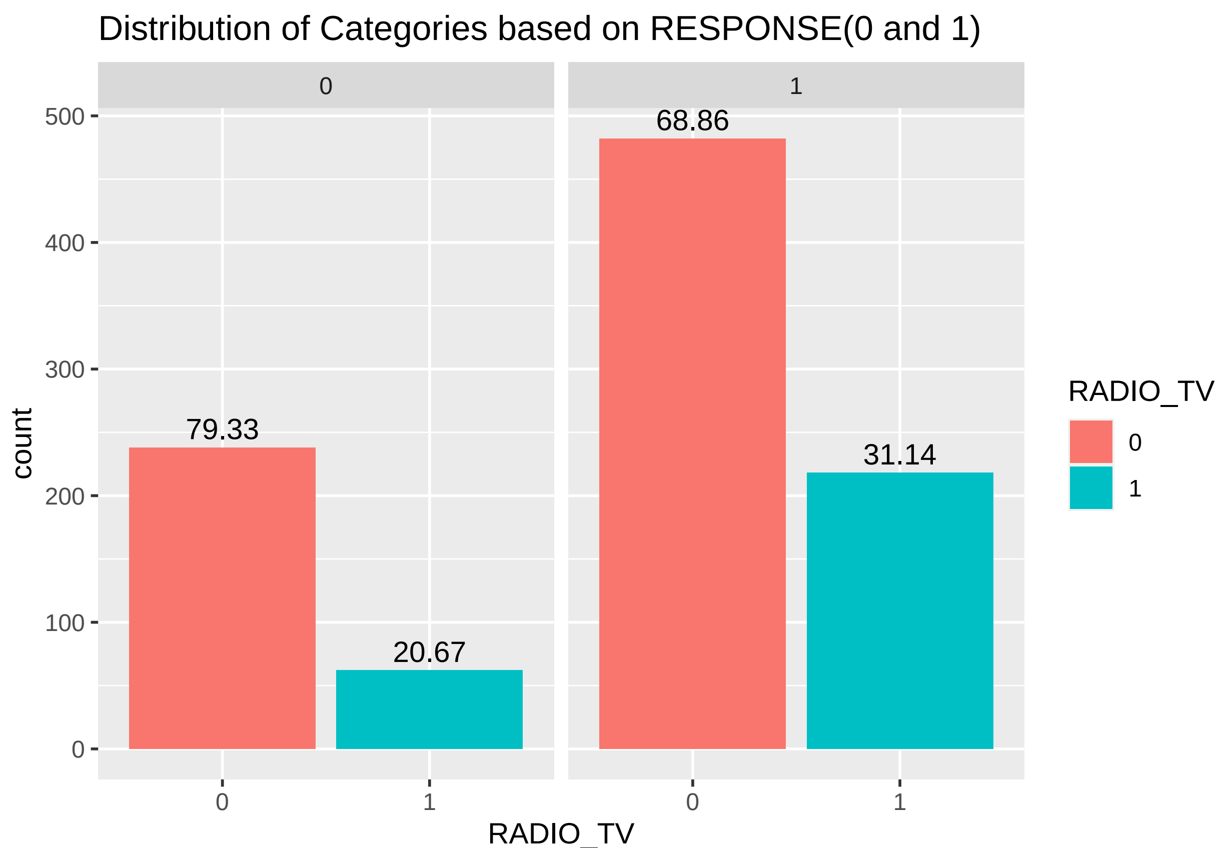

Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
300
200
100
0
56.33
8.33
9.33
9.33
16.67
51.57
34.71
2.14
3
8.57
0
1
2
3
4
HISTORY
HISTORY
0
1
2
3
4

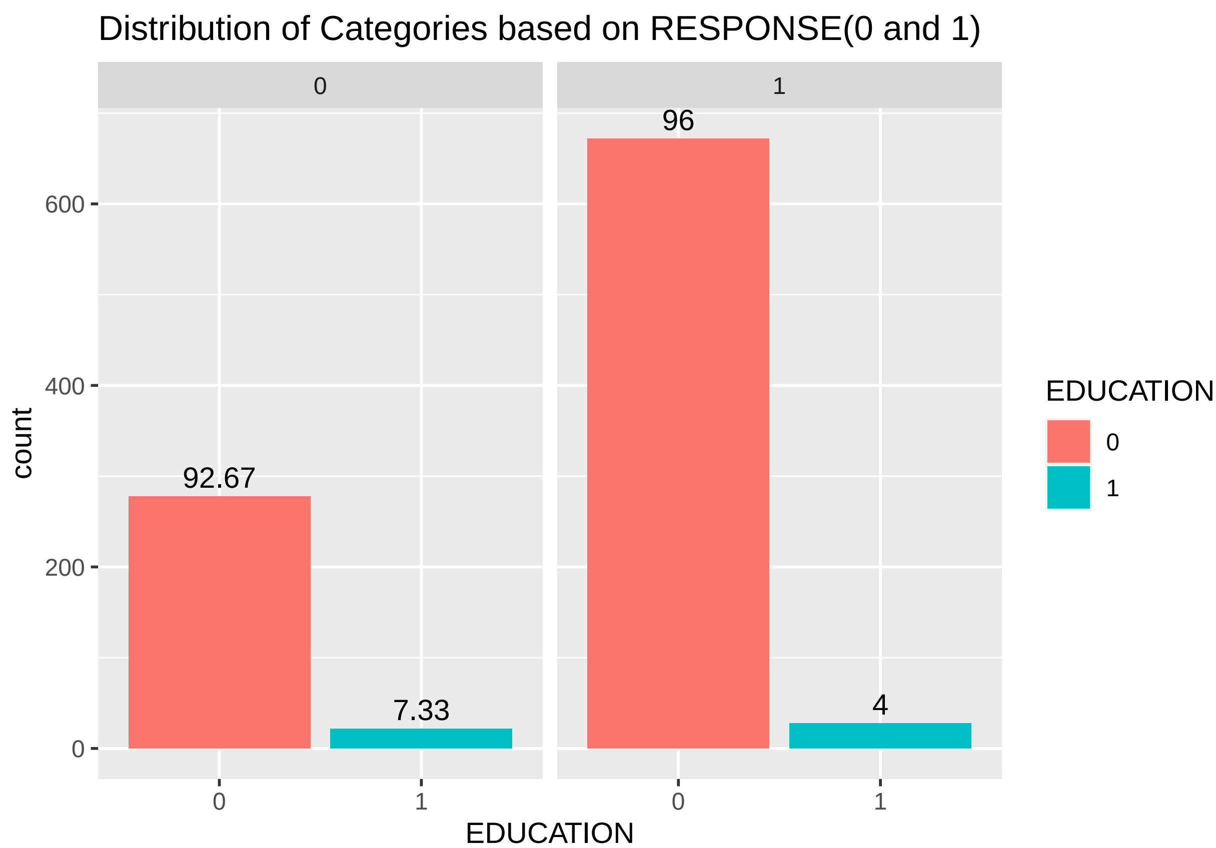

Distribution of Categories based on RESPONSE(0 and 1)
0
1
79.29
70.33
29.67
20.71
400
200
0
count
0
1
0
1
NEW_CAR
NEW_CAR
0
1

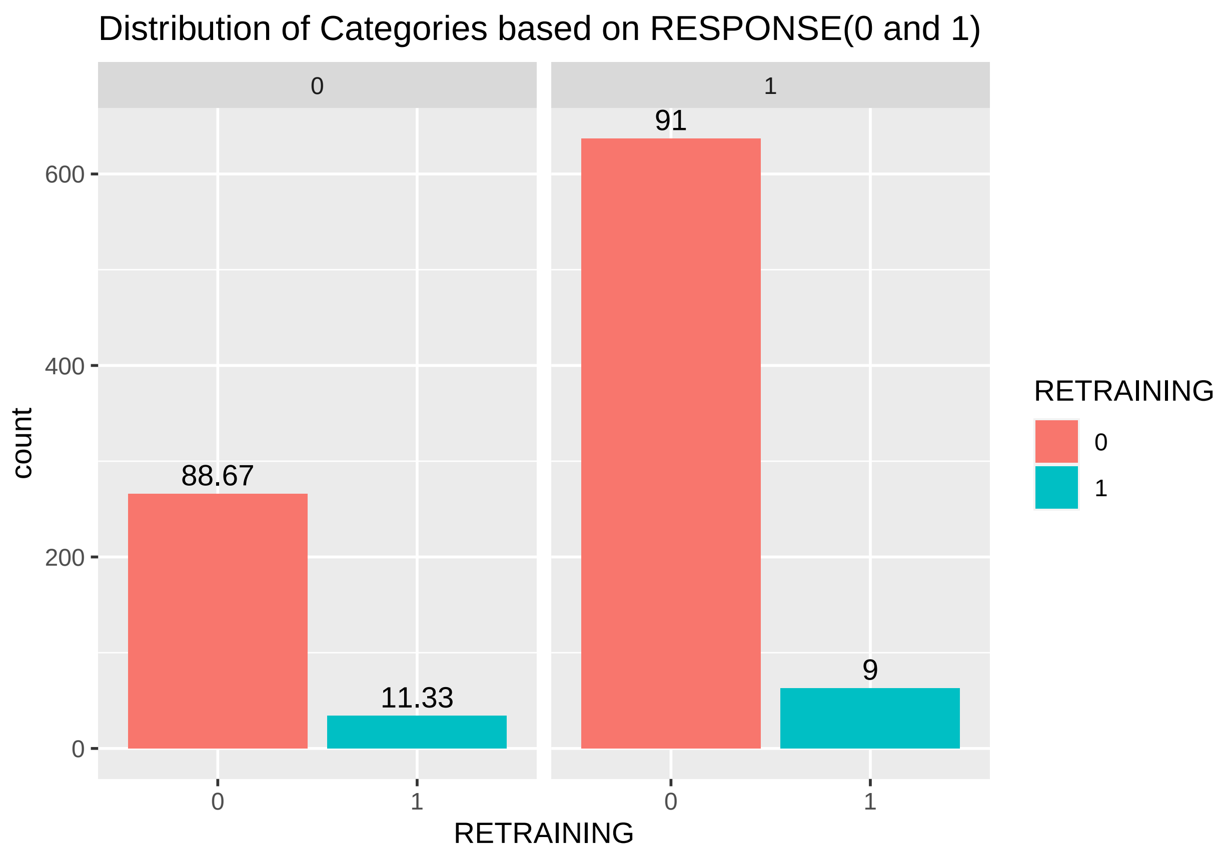

Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
600
400
200
0
94.33
5.67
87.71
12.29
USED_CAR
USED_CAR
0
1

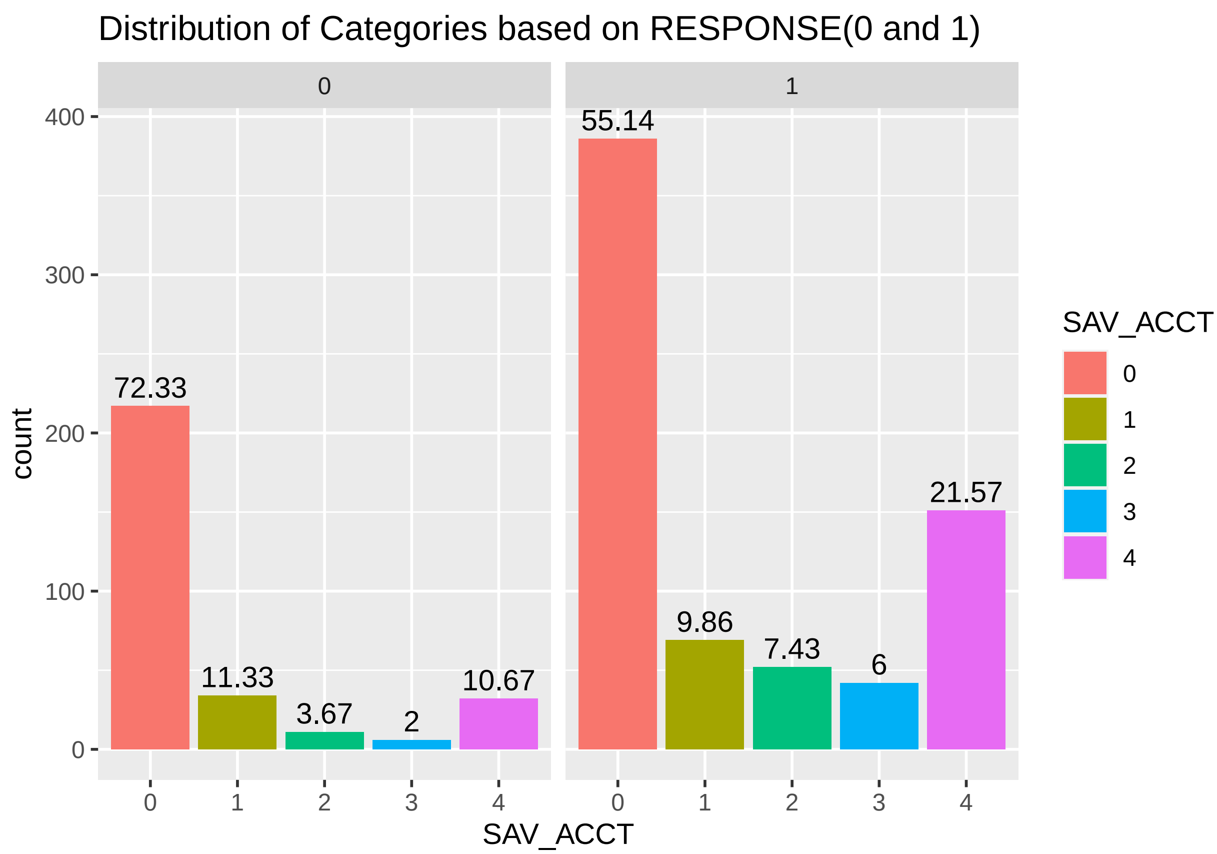


Distribution of Categories based on RESPONSE(0 and 1)
0
1
600
400
200
0
count
82.43
80.67
19.33
17.57
0
1
0
1
FURNITURE
FURNITURE
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
500
400
300
200
100
0
count
79.33
20.67
68.86
31.14
0
1
0
1
RADIO_TV
RADIO_TV
0
1

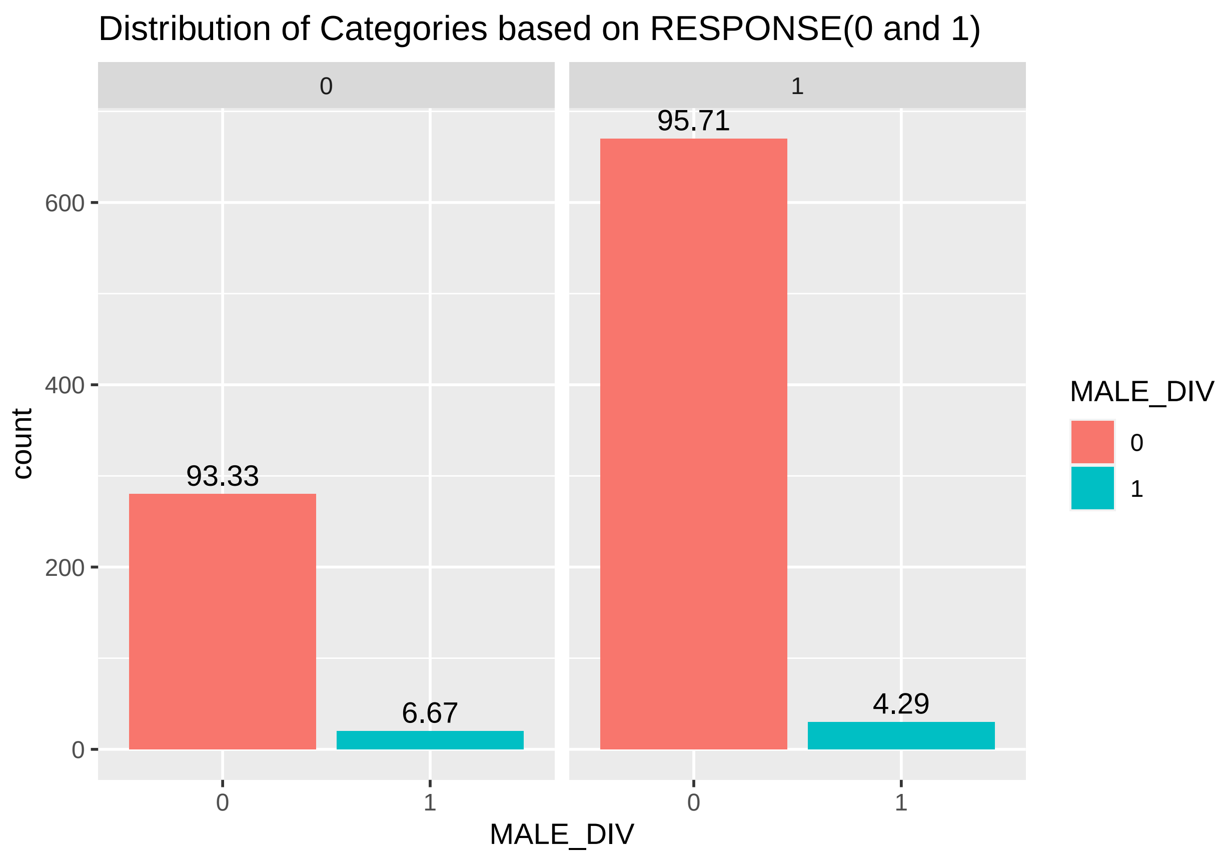

Distribution of Categories based on RESPONSE(0 and 1)
0
1
96
600
400
count
92.67
200
7.33
4
0
0
1
0
1
EDUCATION
EDUCATION
0
1

