BUSINESS DATA MINING

(IDS 572)

Solutions to Homework 5

Group Members

- Amey Pophali (apopha2@uic.edu)