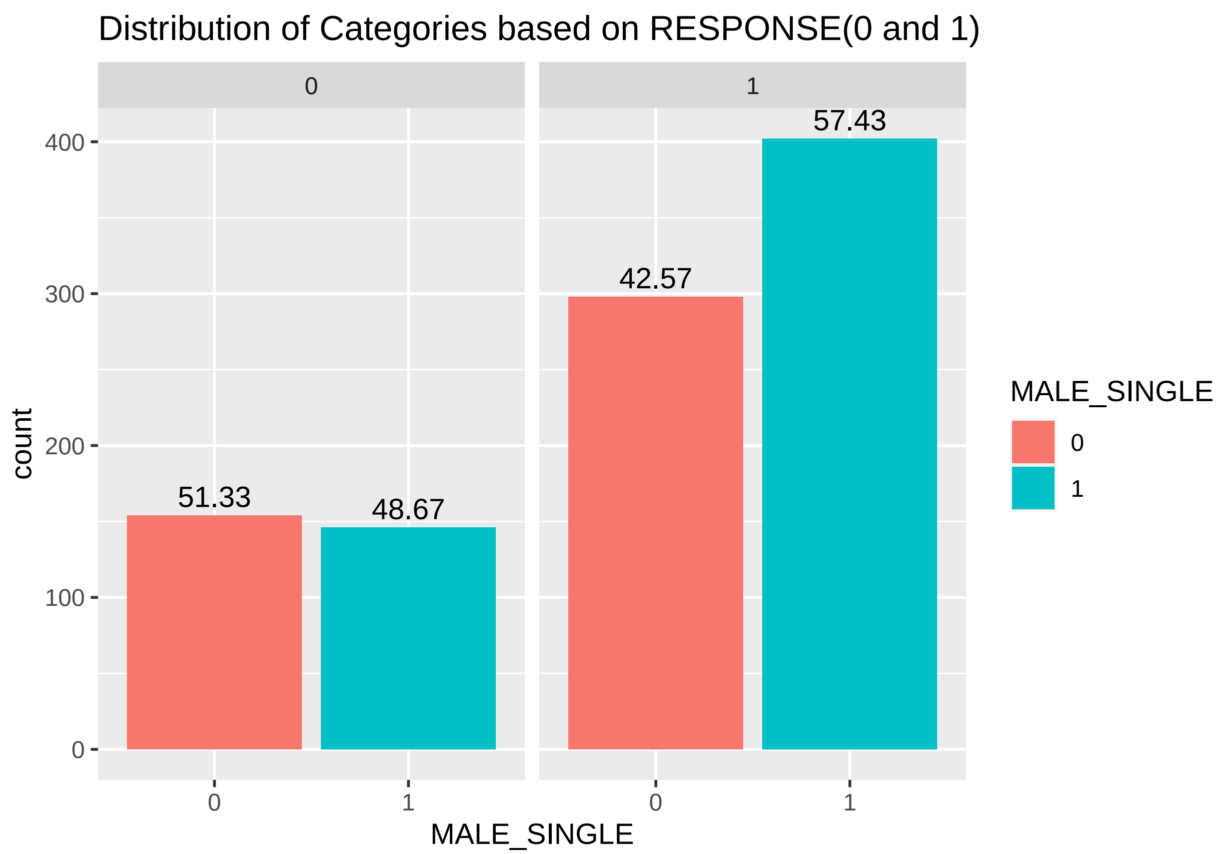

Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
600
400
200
0
88.67
11.33
91
9
RETRAINING
RETRAINING
0
1

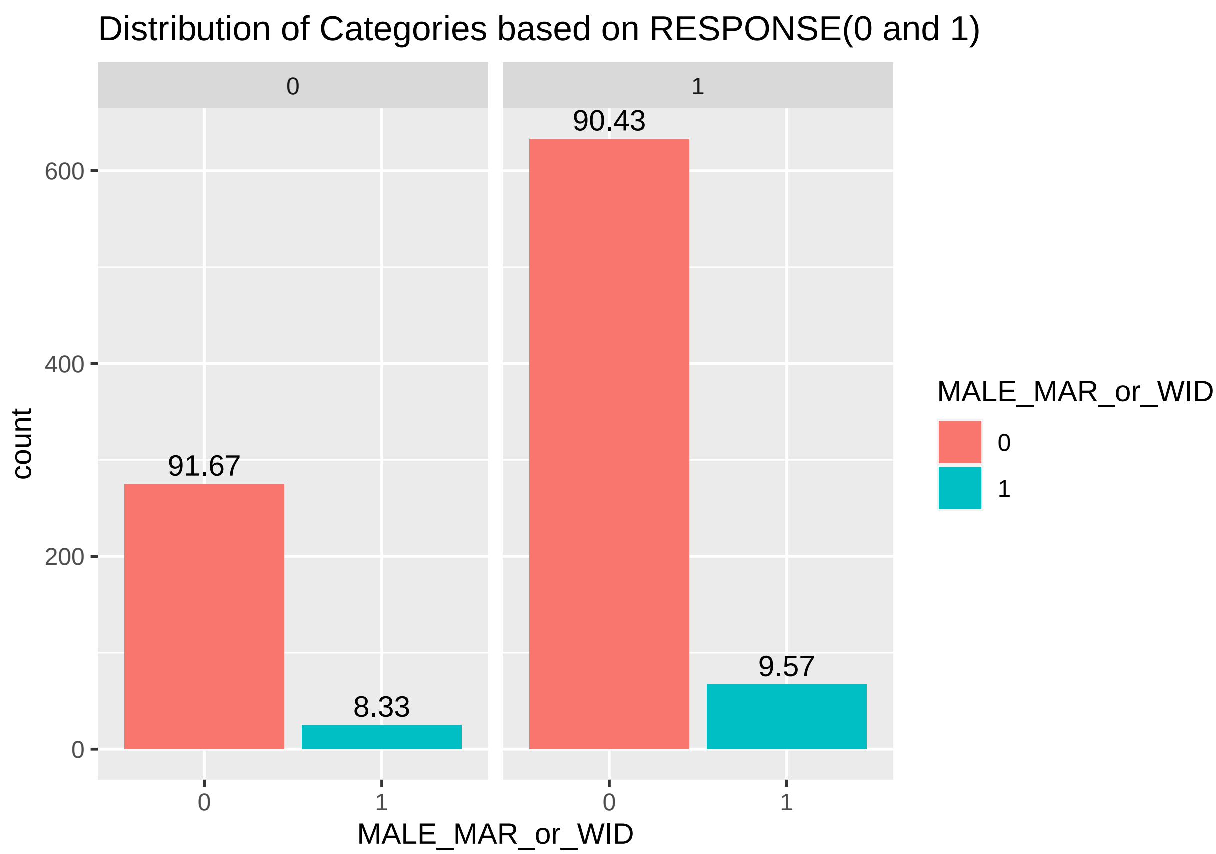


Distribution of Categories based on RESPONSE(0 and 1)
0
1
400
300
200
100
0
count
72.33
11.33
3.67
2
10.67
55.14
9.86
7.43
6
21.57
0
1
2
3
4
SAV_ACCT
SAV_ACCT
0
1
2
3
4

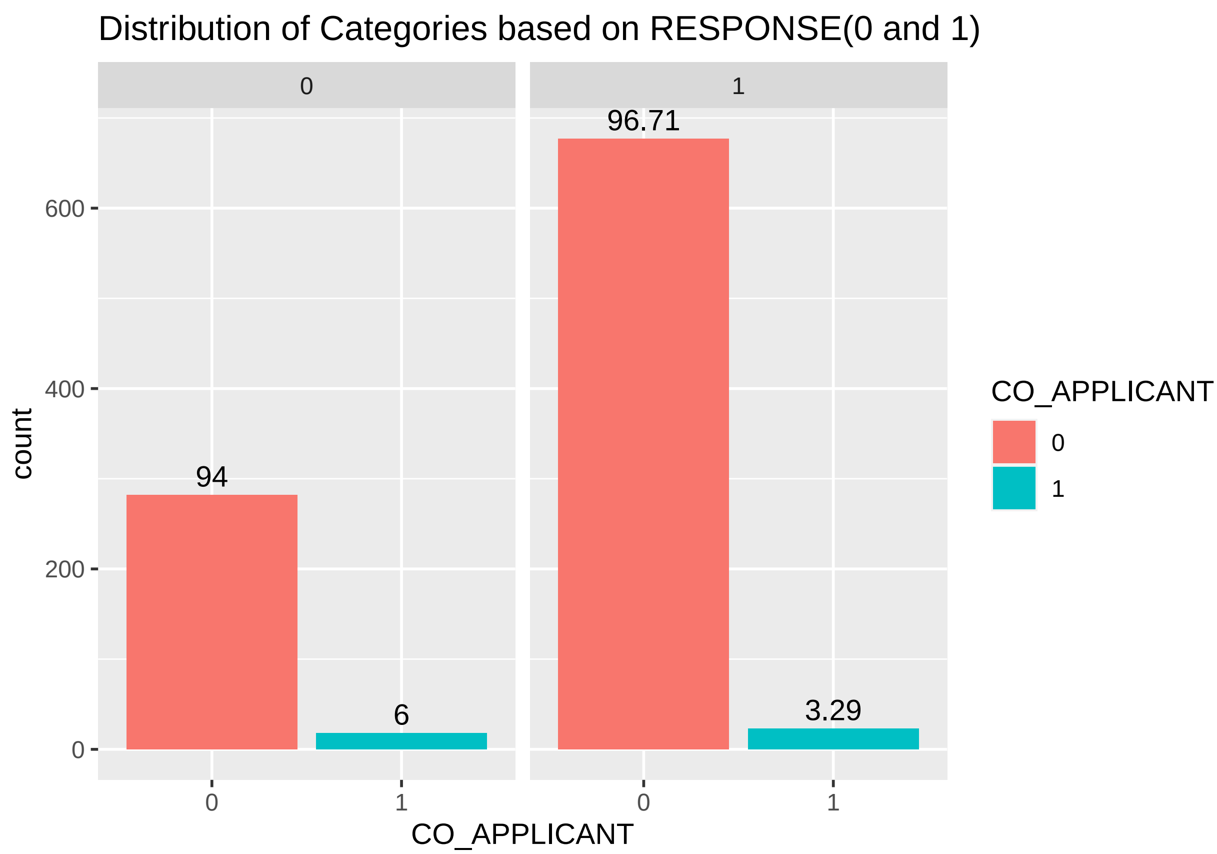

Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
200
150
100
50
0
7.67
23.33
34.67
13
21.33
5.57
14.57
33.57
19.29
27
0
1
2
3
4
0
1
2
3
4
EMPLOYMENT
EMPLOYMENT
0
1
2
3
4


Distribution of Categories based on RESPONSE(0 and 1)
0
1
95.71
93.33
6.67
4.29
600
400
200
0
count
0
1
0
1
MALE_DIV
MALE_DIV
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
400
300
200
100
0
51.33
48.67
42.57
57.43
0
1
0
1
MALE_SINGLE
MALE_SINGLE
0
1

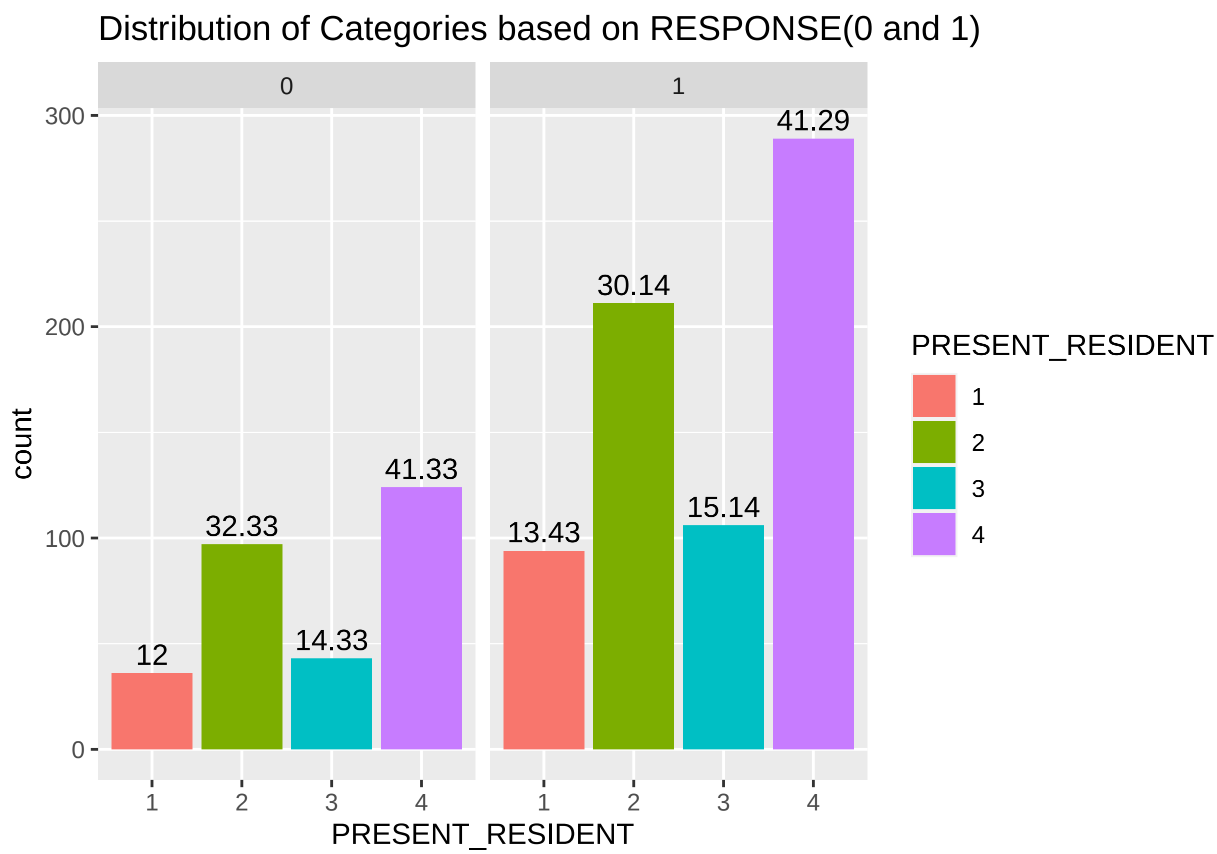


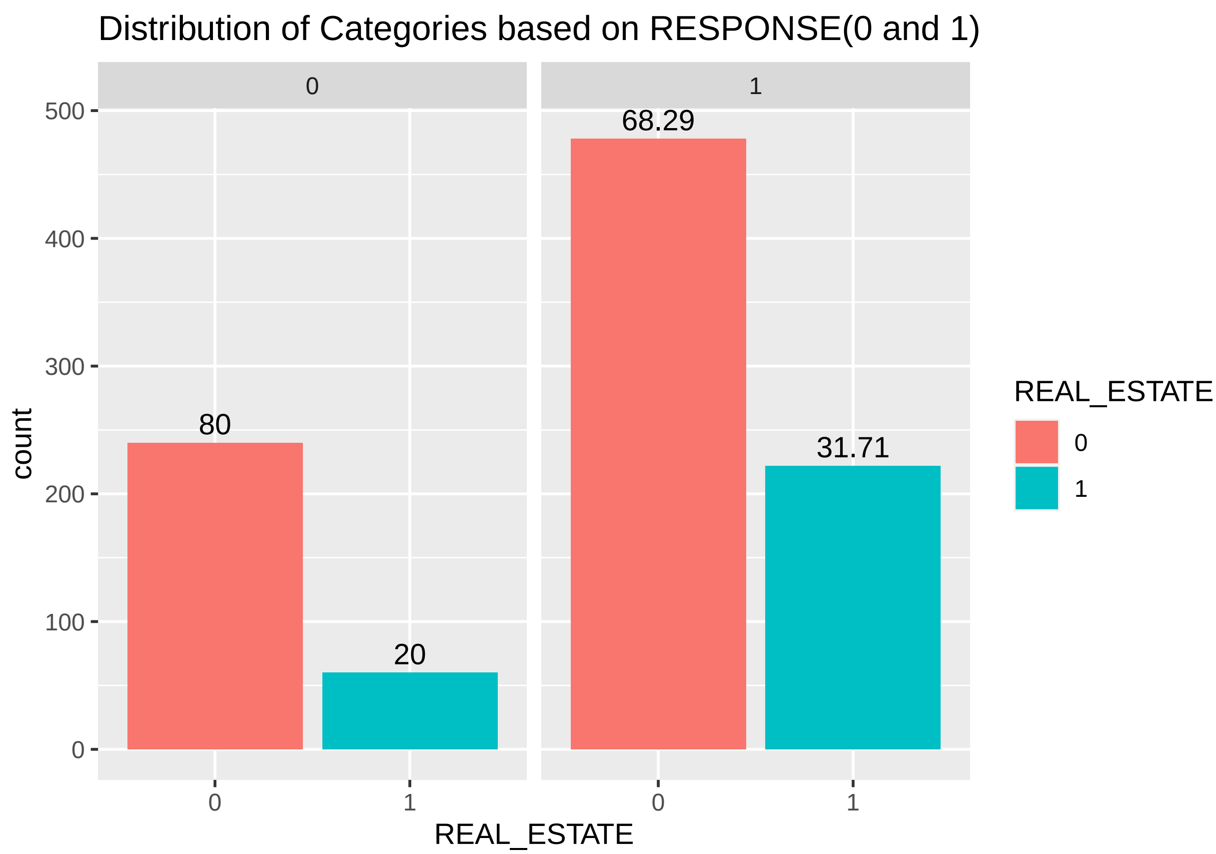
Distribution of Categories based on RESPONSE(0 and 1)
0
1
90.43
91.67
8.33
9.57
600
400
200
0
count
0
1
0
1
MALE_MAR_or_WID
MALE_MAR_or_WID
0
1



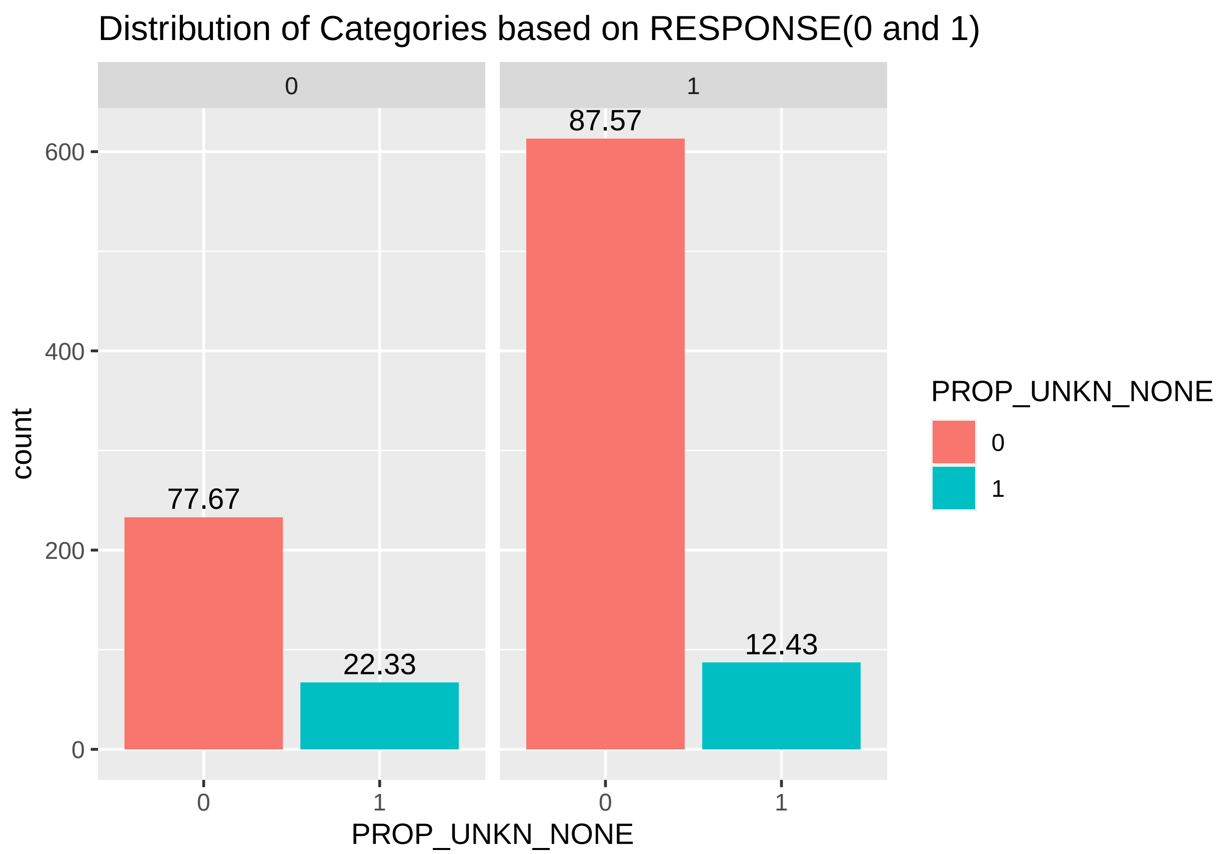
Distribution of Categories based on RESPONSE(0 and 1)
0
1
96.71
94
6
3.29
600
400
200
0
count
0
1
0
1
CO_APPLICANT
CO_APPLICANT
0
1

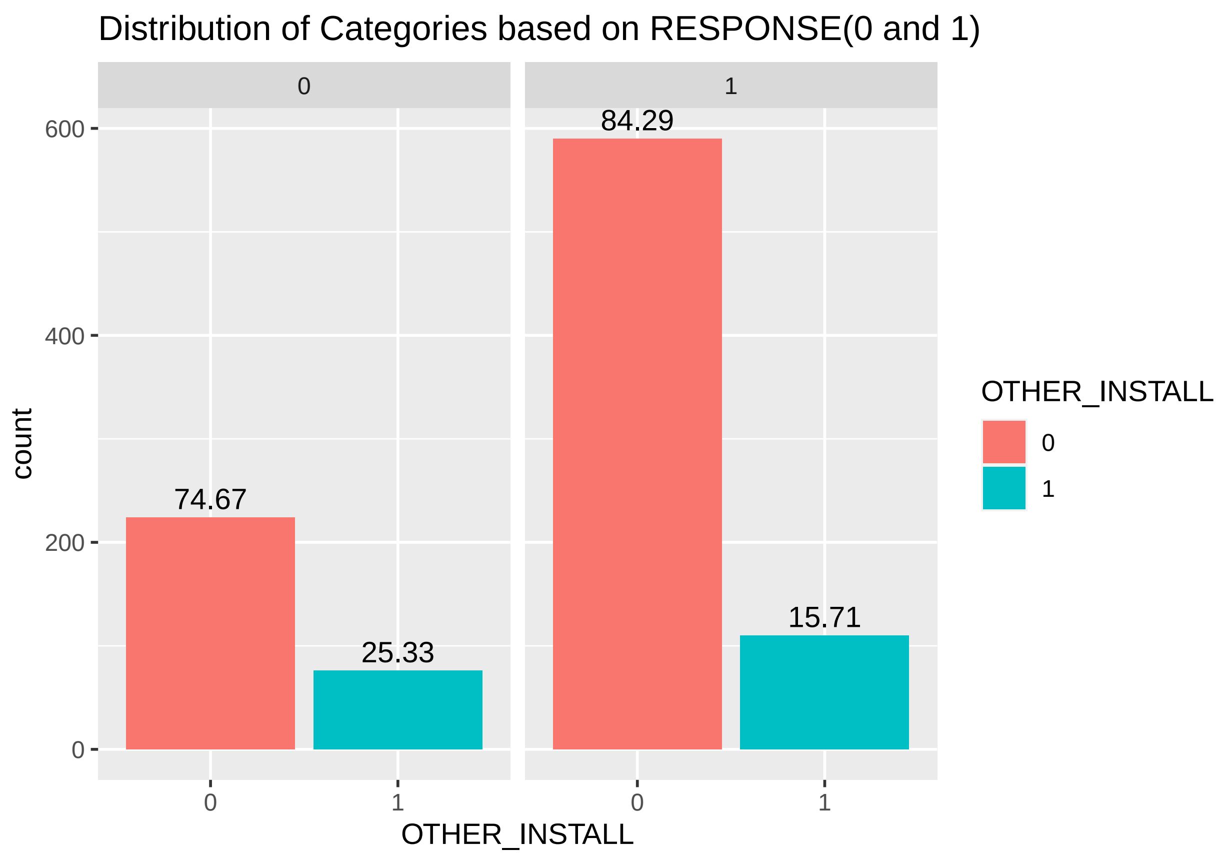

Distribution of Categories based on RESPONSE(0 and 1)
0
1
94
600
400
96.67
count
200
6
3.33
0
0
1
0
1
GUARANTOR
GUARANTOR
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
300
200
100
0
count
12
32.33
14.33
41.33
13.43
30.14
15.14
41.29
1
2
3
4
1
2
3
4
PRESENT_RESIDENT
PRESENT_RESIDENT
1
2
3
4

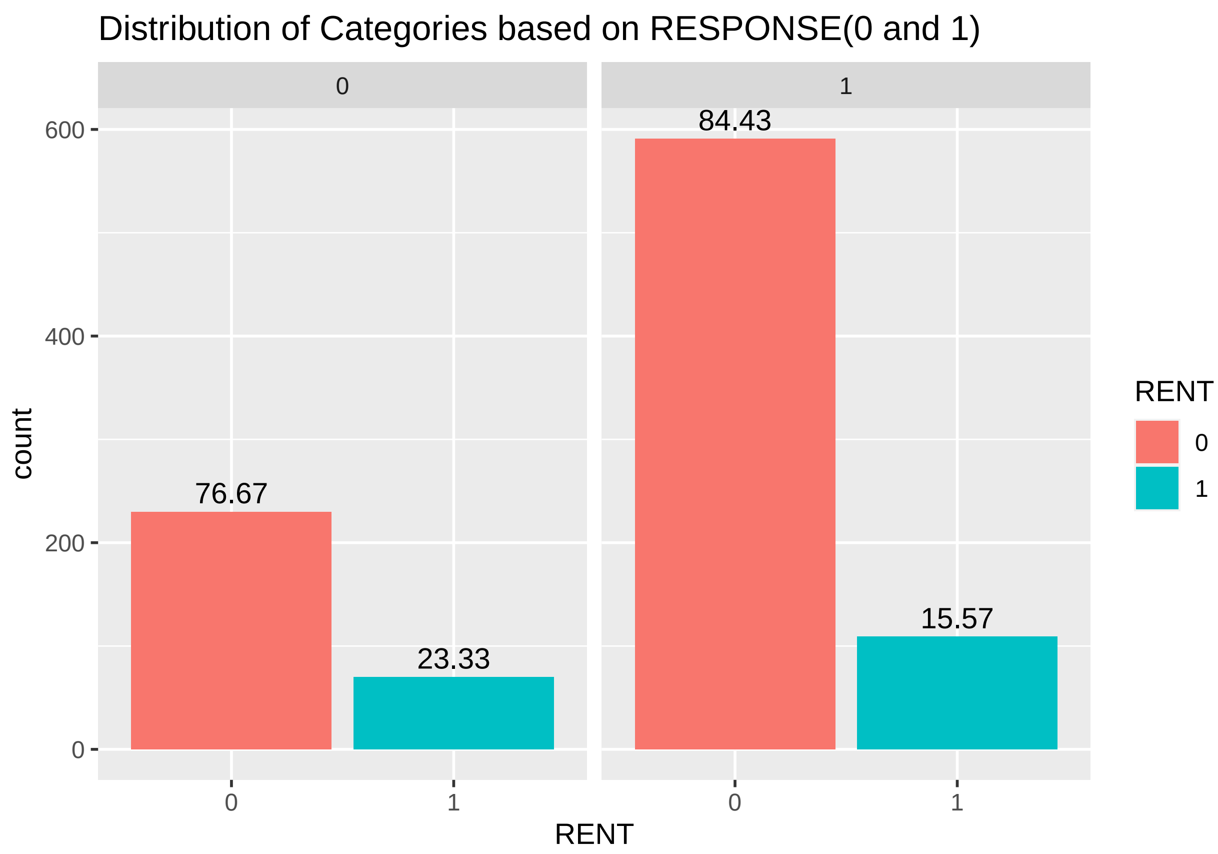


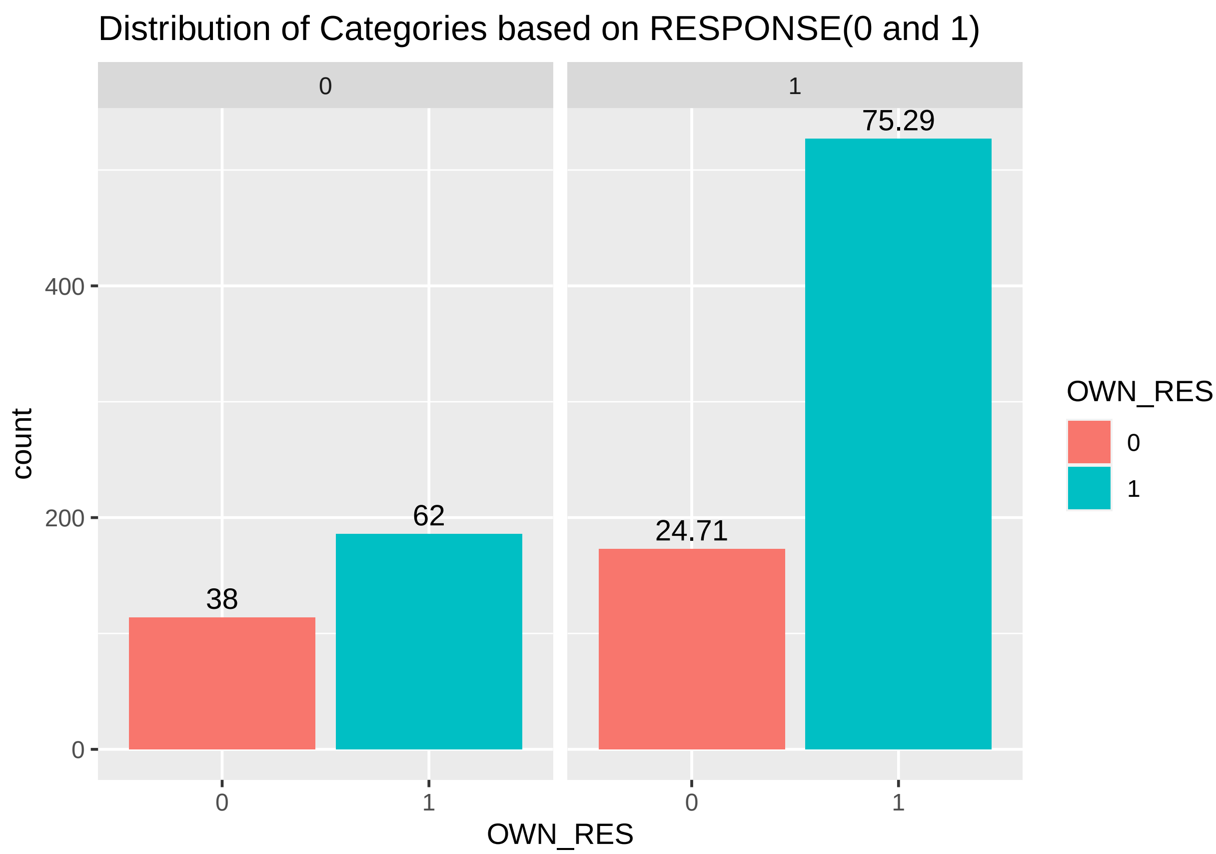
Distribution of Categories based on RESPONSE(0 and 1)
0
1
500
400
300
200
100
0
count
80
20
68.29
31.71
0
1
0
1
REAL_ESTATE
REAL_ESTATE
0
1

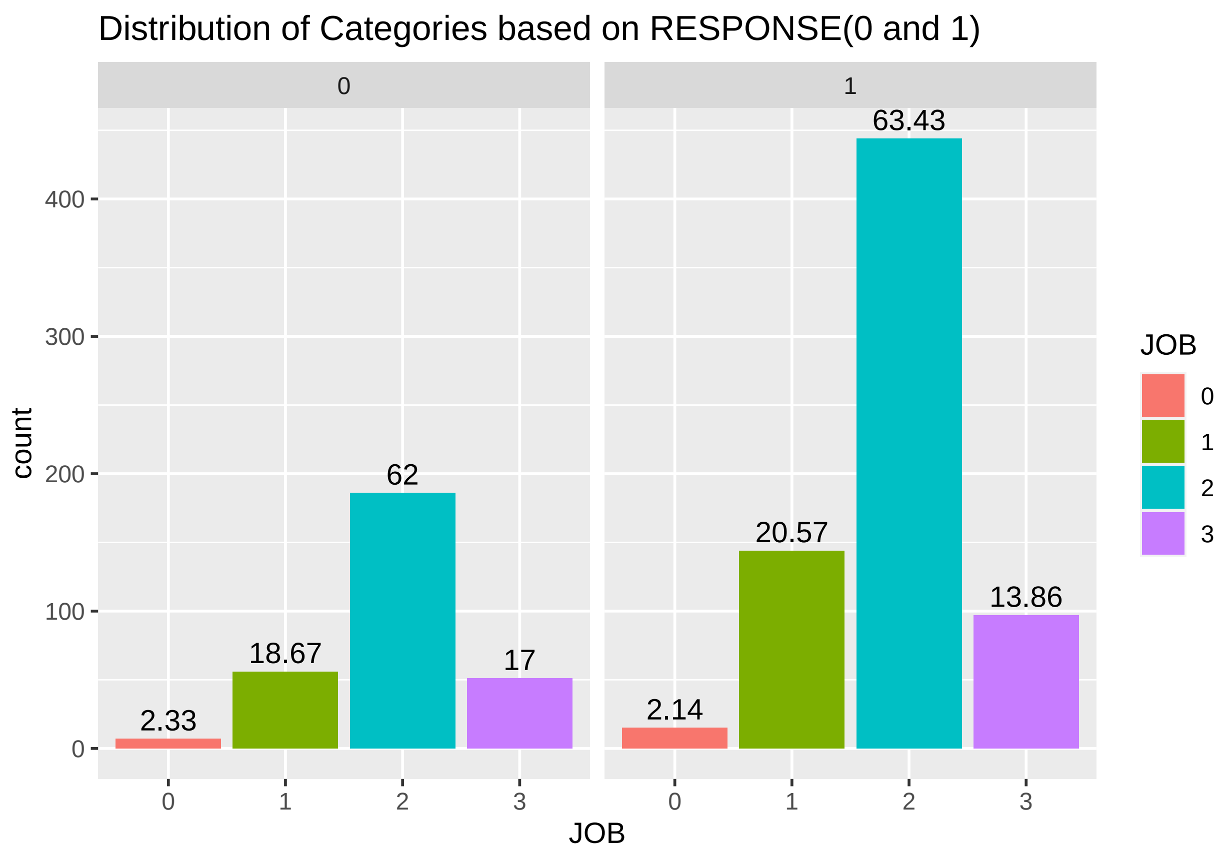

Distribution of Categories based on RESPONSE(0 and 1)
0
1
600
400
200
0
count
87.57
77.67
22.33
12.43
0
1
0
1
PROP_UNKN_NONE
PROP_UNKN_NONE
0
1

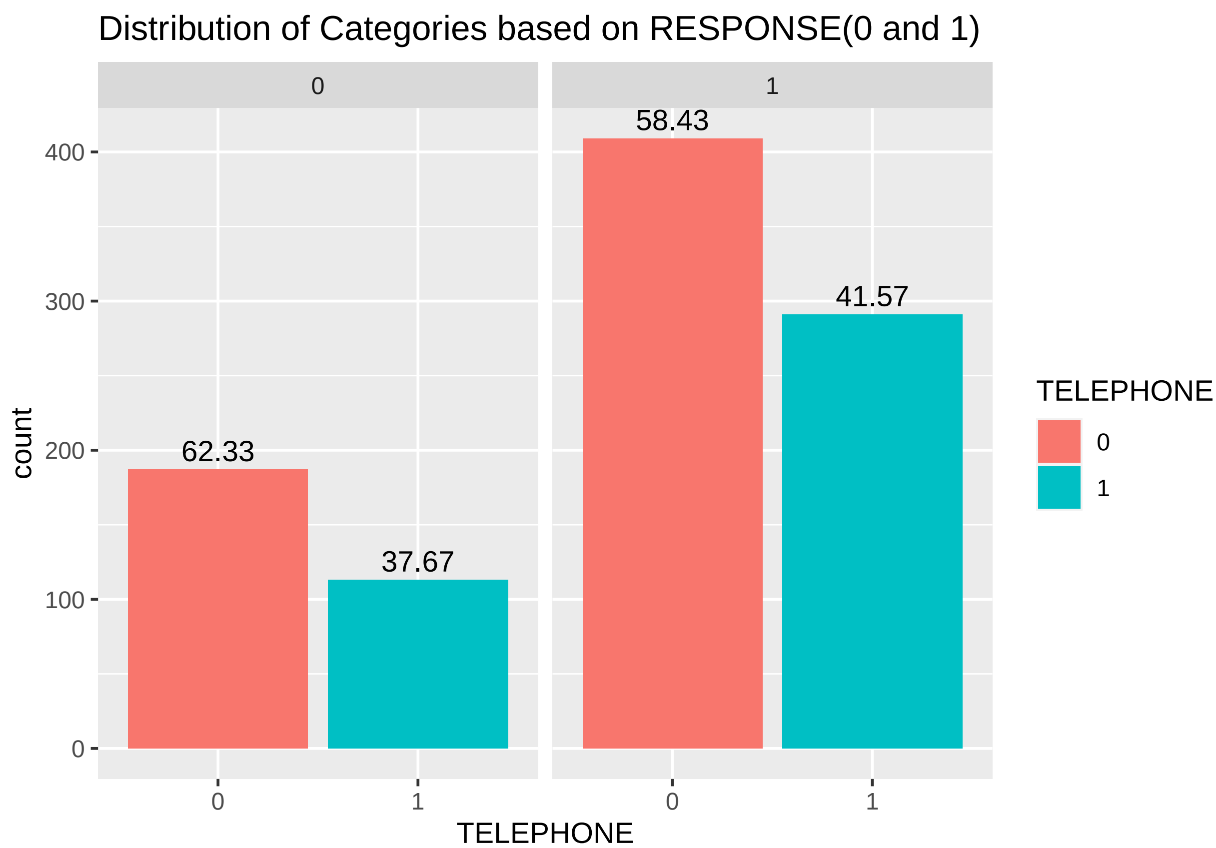


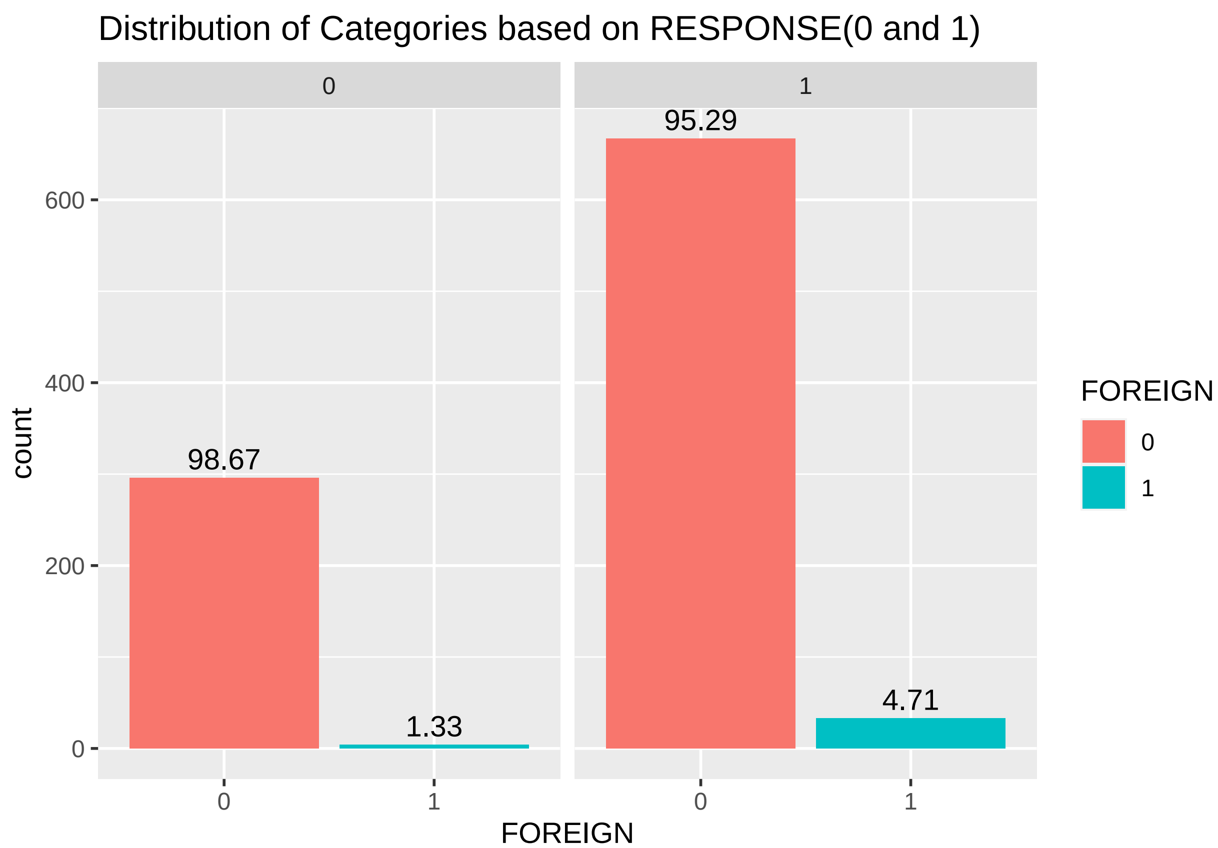
Distribution of Categories based on RESPONSE(0 and 1)
0
1
600
400
200
0
count
74.67
25.33
84.29
15.71
0
1
0
1
OTHER_INSTALL
OTHER_INSTALL
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
600
400
200
0
count
76.67
23.33
84.43
15.57
0
1
0
1
RENT
RENT
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
75.29
400
200
0
count
62
38
24.71
0
1
0
1
OWN_RES
OWN_RES
0
1


Distribution of Categories based on RESPONSE(0 and 1)
0
1
count
400
300
200
100
0
2.33
18.67
62
17
2.14
20.57
63.43
13.86
0
1
2
3
0
1
2
3
JOB
JOB
0
1
2
3


Distribution of Categories based on RESPONSE(0 and 1)
0
1
58.43
41.57
62.33
37.67
count
400
300
200
100
0
0
1
0
1
TELEPHONE
TELEPHONE
0
1




```
# Finding the proportion of Good and Bad customer in each categorical variable

for (i in c[2:25]){
  writeLines("\n\n")
  print("------------------------------------------------------------------------------------------")
  print(paste("Input Variable :",i))
  cat <- unique(df[i])
  print(paste("Categories in",i,":"))
  for (j in cat){
    print(j)

    for (k in j){
      dfx <- filter(df, df[i]==k)
      t_cnt <- count(dfx)
      df_1 <- filter(df1, df1[i]==k)
      c1 <- count(df_1)
      df_0 <- filter(df0, df0[i]==k)
      c0 <- count(df_0)
      writeLines("\n")
      print(paste("Percentage(%) of RESPONSE = 1 for category",k,"in variable",i,"is: ",round(c1/t_cnt*
      print(paste("Percentage(%) of RESPONSE = 0 for category",k,"in variable",i,"is: ",round(c0/t_cnt*
    }
  }
}
```

```
##
```

```
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : CHK_ACCT"
## [1] "Categories in CHK_ACCT :"
## [1] 0 1 3 2
## Levels: 0 1 2 3
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable CHK_ACCT is:  50.73 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable CHK_ACCT is:  49.27 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable CHK_ACCT is:  60.97 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable CHK_ACCT is:  39.03 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable CHK_ACCT is:  88.32 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable CHK_ACCT is:  11.68 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable CHK_ACCT is:  77.78 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable CHK_ACCT is:  22.22 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : HISTORY"
## [1] "Categories in HISTORY :"
## [1] 4 2 3 0 1
## Levels: 0 1 2 3 4
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 4 in variable HISTORY is:  82.94 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 4 in variable HISTORY is:  17.06 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable HISTORY is:  68.11 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable HISTORY is:  31.89 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable HISTORY is:  68.18 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable HISTORY is:  31.82 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable HISTORY is:  37.5 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable HISTORY is:  62.5 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable HISTORY is:  42.86 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable HISTORY is:  57.14 %"
##
##
##
```

```
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : NEW_CAR"
## [1] "Categories in NEW_CAR :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable NEW_CAR is:  72.45 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable NEW_CAR is:  27.55 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable NEW_CAR is:  61.97 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable NEW_CAR is:  38.03 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : USED_CAR"
## [1] "Categories in USED_CAR :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable USED_CAR is:  68.45 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable USED_CAR is:  31.55 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable USED_CAR is:  83.5 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable USED_CAR is:  16.5 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : FURNITURE"
## [1] "Categories in FURNITURE :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable FURNITURE is:  70.45 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable FURNITURE is:  29.55 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable FURNITURE is:  67.96 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable FURNITURE is:  32.04 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : RADIO_TV"
## [1] "Categories in RADIO_TV :"
## [1] 1 0
## Levels: 0 1
##
```

```
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable RADIO_TV is:  77.86 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable RADIO_TV is:  22.14 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable RADIO_TV is:  66.94 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable RADIO_TV is:  33.06 %"
##
##
##
## [1] "----------------------------------------------------------------------------------------"
## [1] "Input Variable : EDUCATION"
## [1] "Categories in EDUCATION :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable EDUCATION is:  70.74 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable EDUCATION is:  29.26 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable EDUCATION is:  56 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable EDUCATION is:  44 %"
##
##
##
## [1] "----------------------------------------------------------------------------------------"
## [1] "Input Variable : RETRAINING"
## [1] "Categories in RETRAINING :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable RETRAINING is:  70.54 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable RETRAINING is:  29.46 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable RETRAINING is:  64.95 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable RETRAINING is:  35.05 %"
##
##
##
## [1] "----------------------------------------------------------------------------------------"
## [1] "Input Variable : SAV_ACCT"
## [1] "Categories in SAV_ACCT :"
## [1] 4 0 2 3 1
## Levels: 0 1 2 3 4
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 4 in variable SAV_ACCT is:  82.51 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 4 in variable SAV_ACCT is:  17.49 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable SAV_ACCT is:  64.01 %"
```

```
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable SAV_ACCT is:  35.99 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable SAV_ACCT is:  82.54 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable SAV_ACCT is:  17.46 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable SAV_ACCT is:  87.5 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable SAV_ACCT is:  12.5 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable SAV_ACCT is:  66.99 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable SAV_ACCT is:  33.01 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : EMPLOYMENT"
## [1] "Categories in EMPLOYMENT :"
## [1] 4 2 3 0 1
## Levels: 0 1 2 3 4
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 4 in variable EMPLOYMENT is:  74.7 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 4 in variable EMPLOYMENT is:  25.3 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable EMPLOYMENT is:  69.32 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable EMPLOYMENT is:  30.68 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable EMPLOYMENT is:  77.59 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable EMPLOYMENT is:  22.41 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable EMPLOYMENT is:  62.9 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable EMPLOYMENT is:  37.1 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable EMPLOYMENT is:  59.3 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable EMPLOYMENT is:  40.7 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : MALE_DIV"
## [1] "Categories in MALE_DIV :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable MALE_DIV is:  70.53 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable MALE_DIV is:  29.47 %"
##
```

```
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable MALE_DIV is:  60 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable MALE_DIV is:  40 %"
##
##
##
## [1] "------------------------------------------------------------------------------------------"
## [1] "Input Variable : MALE_SINGLE"
## [1] "Categories in MALE_SINGLE :"
## [1] 1 0
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable MALE_SINGLE is:  73.36 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable MALE_SINGLE is:  26.64 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable MALE_SINGLE is:  65.93 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable MALE_SINGLE is:  34.07 %"
##
##
##
## [1] "------------------------------------------------------------------------------------------"
## [1] "Input Variable : MALE_MAR_or_WID"
## [1] "Categories in MALE_MAR_or_WID :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable MALE_MAR_or_WID is:  69.71 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable MALE_MAR_or_WID is:  30.29 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable MALE_MAR_or_WID is:  72.83 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable MALE_MAR_or_WID is:  27.17 %"
##
##
##
## [1] "------------------------------------------------------------------------------------------"
## [1] "Input Variable : CO_APPLICANT"
## [1] "Categories in CO_APPLICANT :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable CO_APPLICANT is:  70.59 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable CO_APPLICANT is:  29.41 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable CO_APPLICANT is:  56.1 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable CO_APPLICANT is:  43.9 %"
##
##
##
```

```
## [1] "-------------------------------------------------------------------------------------------"
## [1] "Input Variable : GUARANTOR"
## [1] "Categories in GUARANTOR :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable GUARANTOR is:  69.41 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable GUARANTOR is:  30.59 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable GUARANTOR is:  80.77 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable GUARANTOR is:  19.23 %"
##
##
##
## [1] "-------------------------------------------------------------------------------------------"
## [1] "Input Variable : PRESENT_RESIDENT"
## [1] "Categories in PRESENT_RESIDENT :"
## [1] 4 2 3 1
## Levels: 1 2 3 4
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 4 in variable PRESENT_RESIDENT is:  69.98 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 4 in variable PRESENT_RESIDENT is:  30.02 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable PRESENT_RESIDENT is:  68.51 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable PRESENT_RESIDENT is:  31.49 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable PRESENT_RESIDENT is:  71.14 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable PRESENT_RESIDENT is:  28.86 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable PRESENT_RESIDENT is:  72.31 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable PRESENT_RESIDENT is:  27.69 %"
##
##
##
## [1] "-------------------------------------------------------------------------------------------"
## [1] "Input Variable : REAL_ESTATE"
## [1] "Categories in REAL_ESTATE :"
## [1] 1 0
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable REAL_ESTATE is:  78.72 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable REAL_ESTATE is:  21.28 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable REAL_ESTATE is:  66.57 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable REAL_ESTATE is:  33.43 %"
##
```

```
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : PROP_UNKN_NONE"
## [1] "Categories in PROP_UNKN_NONE :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable PROP_UNKN_NONE is:  72.46 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable PROP_UNKN_NONE is:  27.54 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable PROP_UNKN_NONE is:  56.49 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable PROP_UNKN_NONE is:  43.51 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : OTHER_INSTALL"
## [1] "Categories in OTHER_INSTALL :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable OTHER_INSTALL is:  72.48 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable OTHER_INSTALL is:  27.52 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable OTHER_INSTALL is:  59.14 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable OTHER_INSTALL is:  40.86 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : RENT"
## [1] "Categories in RENT :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable RENT is:  71.99 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable RENT is:  28.01 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable RENT is:  60.89 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable RENT is:  39.11 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : OWN_RES"
## [1] "Categories in OWN_RES :"
## [1] 1 0
```

```
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable OWN_RES is:  73.91 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable OWN_RES is:  26.09 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable OWN_RES is:  60.28 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable OWN_RES is:  39.72 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : JOB"
## [1] "Categories in JOB :"
## [1] 2 1 3 0
## Levels: 0 1 2 3
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 2 in variable JOB is:  70.48 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 2 in variable JOB is:  29.52 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable JOB is:  72 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable JOB is:  28 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 3 in variable JOB is:  65.54 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 3 in variable JOB is:  34.46 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable JOB is:  68.18 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable JOB is:  31.82 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : TELEPHONE"
## [1] "Categories in TELEPHONE :"
## [1] 1 0
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable TELEPHONE is:  72.03 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable TELEPHONE is:  27.97 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable TELEPHONE is:  68.62 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable TELEPHONE is:  31.38 %"
##
##
##
## [1] "-----------------------------------------------------------------------------------------"
## [1] "Input Variable : FOREIGN"
```

```
## [1] "Categories in FOREIGN :"
## [1] 0 1
## Levels: 0 1
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 0 in variable FOREIGN is:  69.26 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 0 in variable FOREIGN is:  30.74 %"
##
##
## [1] "Percentage(%) of RESPONSE = 1 for category 1 in variable FOREIGN is:  89.19 %"
## [1] "Percentage(%) of RESPONSE = 0 for category 1 in variable FOREIGN is:  10.81 %"
```

from the above plots and distribution we can conclude that the following variables have a significant impact on the Target variable: CHK_ACCT NEW_CAR OWN_RES TELEPHONE RETRAINING PROP_UNKN_NONE MALE_SINGLE

#NUMERICAL VARIABLE

```r
# saving all numerical variable column names in "nv"
nv <- col[c(3,11,14,23,27,29)]
print("Numerical variables in the dataset :")
```

```
## [1] "Numerical variables in the dataset :"
```

```r
nv
```

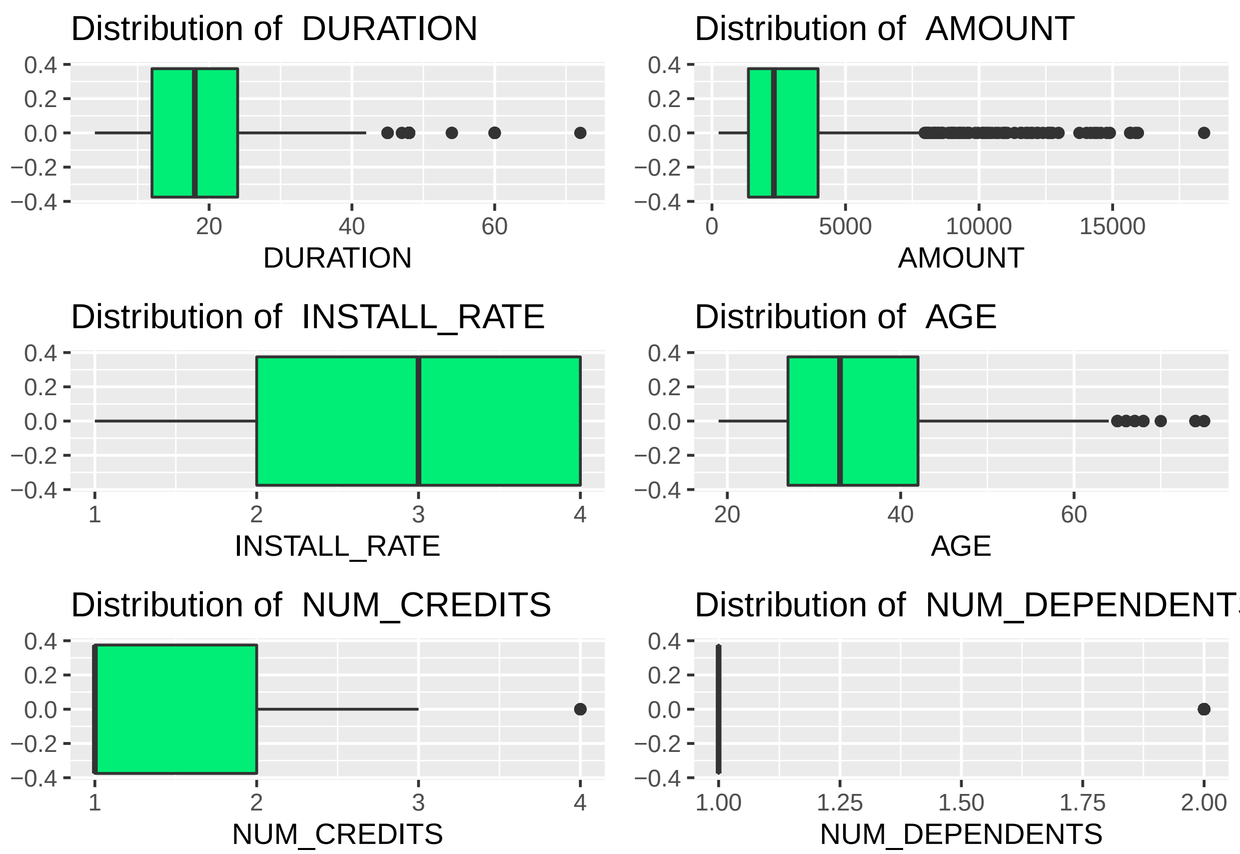
```
## [1] "DURATION"       "AMOUNT"         "INSTALL_RATE"   "AGE"
## [5] "NUM_CREDITS"    "NUM_DEPENDENTS"
```

```r
# Distribution of Numerical variables
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

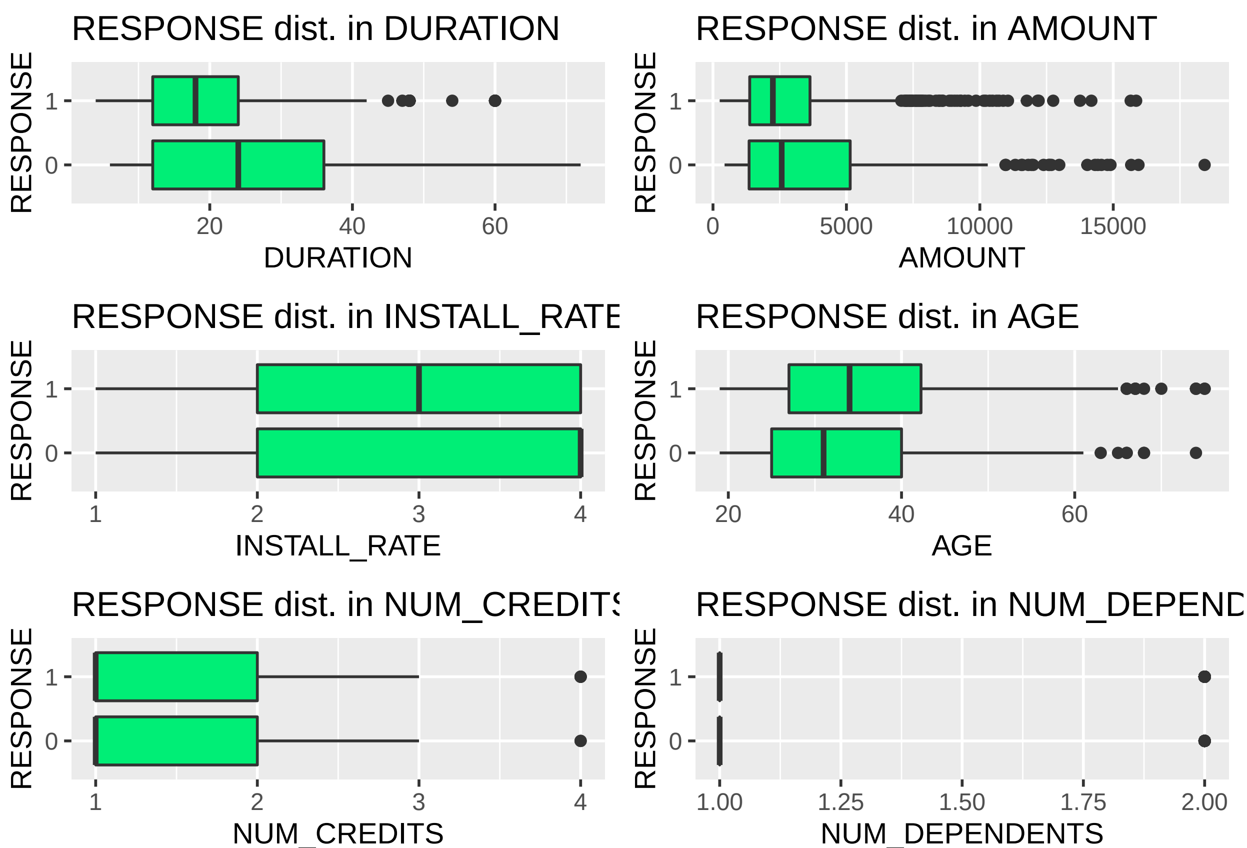
```r
plot_list <- list()
n=1
for (i in nv){
  plot_list[[n]] <- ggplot(df,aes_string(x=i))+geom_boxplot(fill='springgreen2')+
    ggtitle(paste("Distribution of ",i))
  n=n+1
}
grid.arrange(grobs=plot_list,ncol=2)
```



NUMERICAL VARIABLES VS TARGET

```
# Plots to depict the distribution of Numberical variables for Good and
# Bad Customers

plot_list <- list()
n=1
for (i in nv){
    plot_list[[n]] <- ggplot(df,aes_string(x=i, y="RESPONSE", group="RESPONSE"))+
    geom_boxplot(fill="springgreen2")+
    ggtitle(paste("RESPONSE dist. in",i))
  n=n+1
}
grid.arrange(grobs=plot_list,ncol=2)
```



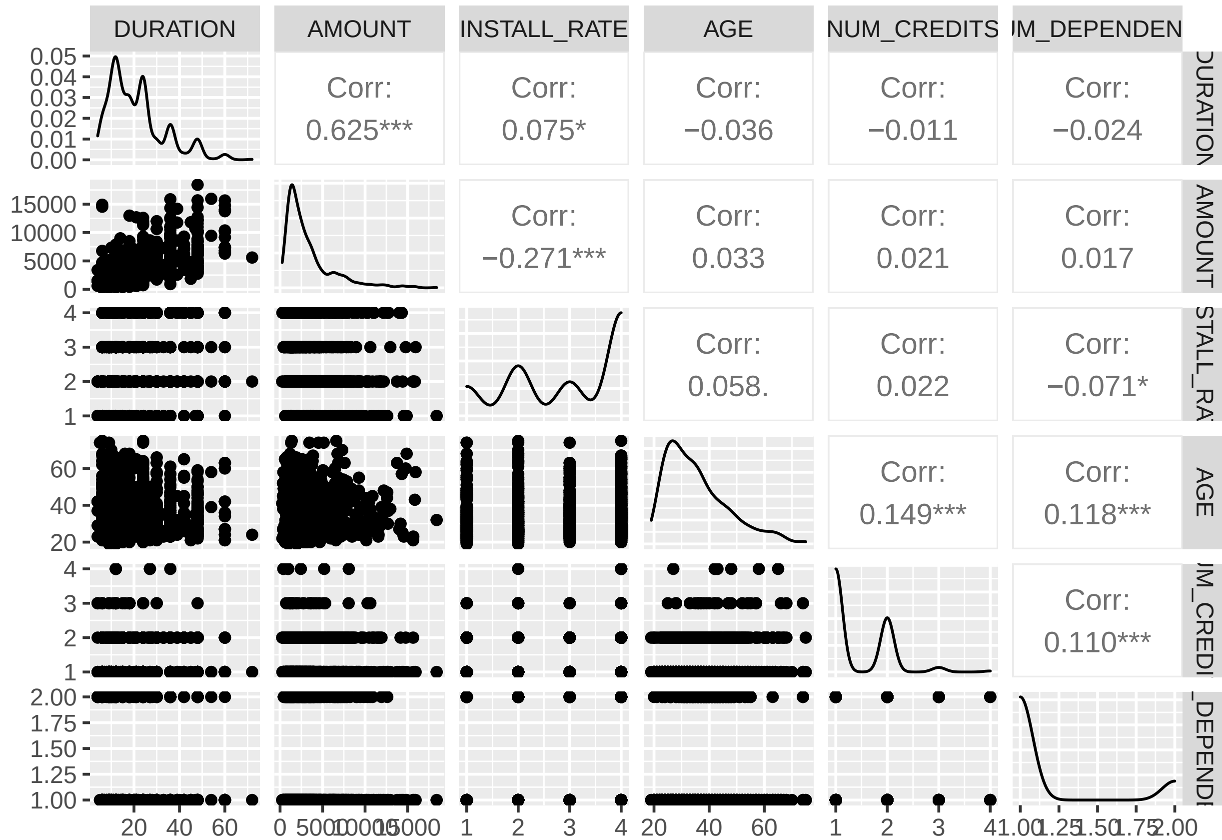
We can see that out of all 6 numerical variables only Duration, Amount, Age are impacting the Target variable.

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

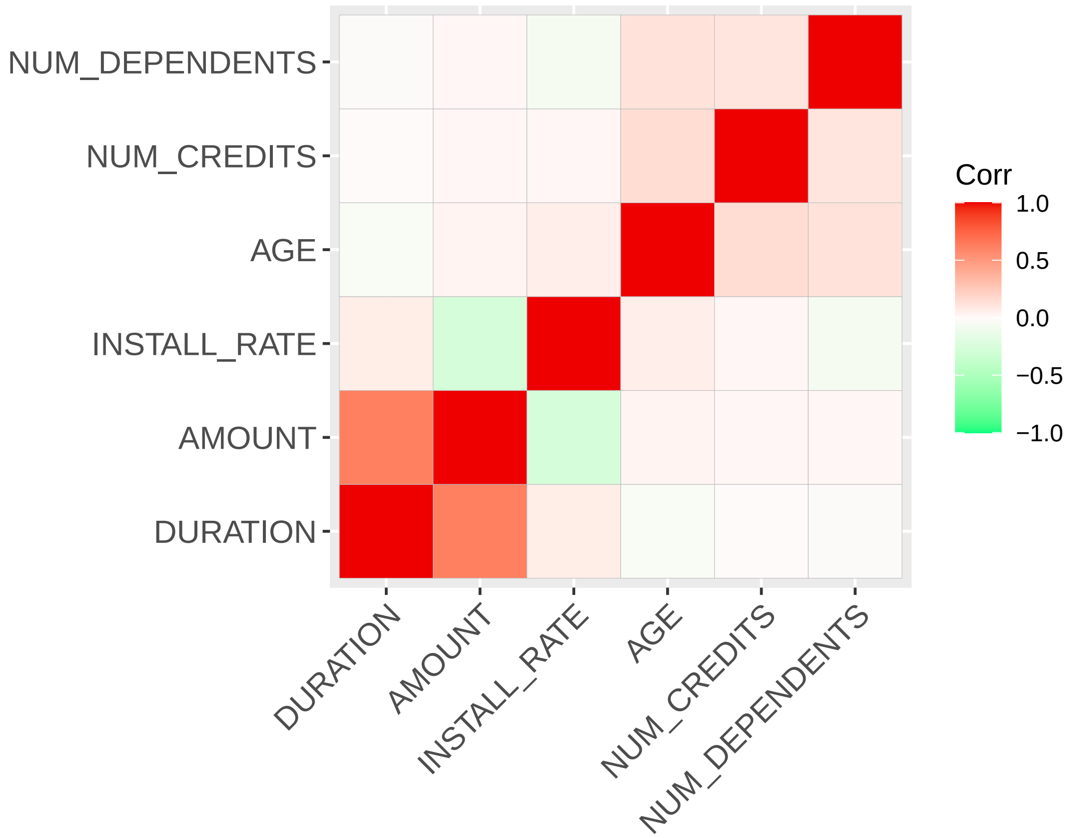
CORELATION BETWEEN INUPT VARIABLES (checking MULTICOLINEARITY)

```
# pair plot for input varibales
ggpairs(df[nv])
```




```
library(ggcorrplot)
```

```
ggcorrplot(cor(df[nv]), ggtheme = 'theme_dark', show.legend = TRUE, colors=c('springgreen1','snow1','re
```



From the above pairplot and heatmap we can observe that except for Duration and Amount none of the other input variables have any correlation with each other. Even for Duration and amount the correlation is only around 0.5 hence there is not a significant multicollinearity in the dataset.

Question (b)

DECISION TREE BUILDING

```
df <- df[-1] # Removing observation column (since all unique-no effect on target)
set.seed(5)
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.8, 0.2))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]
```

```
d_train <- list(dim(train))
d_test <- list(dim(test))

print(paste("Dimension of train data:",d_train))
```

```
## [1] "Dimension of train data: c(785, 31)"
```

```r
print(paste("Dimension of test data:",d_test))
```

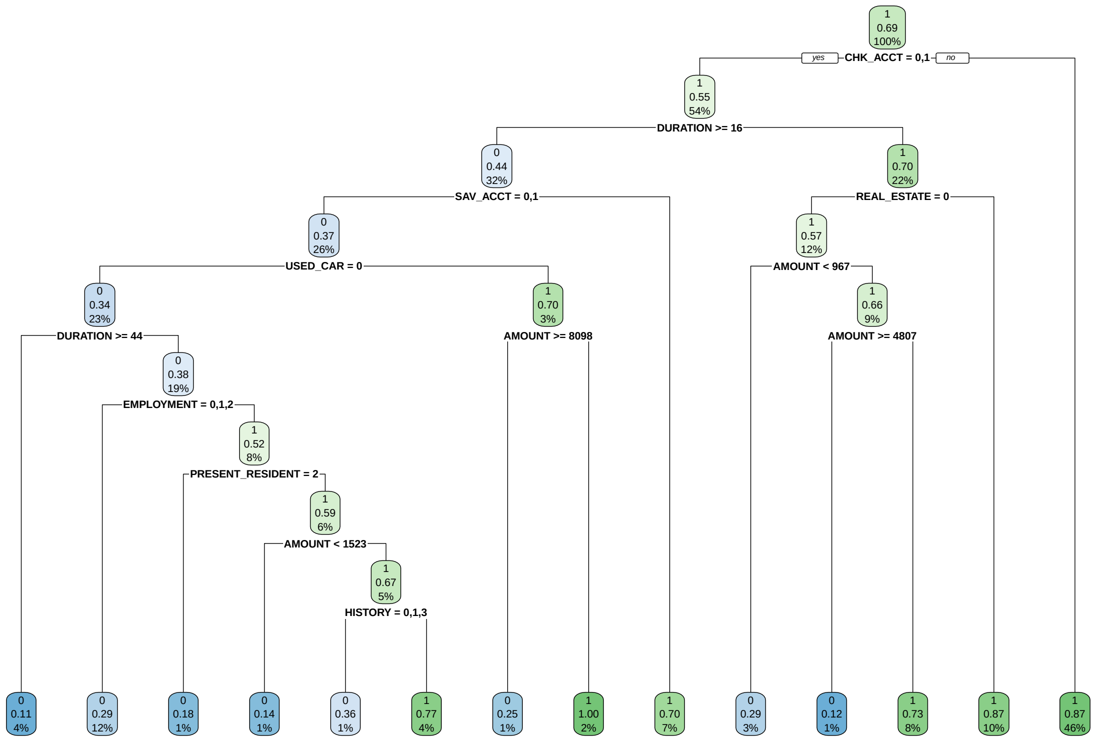
```
## [1] "Dimension of test data: c(215, 31)"
```

```r
library(rpart)
tree_model <- rpart(RESPONSE ~ ., train)
```

```r
print(tree_model)
```

```
## n= 785
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 785 240 1 (0.3057325 0.6942675)
##     2) CHK_ACCT=0,1 425 193 1 (0.4541176 0.5458824)
##       4) DURATION>=15.5 255 113 0 (0.5568627 0.4431373)
##         8) SAV_ACCT=0,1 201  75 0 (0.6268657 0.3731343)
##          16) USED_CAR=0 181  61 0 (0.6629834 0.3370166)
##            32) DURATION>=43.5 28   3 0 (0.8928571 0.1071429) *
##            33) DURATION< 43.5 153  58 0 (0.6209150 0.3790850)
##              66) EMPLOYMENT=0,1,2 93  27 0 (0.7096774 0.2903226) *
##              67) EMPLOYMENT=3,4 60  29 1 (0.4833333 0.5166667)
##               134) PRESENT_RESIDENT=2 11   2 0 (0.8181818 0.1818182) *
##               135) PRESENT_RESIDENT=1,3,4 49  20 1 (0.4081633 0.5918367)
##                 270) AMOUNT< 1522.5 7   1 0 (0.8571429 0.1428571) *
##                 271) AMOUNT>=1522.5 42  14 1 (0.3333333 0.6666667)
##                   542) HISTORY=0,1,3 11   4 0 (0.6363636 0.3636364) *
##                   543) HISTORY=2,4 31   7 1 (0.2258065 0.7741935) *
##          17) USED_CAR=1 20   6 1 (0.3000000 0.7000000)
##            34) AMOUNT>=8097.5 8   2 0 (0.7500000 0.2500000) *
##            35) AMOUNT< 8097.5 12   0 1 (0.0000000 1.0000000) *
##         9) SAV_ACCT=2,3,4 54  16 1 (0.2962963 0.7037037) *
##       5) DURATION< 15.5 170  51 1 (0.3000000 0.7000000)
##        10) REAL_ESTATE=0 95  41 1 (0.4315789 0.5684211)
##          20) AMOUNT< 967 24   7 0 (0.7083333 0.2916667) *
##          21) AMOUNT>=967 71  24 1 (0.3380282 0.6619718)
##            42) AMOUNT>=4807 8   1 0 (0.8750000 0.1250000) *
##            43) AMOUNT< 4807 63  17 1 (0.2698413 0.7301587) *
##        11) REAL_ESTATE=1 75  10 1 (0.1333333 0.8666667) *
##     3) CHK_ACCT=2,3 360  47 1 (0.1305556 0.8694444) *
```

```r
library(rpart.plot)
rpart.plot(tree_model)
```




```r
pred_train_prob <- predict(tree_model, train, type = "prob")
pred_train <- predict(tree_model, train, type = "class")
```

```r
pred_test <- predict(tree_model, test, type = "class")
testerror <- mean(pred_test != test$RESPONSE)
print(testerror)
```

```
## [1] 0.2883721
```

```r
# function to evaluate model performance on Bad credit (test) :
metrics_0 <- function(cm_test){
  print(paste("Test accuracy :", sum(diag(cm_test)) / sum(cm_test)))
  rc0 <- cm_test[1,1]/(cm_test[1,1]+cm_test[1,2])
  pr0 <- cm_test[1,1]/(cm_test[1,1]+cm_test[2,1])
  f0 <- 2*(pr0*rc0/(pr0+rc0))
  print(paste("Recall of 0 :", rc0))
  print(paste("f score of 0 :", f0))

}
```

#C5.0 Tree (parms split = 'infromation')

Pre-Pruning

```r
bucket <- c(10,20,30)
split <- c(50,75,100)

for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                         control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.716279069767442"
## [1] "Recall of 0 : 0.416666666666667"
## [1] "f score of 0 : 0.45045045045045"
##
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.730232558139535"
## [1] "Recall of 0 : 0.55"
## [1] "f score of 0 : 0.532258064516129"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.720930232558139"
## [1] "Recall of 0 : 0.35"
## [1] "f score of 0 : 0.411764705882353"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.720930232558139"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.473684210526316"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.753488372093023"
## [1] "Recall of 0 : 0.516666666666667"
## [1] "f score of 0 : 0.539130434782609"
##
##
```

```
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.366666666666667"
## [1] "f score of 0 : 0.403669724770642"
##
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.67906976744186"
## [1] "Recall of 0 : 0.4"
## [1] "f score of 0 : 0.41025641025641"
```

#C&R Tree (parms split = 'gini')

```r
bucket <- c(10,20,30)
split <- c(50,75,100)

for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "gini"),
                         control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.702325581395349"
## [1] "Recall of 0 : 0.333333333333333"
## [1] "f score of 0 : 0.384615384615385"
##
##
##
## [1] "For bucket = 10 and split = 75"
```

```
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.333333333333333"
## [1] "f score of 0 : 0.388349514563107"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.25"
## [1] "f score of 0 : 0.32258064516129"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.693023255813954"
## [1] "Recall of 0 : 0.483333333333333"
## [1] "f score of 0 : 0.467741935483871"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.333333333333333"
## [1] "f score of 0 : 0.388349514563107"
##
##
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.25"
## [1] "f score of 0 : 0.32258064516129"
##
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.67906976744186"
## [1] "Recall of 0 : 0.4"
## [1] "f score of 0 : 0.41025641025641"
```

We can observe that C5.0 performs better than C&R for our given parameters, For C5.0, the best values for minbucket and minsplit seem to be: - minbucket = 10 & minsplit =75 - minbucket = 20 & minsplit = 75

So, let's first determine the best bucket size for split =75:

```
# determining best bucket size for split = 75:

bucket <- c(5,10,15,20,25,30)
split <- 75

for (i in bucket){
  print(paste("For bucket =",i,"and split =",split))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
               control = rpart.control(minbucket = i, minsplit = split, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 5 and split = 75"
## [1] "Test accuracy : 0.720930232558139"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.473684210526316"
##
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.730232558139535"
## [1] "Recall of 0 : 0.55"
## [1] "f score of 0 : 0.532258064516129"
##
##
##
## [1] "For bucket = 15 and split = 75"
## [1] "Test accuracy : 0.753488372093023"
## [1] "Recall of 0 : 0.516666666666667"
## [1] "f score of 0 : 0.539130434782609"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.753488372093023"
## [1] "Recall of 0 : 0.516666666666667"
## [1] "f score of 0 : 0.539130434782609"
##
##
##
## [1] "For bucket = 25 and split = 75"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.583333333333333"
## [1] "f score of 0 : 0.526315789473684"
##
##
##
```

```
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
```

Bucket = 25 seems like the optimal condition with regards to the business problem Even though, bucket = 10 gives slightly better accuracy and f score, bucket = 25 gives a higher recall of Bad credit which outweighs the other criteria.

```
# determinig best split for bucket = 25:

bucket <- 25
split <- c(65,70,75,80)

for (i in split){
  print(paste("For bucket =",bucket,"and split =",i))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
              control = rpart.control(minbucket = bucket, minsplit = i, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 25 and split = 65"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.45"
## [1] "f score of 0 : 0.453781512605042"
##
##
##
## [1] "For bucket = 25 and split = 70"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.583333333333333"
## [1] "f score of 0 : 0.526315789473684"
##
##
##
## [1] "For bucket = 25 and split = 75"
## [1] "Test accuracy : 0.706976744186047"
## [1] "Recall of 0 : 0.583333333333333"
## [1] "f score of 0 : 0.526315789473684"
##
##
##
## [1] "For bucket = 25 and split = 80"
## [1] "Test accuracy : 0.734883720930233"
## [1] "Recall of 0 : 0.516666666666667"
## [1] "f score of 0 : 0.521008403361345"
```

Both 70 & 75 gives us same results, so let's take split = 70, since it's a multiple of 10.

Hence, best values for control : minbucket = 25 minsplit = 70

Post - Pruning

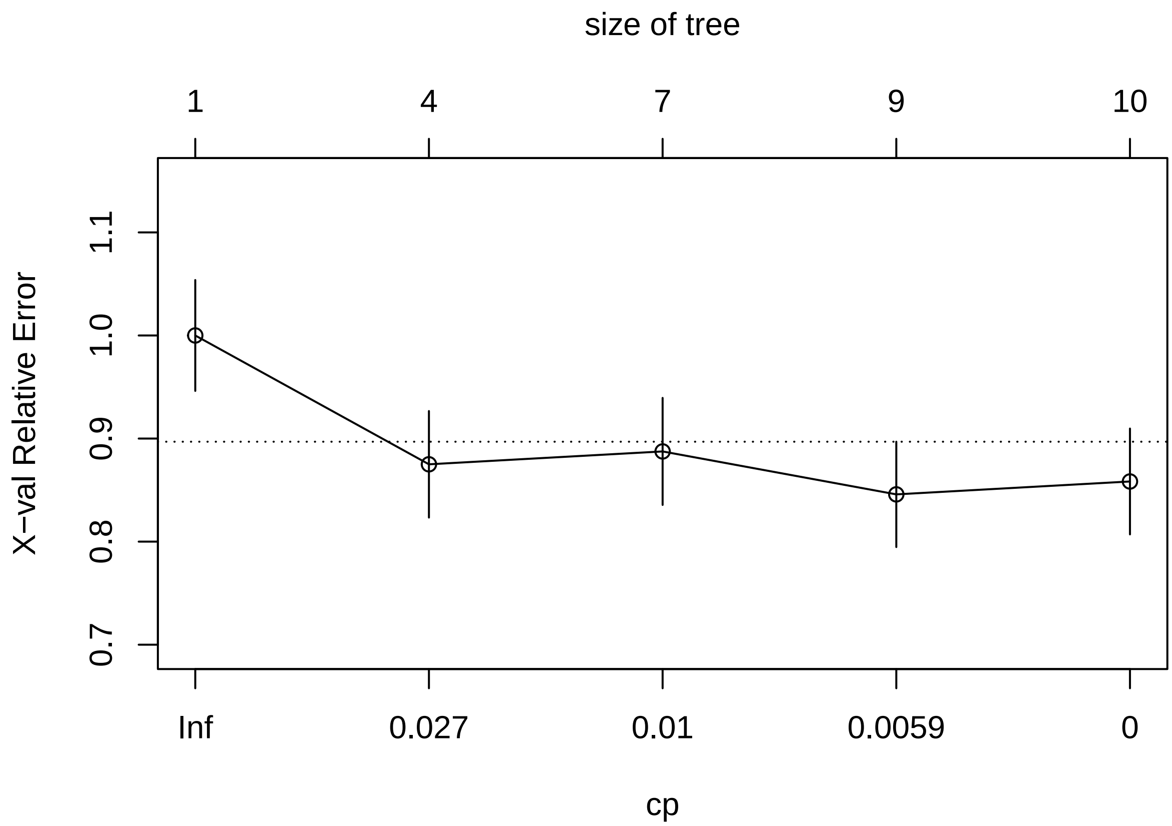
```r
# Determining best cp value for best minbucket and minsplit

set.seed(50)
tree_model_tune <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                 control = rpart.control(minbucket = 25, minsplit = 70, cp=0))

printcp(tree_model_tune)
```

```
##
## Classification tree:
## rpart(formula = RESPONSE ~ ., data = train, parms = list(split = "information"),
##     control = rpart.control(minbucket = 25, minsplit = 70, cp = 0))
##
## Variables actually used in tree construction:
## [1] AMOUNT       CHK_ACCT     DURATION     EMPLOYMENT   HISTORY
## [6] INSTALL_RATE REAL_ESTATE  SAV_ACCT
##
## Root node error: 240/785 = 0.30573
##
## n= 785
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0604167      0   1.00000 1.00000 0.053785
## 2 0.0125000      3   0.78750 0.87500 0.051677
## 3 0.0083333      6   0.75000 0.88750 0.051909
## 4 0.0041667      8   0.73333 0.84583 0.051117
## 5 0.0000000      9   0.72917 0.85833 0.051360
```

```r
plotcp(tree_model_tune)
```



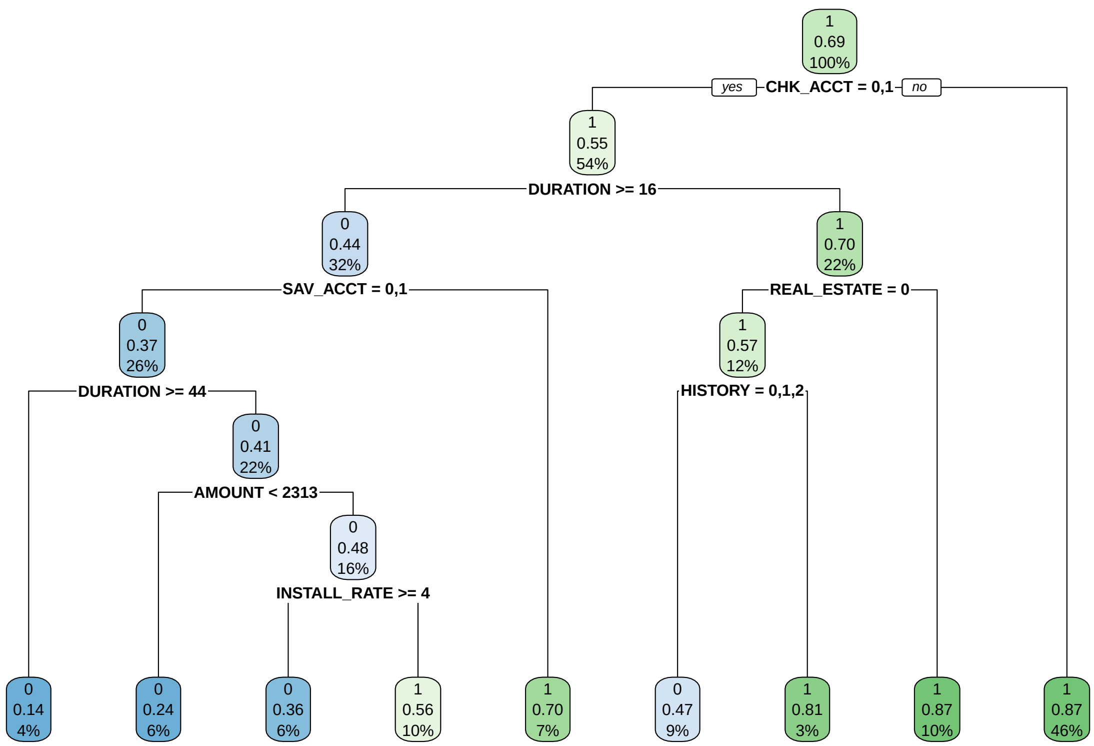

```
cp <- 0.00416
prunedTree <- prune(tree_model2, cp = cp)
print(prunedTree)
```

```
## n= 785
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 785 240 1 (0.3057325 0.6942675)
##    2) CHK_ACCT=0,1 425 193 1 (0.4541176 0.5458824)
##      4) DURATION>=15.5 255 113 0 (0.5568627 0.4431373)
##        8) SAV_ACCT=0,1 201  75 0 (0.6268657 0.3731343)
##         16) DURATION>=43.5 29   4 0 (0.8620690 0.1379310) *
##         17) DURATION< 43.5 172  71 0 (0.5872093 0.4127907)
##           34) AMOUNT< 2313 50  12 0 (0.7600000 0.2400000) *
##           35) AMOUNT>=2313 122  59 0 (0.5163934 0.4836066)
##             70) INSTALL_RATE>=3.5 45  16 0 (0.6444444 0.3555556) *
##             71) INSTALL_RATE< 3.5 77  34 1 (0.4415584 0.5584416) *
##        9) SAV_ACCT=2,3,4 54  16 1 (0.2962963 0.7037037) *
##      5) DURATION< 15.5 170  51 1 (0.3000000 0.7000000)
##       10) REAL_ESTATE=0 95  41 1 (0.4315789 0.5684211)
##         20) HISTORY=0,1,2 68  32 0 (0.5294118 0.4705882) *
##         21) HISTORY=3,4 27   5 1 (0.1851852 0.8148148) *
##       11) REAL_ESTATE=1 75  10 1 (0.1333333 0.8666667) *
##    3) CHK_ACCT=2,3 360  47 1 (0.1305556 0.8694444) *
```

```r
# Checking classification nmetrics after post pruning
pred_test_prune <- predict(prunedTree, test, type = "class")
cm_test_prune <- table(test$RESPONSE, pred_test_prune)
metrics_0(cm_test_prune)
```

```
## [1] "Test accuracy : 0.734883720930233"
## [1] "Recall of 0 : 0.516666666666667"
## [1] "f score of 0 : 0.521008403361345"
```

```r
rpart.plot(prunedTree)
```




```r
# In this function for both Train and Test data we're calculating the
# classification metrics based on their confusion matrix

# We're finding the Precision, Recall and F score for both 1 and 0
# classifications

metrics <- function(cm_train,cm_test){
  print("                    FOR TRANING DATA                    ")
  writeLines("\n")
  print("Confusion Matrix (Train) :")
  print(cm_train)
  print(paste("Train accuracy :", sum(diag(cm_train)) / sum(cm_train)))
```

```
  pr1 <- cm_train[2,2]/(cm_train[2,2]+cm_train[1,2])  # for predicted 1
  rc1 <- cm_train[2,2]/(cm_train[2,2]+cm_train[2,1])
  f1 <- 2*(pr1*rc1/(pr1+rc1))

  pr0 <- cm_train[1,1]/(cm_train[1,1]+cm_train[2,1])  # for predicted 0
  rc0 <- cm_train[1,1]/(cm_train[1,1]+cm_train[1,2])
  f0 <- 2*(pr0*rc0/(pr0+rc0))


  writeLines("\n")
  print("Classification Report (traning data) :")
  target_variable <- c( 0,1)
  precision <- c(pr0, pr1)
  recall <- c(rc0, rc1)
  f_score <- c(f0,f1)
  cf_r <- data.frame(target_variable,precision,recall,f_score)
  rownames(cf_r) <- 0:1
  print(cf_r)

  writeLines("\n")
  print("                       FOR TESTING DATA                      ")
  writeLines("\n")
  print("Confusion Matrix (Test) :")
  print(cm_test)
  print(paste("Test accuracy :", sum(diag(cm_test)) / sum(cm_test)))
  writeLines("\n")
  pr1 <- cm_test[2,2]/(cm_test[2,2]+cm_test[1,2])
  rc1 <- cm_test[2,2]/(cm_test[2,2]+cm_test[2,1])
  f1 <- 2*(pr1*rc1/(pr1+rc1))

  pr0 <- cm_test[1,1]/(cm_test[1,1]+cm_test[2,1])
  rc0 <- cm_test[1,1]/(cm_test[1,1]+cm_test[1,2])
  f0 <- 2*(pr0*rc0/(pr0+rc0))

  print("Classification Report (testing data) :")
  target_variable <- c( 0,1)
  precision <- c(pr0, pr1)
  recall <- c(rc0, rc1)
  f_score <- c(f0,f1)
  cf_r <- data.frame(target_variable,precision,recall,f_score)
  rownames(cf_r) <- 0:1
  print(cf_r)  # cf_r is a dataframe containing the calculated metrics

}
```

#For 80:20 Train test split: building C5.0 and C&R Tree to do complete evaluation:

```
# Tree building

tree_82_C5 <-rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                control = rpart.control(minbucket = 25, minsplit = 70, cp=0.00416))

tree_82_CR <-rpart(RESPONSE ~ ., train, parms = list(split = "gini"),
```

```r
                 control = rpart.control(minbucket = 25, minsplit = 70, cp=0.00416))

# Predicting train and test RESPONSE:

pred_train_C5 <- predict(tree_82_C5, train, type = "class")
pred_test_C5 <- predict(tree_82_C5, test, type = "class")

pred_train_CR <- predict(tree_82_CR, train, type = "class")
pred_test_CR <- predict(tree_82_CR, test, type = "class")


# Genetrating Confusion matrix and classification report:

cm_train_C5 <- table(train$RESPONSE, pred_train_C5)
cm_test_C5 <- table(test$RESPONSE, pred_test_C5)

cm_train_CR <- table(train$RESPONSE, pred_train_CR)
cm_test_CR <- table(test$RESPONSE, pred_test_CR)

print("------------------------ FOR C5.0 DT ------------------------------")
```

```
## [1] "------------------------ FOR C5.0 DT ------------------------------"
```

```r
metrics(cm_train_C5, cm_test_C5)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_C5
##       0   1
##   0 147  93
##   1  82 463
## [1] "Train accuracy : 0.777070063694268"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.6419214 0.6125000 0.6268657
## 1               1 0.8327338 0.8495413 0.8410536
##
##
## [1] "                       FOR TESTING DATA                       "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5
##       0   1
##   0  35  25
##   1  38 117
## [1] "Test accuracy : 0.706976744186047"
##
```

```
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0 0.4794521 0.5833333 0.5263158
## 1               1 0.8239437 0.7548387 0.7878788
```

```
print("------------------------ FOR C&R DT ------------------------------")
```

```
## [1] "------------------------ FOR C&R DT ------------------------------"
```

```
metrics(cm_train_CR, cm_test_CR)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_CR
##       0   1
##   0 131 109
##   1  69 476
## [1] "Train accuracy : 0.773248407643312"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.6550000 0.5458333 0.5954545
## 1               1 0.8136752 0.8733945 0.8424779
##
##
## [1] "                       FOR TESTING DATA                       "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_CR
##       0   1
##   0  31  29
##   1  33 122
## [1] "Test accuracy : 0.711627906976744"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0  0.484375 0.5166667 0.5000000
## 1               1  0.807947 0.7870968 0.7973856
```

QUESTION (c)

#USING WEIGHTED LOSS CRITERIA FOR FALSE POSITIVE & FALE NEGATIVE

```r
loss_m <- matrix(c(0, 5, 1, 0), byrow=TRUE, ncol=2)
print("Loss matrix :")  # Weight of FP is 5 times that of FN.
```

```
## [1] "Loss matrix :"
```

```r
loss_m
```

```
##      [,1] [,2]
## [1,]    0    5
## [2,]    1    0
```

```r
tree_82_C5_loss <-rpart(RESPONSE ~ ., train, parms = list(split = "information", loss=loss_m),
                     control = rpart.control(minbucket = 25, minsplit = 70, cp=0.00416))

pred_train_C5_loss <- predict(tree_82_C5_loss, train, type = "class")
pred_test_C5_loss <- predict(tree_82_C5_loss, test, type = "class")

cm_train_C5_loss <- table(train$RESPONSE, pred_train_C5_loss)
cm_test_C5_loss <- table(test$RESPONSE, pred_test_C5_loss)

metrics(cm_train_C5_loss, cm_test_C5_loss)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_C5_loss
##       0   1
##   0 229  11
##   1 300 245
## [1] "Train accuracy : 0.603821656050955"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.4328922 0.9541667 0.5955787
## 1               1 0.9570312 0.4495413 0.6117353
##
##
## [1] "                       FOR TESTING DATA                       "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5_loss
##      0  1
##   0 54  6
##   1 97 58
## [1] "Test accuracy : 0.52093023255814"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
```

```
## 0               0 0.3576159 0.9000000 0.5118483
## 1               1 0.9062500 0.3741935 0.5296804
```

#Inference after applying weighted loss to FP & FN:

We can observe that the False positive cases in test data has come down from 29 to just 6 out of 60 observations. Hence the RECALL of category 0 (BAD) has increased tremendously to a value of 90%. But we have to keep in mind that there's a tradeoff for this, as we can see that our FN cases has increased from 33 to 97, so it has amost tripled. We need to check with our business clients if this tradeoff is worth it or not.

Question (b) continued..

70:30 SPLIT :

```
set.seed(5)
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.7, 0.3))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]
```

```
bucket <- c(10,20,30)
split <- c(50,75,100)

print("-------------------------  For C5.0 Tree  -------------------------------")
```

```
## [1] "-------------------------  For C5.0 Tree  -------------------------------"
```

```
for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                         control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.717607973421927"
## [1] "Recall of 0 : 0.426966292134831"
## [1] "f score of 0 : 0.472049689440994"
##
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.724252491694352"
## [1] "Recall of 0 : 0.393258426966292"
```

```
## [1] "f score of 0 : 0.457516339869281"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.481927710843373"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.700996677740864"
## [1] "Recall of 0 : 0.415730337078652"
## [1] "f score of 0 : 0.451219512195122"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.710963455149502"
## [1] "Recall of 0 : 0.348314606741573"
## [1] "f score of 0 : 0.416107382550336"
##
##
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
##
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.370786516853933"
## [1] "f score of 0 : 0.420382165605096"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
```

```r
print("------------------------  For C&R Tree ----------------------------------")
```

```
## [1] "------------------------  For C&R Tree ----------------------------------"
```

```r
for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "gini"),
                         control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.438202247191011"
## [1] "f score of 0 : 0.461538461538462"
##
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.704318936877076"
## [1] "Recall of 0 : 0.404494382022472"
## [1] "f score of 0 : 0.447204968944099"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.704318936877076"
## [1] "Recall of 0 : 0.404494382022472"
## [1] "f score of 0 : 0.447204968944099"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.700996677740864"
## [1] "Recall of 0 : 0.415730337078652"
## [1] "f score of 0 : 0.451219512195122"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.710963455149502"
## [1] "Recall of 0 : 0.348314606741573"
## [1] "f score of 0 : 0.416107382550336"
##
##
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
##
```

```
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.370786516853933"
## [1] "f score of 0 : 0.420382165605096"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.697674418604651"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.467836257309941"
```

Best Parameters: C5.0 Tree : minbucket = 10 and minsplit =100

```r
# Detrminig best minsplit
bucket <- 10
split <- c(80,90,100,120)

for (i in split){
  print(paste("For bucket =",bucket,"and split =",i))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
              control = rpart.control(minbucket = bucket, minsplit = i, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 10 and split = 80"
## [1] "Test accuracy : 0.724252491694352"
## [1] "Recall of 0 : 0.393258426966292"
## [1] "f score of 0 : 0.457516339869281"
##
##
##
## [1] "For bucket = 10 and split = 90"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.481927710843373"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.449438202247191"
```

```
## [1] "f score of 0 : 0.481927710843373"
##
##
##
## [1] "For bucket = 10 and split = 120"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.325842696629214"
## [1] "f score of 0 : 0.402777777777778"
```

```
# Detrminig best minbucket

bucket <- c(5,10,15)
split <- 100

for (i in bucket){
  print(paste("For bucket =",i,"and split =",split))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                control = rpart.control(minbucket = i, minsplit = split, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 5 and split = 100"
## [1] "Test accuracy : 0.704318936877076"
## [1] "Recall of 0 : 0.415730337078652"
## [1] "f score of 0 : 0.45398773006135"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.481927710843373"
##
##
##
## [1] "For bucket = 15 and split = 100"
## [1] "Test accuracy : 0.714285714285714"
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.481927710843373"
```

Hence Best parameters for C5.0, 70:30 split is: - minbucket = 10, minsplit = 100

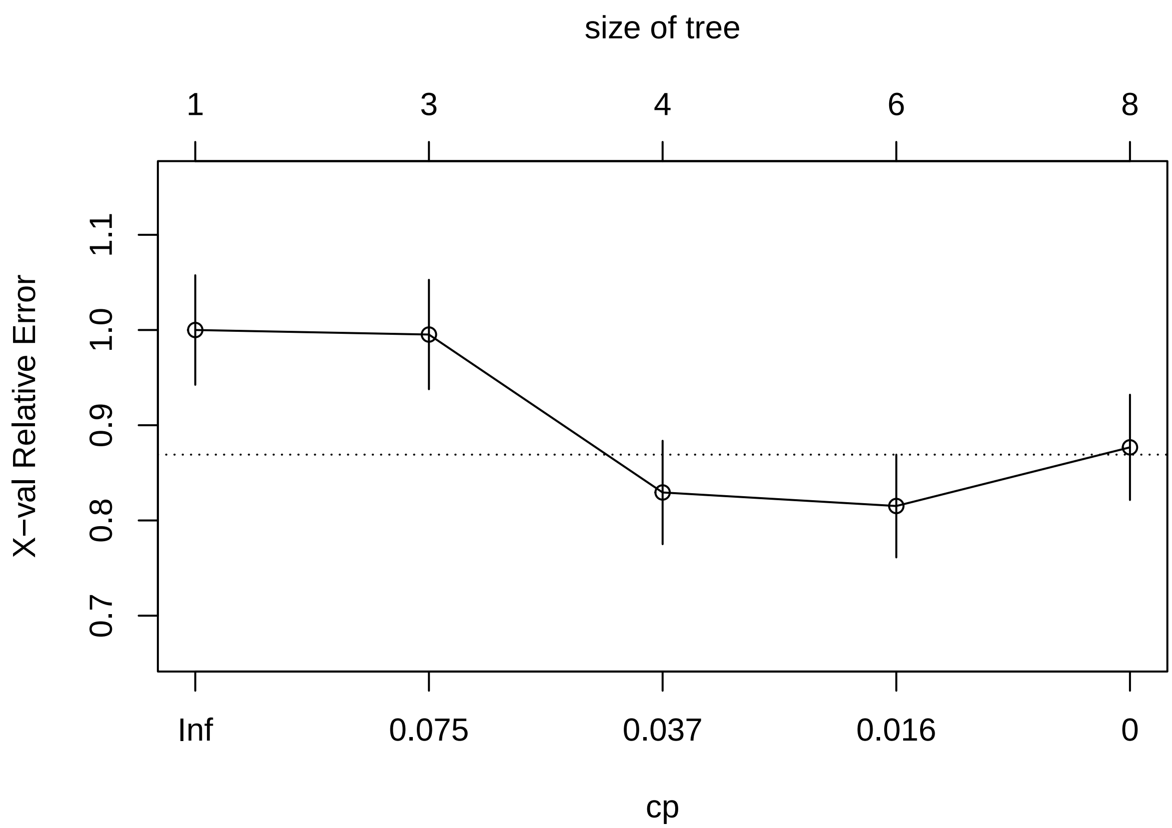
```
set.seed(50)
tree_model_tune <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                control = rpart.control(minbucket = 10, minsplit = 100, cp=0))

printcp(tree_model_tune)
```

```
##
## Classification tree:
```

```
## rpart(formula = RESPONSE ~ ., data = train, parms = list(split = "information"),
##     control = rpart.control(minbucket = 10, minsplit = 100, cp = 0))
##
## Variables actually used in tree construction:
## [1] AMOUNT      CHK_ACCT    DURATION    REAL_ESTATE SAV_ACCT    USED_CAR
##
## Root node error: 211/699 = 0.30186
##
## n= 699
##
##         CP nsplit rel error  xerror     xstd
## 1 0.078199      0   1.00000 1.00000 0.057521
## 2 0.071090      2   0.84360 0.99526 0.057444
## 3 0.018957      3   0.77251 0.82938 0.054283
## 4 0.014218      5   0.73460 0.81517 0.053970
## 5 0.000000      7   0.70616 0.87678 0.055277
```

```
plotcp(tree_model_tune)
```

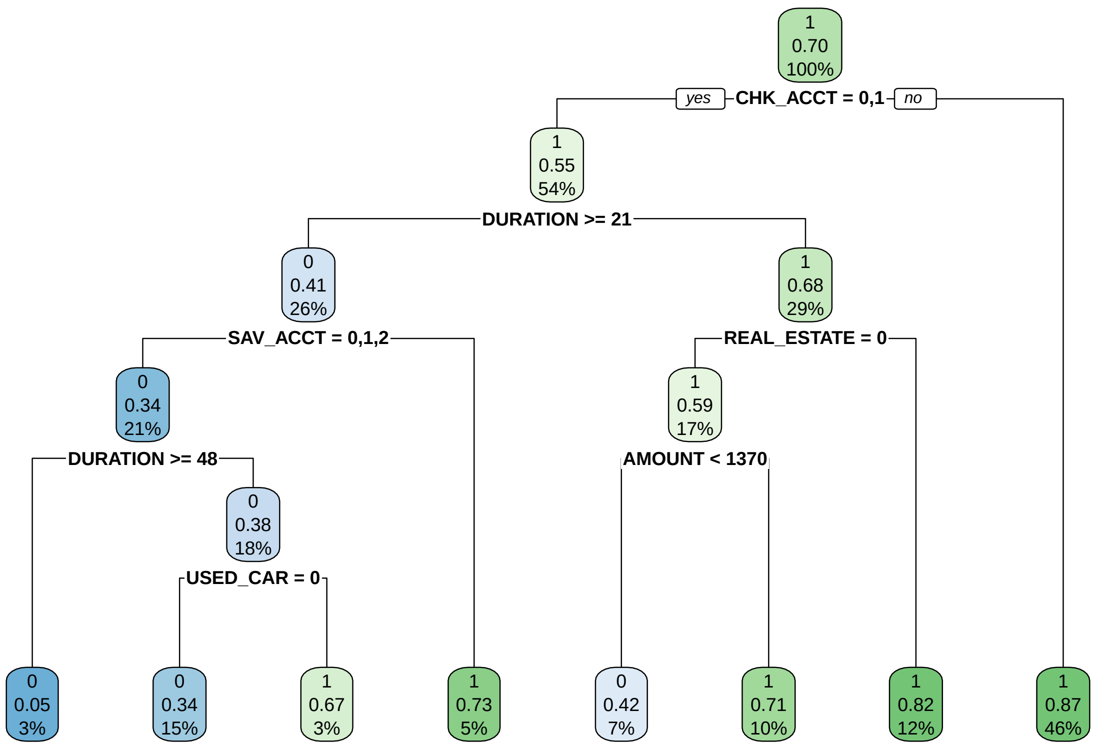



```
cp <- 0.014218
prunedTree <- prune(tree_model2, cp = cp)
pred_test_prune <- predict(prunedTree, test, type = "class")
cm_test_prune <- table(test$RESPONSE, pred_test_prune)
metrics_0(cm_test_prune)
```

```
## [1] "Test accuracy : 0.714285714285714"
```

```
## [1] "Recall of 0 : 0.449438202247191"
## [1] "f score of 0 : 0.481927710843373"
```

```r
rpart.plot(prunedTree)
```




```r
tree_73_C5 <-rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                   control = rpart.control(minbucket = 10, minsplit = 100, cp=cp))
pred_train_C5 <- predict(tree_73_C5, train, type = "class")
pred_test_C5 <- predict(tree_73_C5, test, type = "class")
cm_train_C5 <- table(train$RESPONSE, pred_train_C5)
cm_test_C5 <- table(test$RESPONSE, pred_test_C5)
metrics(cm_train_C5, cm_test_C5)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_C5
##       0   1
##   0 119  92
##   1  57 431
## [1] "Train accuracy : 0.786838340486409"
##
##
## [1] "Classification Report (traning data) :"
```

```
##   target_variable precision    recall   f_score
## 0               0 0.6761364 0.5639810 0.6149871
## 1               1 0.8240918 0.8831967 0.8526212
##
##
## [1] "                       FOR TESTING DATA                      "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5
##       0   1
##   0  40  49
##   1  37 175
## [1] "Test accuracy : 0.714285714285714"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0 0.5194805 0.4494382 0.4819277
## 1               1 0.7812500 0.8254717 0.8027523
```

```
tree_73_C5_loss <-rpart(RESPONSE ~ ., train, parms = list(split = "information", loss=loss_m),
                        control = rpart.control(minbucket = 10, minsplit = 100, cp=0.00416))
pred_train_C5_loss <- predict(tree_73_C5_loss, train, type = "class")
pred_test_C5_loss <- predict(tree_73_C5_loss, test, type = "class")
cm_train_C5_loss <- table(train$RESPONSE, pred_train_C5_loss)
cm_test_C5_loss <- table(test$RESPONSE, pred_test_C5_loss)
metrics(cm_train_C5_loss, cm_test_C5_loss)
```

```
## [1] "                     FOR TRANING DATA                      "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_C5_loss
##       0   1
##   0 210   1
##   1 319 169
## [1] "Train accuracy : 0.542203147353362"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.3969754 0.9952607 0.5675676
## 1               1 0.9941176 0.3463115 0.5136778
##
##
## [1] "                       FOR TESTING DATA                      "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5_loss
##       0   1
##   0  80   9
##   1 144  68
```

```
## [1] "Test accuracy : 0.491694352159468"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0 0.3571429 0.8988764 0.5111821
## 1               1 0.8831169 0.3207547 0.4705882
```

Inference:

From the above classification report we can observe that the recall rate of category 0 has increased from 45% to nearly 90%. But also FN cases is also much higher than the 80:20 split tree. (this is due to higher number of test cases)

Objectively speaking between 80:20 and 70:30 we can't clearly call a winner as of now, since we to compare them side by side. But we can observe that the proportion of FN is comparatively slightly lesser than 80:20 split.

50 : 50 Split

```
set.seed(5)
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.5, 0.5))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]
```

```
bucket <- c(10,20,30)
split <- c(50,75,100)

print("-------------------------  For C5.0 Tree  -------------------------------")
```

```
## [1] "-------------------------  For C5.0 Tree  -------------------------------"
```

```
for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                         control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.699410609037328"
## [1] "Recall of 0 : 0.479452054794521"
## [1] "f score of 0 : 0.477815699658703"
##
```

```
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.713163064833006"
## [1] "Recall of 0 : 0.445205479452055"
## [1] "f score of 0 : 0.471014492753623"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.660117878192534"
## [1] "Recall of 0 : 0.602739726027397"
## [1] "f score of 0 : 0.504297994269341"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.705304518664047"
## [1] "Recall of 0 : 0.554794520547945"
## [1] "f score of 0 : 0.519230769230769"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.738703339882122"
## [1] "Recall of 0 : 0.445205479452055"
## [1] "f score of 0 : 0.494296577946768"
##
##
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.68762278978389"
## [1] "Recall of 0 : 0.595890410958904"
## [1] "f score of 0 : 0.522522522522523"
##
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.722986247544204"
## [1] "Recall of 0 : 0.472602739726027"
## [1] "f score of 0 : 0.494623655913978"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.722986247544204"
## [1] "Recall of 0 : 0.472602739726027"
## [1] "f score of 0 : 0.494623655913978"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.68762278978389"
## [1] "Recall of 0 : 0.595890410958904"
```

```
## [1] "f score of 0 : 0.522522522522523"
```

```
print("------------------------  For C&R Tree -----------------------------------")
```

```
## [1] "------------------------  For C&R Tree -----------------------------------"
```

```
for (i in bucket){
  for (j in split){
    print(paste("For bucket =",i,"and split =",j))
    tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "gini"),
                     control = rpart.control(minbucket = i, minsplit = j, maxdepth = 10, cp =0))
    pred_test <- predict(tree_model2, test, type = "class")
    cm_test <- table(test$RESPONSE, pred_test)
    metrics_0(cm_test)
    writeLines("\n\n")

  }
}
```

```
## [1] "For bucket = 10 and split = 50"
## [1] "Test accuracy : 0.703339882121807"
## [1] "Recall of 0 : 0.486301369863014"
## [1] "f score of 0 : 0.484641638225256"
##
##
##
## [1] "For bucket = 10 and split = 75"
## [1] "Test accuracy : 0.703339882121807"
## [1] "Recall of 0 : 0.486301369863014"
## [1] "f score of 0 : 0.484641638225256"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.695481335952849"
## [1] "Recall of 0 : 0.438356164383562"
## [1] "f score of 0 : 0.452296819787986"
##
##
##
## [1] "For bucket = 20 and split = 50"
## [1] "Test accuracy : 0.744597249508841"
## [1] "Recall of 0 : 0.479452054794521"
## [1] "f score of 0 : 0.518518518518518"
##
##
##
## [1] "For bucket = 20 and split = 75"
## [1] "Test accuracy : 0.709233791748527"
## [1] "Recall of 0 : 0.568493150684932"
## [1] "f score of 0 : 0.528662420382166"
##
##
```

```
##
## [1] "For bucket = 20 and split = 100"
## [1] "Test accuracy : 0.709233791748527"
## [1] "Recall of 0 : 0.568493150684932"
## [1] "f score of 0 : 0.528662420382166"
##
##
##
## [1] "For bucket = 30 and split = 50"
## [1] "Test accuracy : 0.744597249508841"
## [1] "Recall of 0 : 0.479452054794521"
## [1] "f score of 0 : 0.518518518518518"
##
##
##
## [1] "For bucket = 30 and split = 75"
## [1] "Test accuracy : 0.709233791748527"
## [1] "Recall of 0 : 0.568493150684932"
## [1] "f score of 0 : 0.528662420382166"
##
##
##
## [1] "For bucket = 30 and split = 100"
## [1] "Test accuracy : 0.709233791748527"
## [1] "Recall of 0 : 0.568493150684932"
## [1] "f score of 0 : 0.528662420382166"
```

Considering C5.0, minbucket = 10, minsplit = 100 as it gave best results.

```
# Detrminig best minsplit
bucket <- 10
split <- c(80,90,100,120)

for (i in split){
  print(paste("For bucket =",bucket,"and split =",i))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
              control = rpart.control(minbucket = bucket, minsplit = i, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 10 and split = 80"
## [1] "Test accuracy : 0.713163064833006"
## [1] "Recall of 0 : 0.445205479452055"
## [1] "f score of 0 : 0.471014492753623"
##
##
##
## [1] "For bucket = 10 and split = 90"
## [1] "Test accuracy : 0.660117878192534"
## [1] "Recall of 0 : 0.602739726027397"
## [1] "f score of 0 : 0.504297994269341"
```

```
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.660117878192534"
## [1] "Recall of 0 : 0.602739726027397"
## [1] "f score of 0 : 0.504297994269341"
##
##
##
## [1] "For bucket = 10 and split = 120"
## [1] "Test accuracy : 0.660117878192534"
## [1] "Recall of 0 : 0.602739726027397"
## [1] "f score of 0 : 0.504297994269341"
```

```
# Detrminig best minbucket

bucket <- c(5,10,15)
split <- 100

for (i in bucket){
  print(paste("For bucket =",i,"and split =",split))
  tree_model2 <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
              control = rpart.control(minbucket = i, minsplit = split, maxdepth = 10, cp =0))
  pred_test <- predict(tree_model2, test, type = "class")
  cm_test <- table(test$RESPONSE, pred_test)
  metrics_0(cm_test)
  writeLines("\n\n")
}
```

```
## [1] "For bucket = 5 and split = 100"
## [1] "Test accuracy : 0.642436149312377"
## [1] "Recall of 0 : 0.595890410958904"
## [1] "f score of 0 : 0.48876404494382"
##
##
##
## [1] "For bucket = 10 and split = 100"
## [1] "Test accuracy : 0.660117878192534"
## [1] "Recall of 0 : 0.602739726027397"
## [1] "f score of 0 : 0.504297994269341"
##
##
##
## [1] "For bucket = 15 and split = 100"
## [1] "Test accuracy : 0.68762278978389"
## [1] "Recall of 0 : 0.595890410958904"
## [1] "f score of 0 : 0.522522522522523"
```

Best parameters for 50:50 are: - minbucket = 10 and minsplit = 100

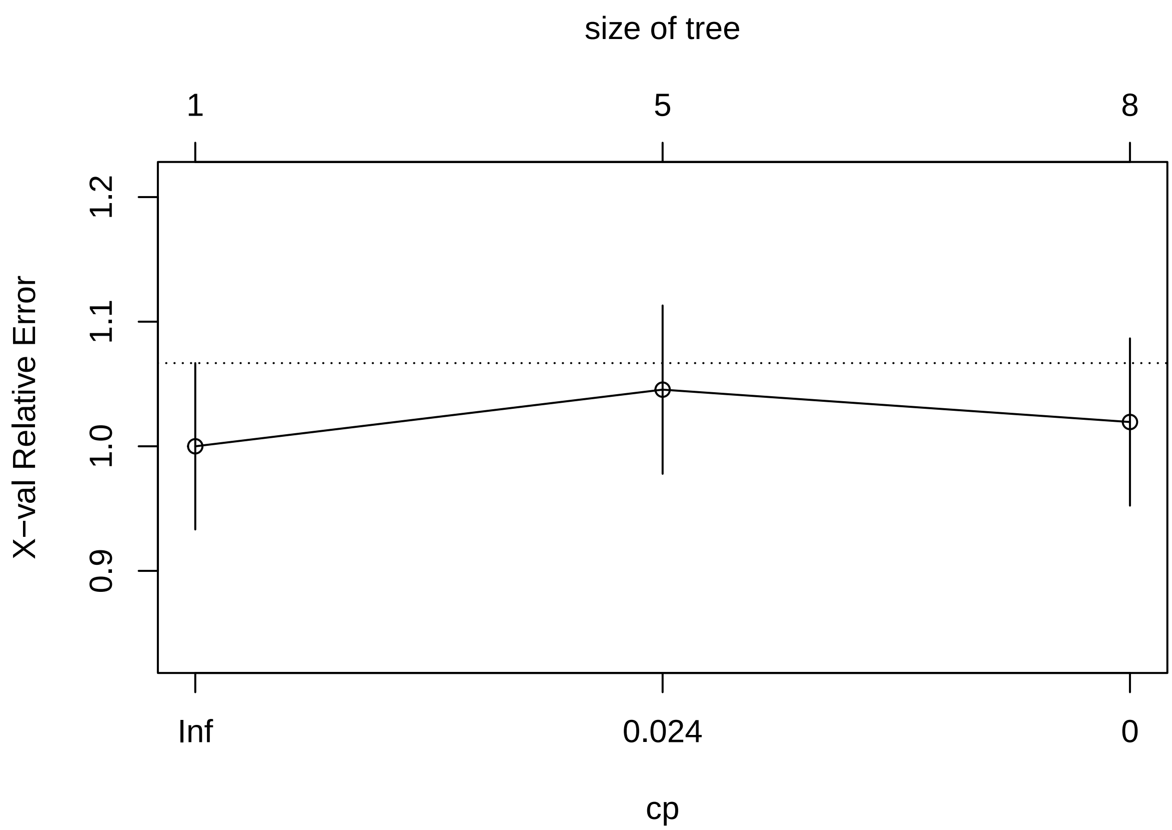
```
set.seed(50)
tree_model_tune <- rpart(RESPONSE ~ ., train, parms = list(split = "information"),
```

```
                  control = rpart.control(minbucket = 10, minsplit = 100, cp=0))

printcp(tree_model_tune)
```

```
##
## Classification tree:
## rpart(formula = RESPONSE ~ ., data = train, parms = list(split = "information"),
##     control = rpart.control(minbucket = 10, minsplit = 100, cp = 0))
##
## Variables actually used in tree construction:
## [1] AMOUNT     CHK_ACCT   DURATION   EMPLOYMENT HISTORY    NEW_CAR    RETRAINING
##
## Root node error: 154/491 = 0.31365
##
## n= 491
##
##          CP nsplit rel error xerror     xstd
## 1 0.045455      0   1.00000 1.0000 0.066760
## 2 0.012987      4   0.79870 1.0455 0.067547
## 3 0.000000      7   0.75974 1.0195 0.067106
```

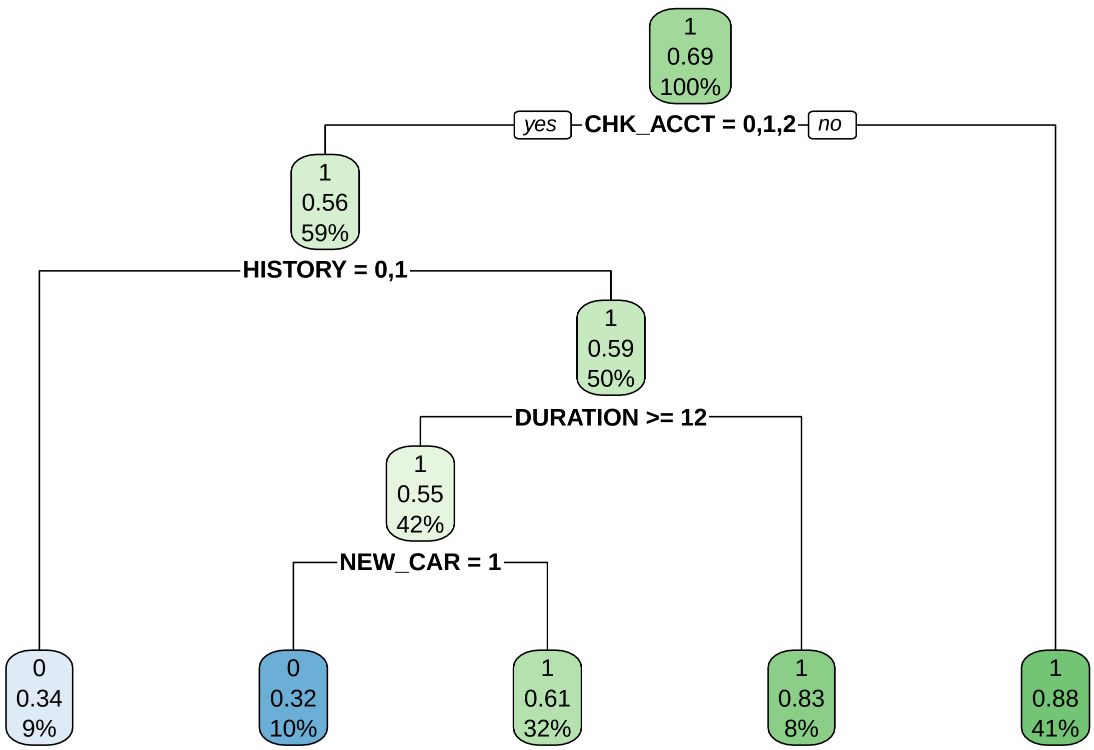
```
plotcp(tree_model_tune)
```



```r
cp <- 0.04545
prunedTree <- prune(tree_model2, cp = cp)
pred_test_prune <- predict(prunedTree, test, type = "class")
cm_test_prune <- table(test$RESPONSE, pred_test_prune)
metrics_0(cm_test_prune)
```

```
## [1] "Test accuracy : 0.726915520628684"
## [1] "Recall of 0 : 0.321917808219178"
## [1] "f score of 0 : 0.40343347639485"
```

```r
rpart.plot(prunedTree)
```




```r
tree_55_C5 <-rpart(RESPONSE ~ ., train, parms = list(split = "information"),
                   control = rpart.control(minbucket = 10, minsplit = 100, cp=cp))
pred_train_C5 <- predict(tree_55_C5, train, type = "class")
pred_test_C5 <- predict(tree_55_C5, test, type = "class")
cm_train_C5 <- table(train$RESPONSE, pred_train_C5)
cm_test_C5 <- table(test$RESPONSE, pred_test_C5)
metrics(cm_train_C5, cm_test_C5)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
```

```
##    pred_train_C5
##       0   1
##   0  61  93
##   1  30 307
## [1] "Train accuracy : 0.74949083503055"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.6703297 0.3961039 0.4979592
## 1               1 0.7675000 0.9109792 0.8331072
##
##
## [1] "                       FOR TESTING DATA                      "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5
##       0   1
##   0  47  99
##   1  40 323
## [1] "Test accuracy : 0.726915520628684"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0 0.5402299 0.3219178 0.4034335
## 1               1 0.7654028 0.8898072 0.8229299
```

```
tree_55_C5_loss <-rpart(RESPONSE ~ ., train, parms = list(split = "information", loss=loss_m),
                         control = rpart.control(minbucket = 10, minsplit = 100, cp=cp))
pred_train_C5_loss <- predict(tree_55_C5_loss, train, type = "class")
pred_test_C5_loss <- predict(tree_55_C5_loss, test, type = "class")
cm_train_C5_loss <- table(train$RESPONSE, pred_train_C5_loss)
cm_test_C5_loss <- table(test$RESPONSE, pred_test_C5_loss)
metrics(cm_train_C5_loss, cm_test_C5_loss)
```

```
## [1] "                     FOR TRANING DATA                     "
##
##
## [1] "Confusion Matrix (Train) :"
##    pred_train_C5_loss
##       0   1
##   0 149   5
##   1 206 131
## [1] "Train accuracy : 0.570264765784114"
##
##
## [1] "Classification Report (traning data) :"
##   target_variable precision    recall   f_score
## 0               0 0.4197183 0.9675325 0.5854617
## 1               1 0.9632353 0.3887240 0.5539112
##
##
```

```
## [1] "                       FOR TESTING DATA                       "
##
##
## [1] "Confusion Matrix (Test) :"
##    pred_test_C5_loss
##       0   1
##   0 134  12
##   1 248 115
## [1] "Test accuracy : 0.489194499017682"
##
##
## [1] "Classification Report (testing data) :"
##   target_variable precision    recall   f_score
## 0               0 0.3507853 0.9178082 0.5075758
## 1               1 0.9055118 0.3168044 0.4693878
```

While our Recall for 0 has improved more that 70:30 split, this is due to the enormous amount of FN cases that we got. While the FP cases are only 13 instances, the amount of good clients who are marked as bad is too much, this might have a lot of detrimental effect such as loosing clients etc.

Comparing 80:20, 70:30, 50:50 split (TEST DATA):

```r
set.seed(5)
# For 80:20
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.8, 0.2))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]

pred_test_C5_loss_82 <- predict(tree_82_C5_loss, test, type = "class")
cm_test_82 <- table(test$RESPONSE, pred_test_C5_loss_82)
print("Confusion Matrix for (80:20) with missclassification Loss weights :")
```

```
## [1] "Confusion Matrix for (80:20) with missclassification Loss weights :"
```

```r
print(cm_test_82)
```

```
##    pred_test_C5_loss_82
##      0  1
##   0 54  6
##   1 97 58
```

```r
print(paste("Test accuracy :", sum(diag(cm_test)) / sum(cm_test)))
```

```
## [1] "Test accuracy : 0.68762278978389"
```

```r
writeLines("\n")
```

```r
pr1 <- cm_test_82[2,2]/(cm_test_82[2,2]+cm_test_82[1,2])
rc1 <- cm_test_82[2,2]/(cm_test_82[2,2]+cm_test_82[2,1])
f1 <- 2*(pr1*rc1/(pr1+rc1))
pr0 <- cm_test_82[1,1]/(cm_test_82[1,1]+cm_test_82[2,1])
rc0 <- cm_test_82[1,1]/(cm_test_82[1,1]+cm_test_82[1,2])
f0 <- 2*(pr0*rc0/(pr0+rc0))
print("Classification Report (80:20) :")
```

```
## [1] "Classification Report (80:20) :"
```

```r
target_variable <- c(0,1)
precision <- c(pr0, pr1)
recall <- c(rc0, rc1)
f_score <- c(f0,f1)
cf_r_82 <- data.frame(target_variable,precision,recall,f_score)
rownames(cf_r_82) <- 0:1
print(cf_r_82)
```

```
##   target_variable precision    recall   f_score
## 0               0 0.3576159 0.9000000 0.5118483
## 1               1 0.9062500 0.3741935 0.5296804
```

```r
# For 70:30
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.7, 0.3))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]

pred_test_C5_loss_73<- predict(tree_73_C5_loss, test, type = "class")
cm_test_73 <- table(test$RESPONSE, pred_test_C5_loss_73)
print("Confusion Matrix for (70:30) with missclassification Loss weights :")
```

```
## [1] "Confusion Matrix for (70:30) with missclassification Loss weights :"
```

```r
print(cm_test_73)
```

```
##    pred_test_C5_loss_73
##       0   1
##   0  79   1
##   1 145  77
```

```r
print(paste("Test accuracy :", sum(diag(cm_test)) / sum(cm_test)))
```

```
## [1] "Test accuracy : 0.68762278978389"
```

```r
writeLines("\n")
```

```r
pr1 <- cm_test_73[2,2]/(cm_test_73[2,2]+cm_test_73[1,2])
rc1 <- cm_test_73[2,2]/(cm_test_73[2,2]+cm_test_73[2,1])
f1 <- 2*(pr1*rc1/(pr1+rc1))
```

```r
pr0 <- cm_test_73[1,1]/(cm_test_73[1,1]+cm_test_73[2,1])
rc0 <- cm_test_73[1,1]/(cm_test_73[1,1]+cm_test_73[1,2])
f0 <- 2*(pr0*rc0/(pr0+rc0))
print("Classification Report (70:30) :")
```

```
## [1] "Classification Report (70:30) :"
```

```r
target_variable <- c(0,1)
precision <- c(pr0, pr1)
recall <- c(rc0, rc1)
f_score <- c(f0,f1)
cf_r_73 <- data.frame(target_variable,precision,recall,f_score)
rownames(cf_r_73) <- 0:1
print(cf_r_73)
```

```
##   target_variable precision    recall   f_score
## 0               0 0.3526786 0.9875000 0.5197368
## 1               1 0.9871795 0.3468468 0.5133333
```

```r
# For 50:50
indx <- sample(2, nrow(df), replace= TRUE, prob = c(0.5, 0.5))
train <- df[indx == 1,  ]
test <- df[indx == 2, ]

pred_test_C5_loss_55 <- predict(tree_55_C5_loss, test, type = "class")
cm_test_55 <- table(test$RESPONSE, pred_test_C5_loss_55)
print("Confusion Matrix for (50:50) with missclassification Loss weights :")
```

```
## [1] "Confusion Matrix for (50:50) with missclassification Loss weights :"
```

```r
print(cm_test_55)
```

```
##    pred_test_C5_loss_55
##       0   1
##   0 143   6
##   1 221 114
```

```r
print(paste("Test accuracy :", sum(diag(cm_test)) / sum(cm_test)))
```

```
## [1] "Test accuracy : 0.68762278978389"
```

```r
writeLines("\n")
```

```r
pr1 <- cm_test_55[2,2]/(cm_test_55[2,2]+cm_test_55[1,2])
rc1 <- cm_test_55[2,2]/(cm_test_55[2,2]+cm_test_55[2,1])
f1 <- 2*(pr1*rc1/(pr1+rc1))
pr0 <- cm_test_55[1,1]/(cm_test_55[1,1]+cm_test_55[2,1])
rc0 <- cm_test_55[1,1]/(cm_test_55[1,1]+cm_test_55[1,2])
f0 <- 2*(pr0*rc0/(pr0+rc0))
print("Classification Report (50:50) :")
```

```
## [1] "Classification Report (50:50) :"
```

```r
target_variable <- c(0,1)
precision <- c(pr0, pr1)
recall <- c(rc0, rc1)
f_score <- c(f0,f1)
cf_r_55 <- data.frame(target_variable,precision,recall,f_score)
rownames(cf_r_55) <- 0:1
print(cf_r_55)
```

```
##   target_variable precision    recall   f_score
## 0               0 0.3928571 0.9597315 0.5575049
## 1               1 0.9500000 0.3402985 0.5010989
```

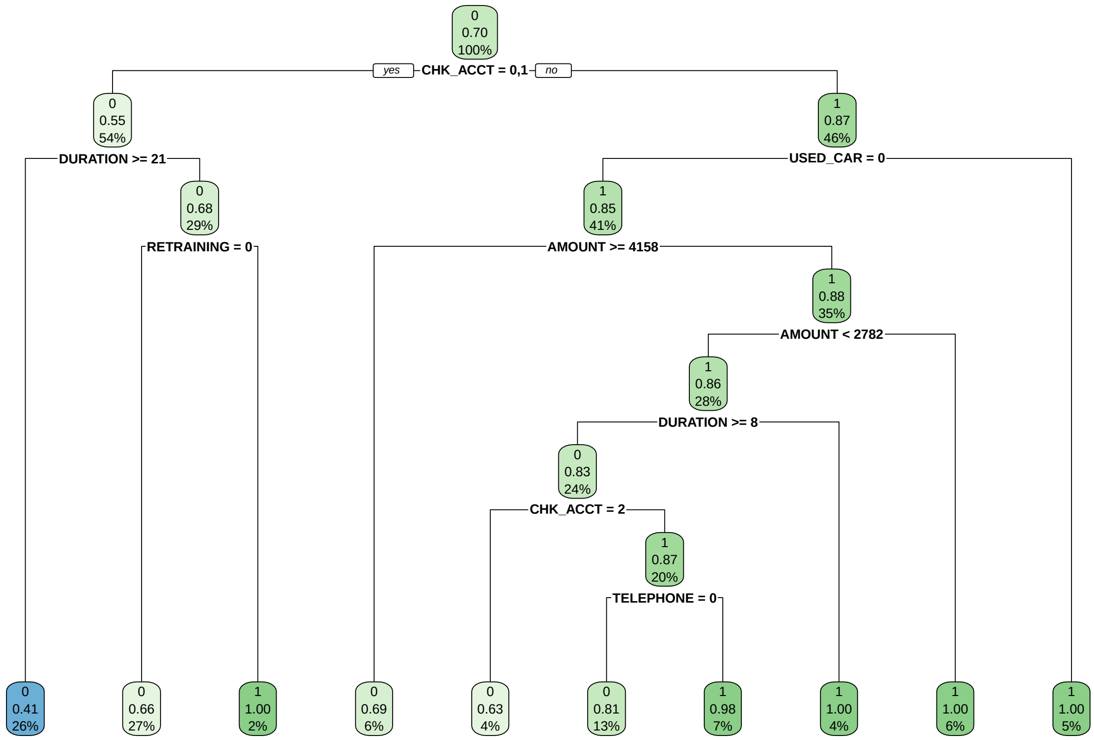
Comparing Splits:

- We can observe that 70:30 split seems to be the best case scenario, Not only is the Recall rate highest in it (97.5%), with only 2 FP out of 80 observation, but the proportion of FN is also nearly 50-60% of total which is much lesser when compared to that of other splits where more than 70% customers with good credit are predicted as bad credit. This might lead the company to loose a lot of potential customers.

The trade off between cost-cutting by reducing FP and the loss of revenue due to increased FN should be considered to decide the best ML model. For this we need further data or we can consult the company's executives.

Question (d)

```r
rpart.plot(tree_73_C5_loss)
```



From the above Tree we can derive the following decision rules for Good customers (Target =1):

The 3 strongest rules are:

- if CHK_ACCT is not = 0 or 1, and if USED_CAR is not= 0, then Target is 1 (Good customer)
- if CHK_ACCT != 0 or 1, and if USED_CAR != 0, and
  if AMOUNT is not >= 4168 and if AMOUNT is not < 2782 then GOOD CUSTOMER

-if CHK_ACCT != 0 or 1, and if USED_CAR != 0, and if AMOUNT is not >= 4168 and if AMOUNT is < 2782 if DURATION is not >= 8, then GOOD CUSTOMER

According to the best Tree model obtained, If any observation satisfies the above 3 rules then it should be categorized as RESPONE 1, i.e. Good Customer.

QUESTION (e)

FINDINGS:

- Out of 26 categorical variables, the below variables have a significant impact on the Target Variable CHK_ACCT NEW_CAR OWN_RES TELEPHONE RETRAINING PROP_UNKN_NONE MALE_SINGLE
- We can see that out of all 6 numerical variables only Duration, Amount, Age are impacting the Target variable.

- Train Test Split size: Without applying Weighted Loss for FN and FP, both 80:20 and 70:30 splits were giving us similar Recall for Bad customers., both perfrom better tha 50:50.
- Optimal minbucket and minsplit for models For 80:20 split, best values for minbucket and minsplit are - 25 & 70 respectively. For 70:30 split, best values for minbucket and minsplit are - 10 & 100 respectively. For 50:50 split, best values for minbucket and minsplit are - 10 & 100 respectively.
- In general we can observe than C5.0 models are performing better than C&R for the obtained optimal values of minsplit and minbucket.
- After applying weighted Loss for FN and FP cases, we can observe that 70:30 model outperforms the 80:20 model by a sizable margin. The Recall of Bad customers increases which means we are getting less FP cases and the proportion of FN is also much lesser than other splits. We are getting an almost equal proportion of FN and TP in 70:30, while for 80:20 amd 50:50 the proportion of FN is far greater than TP.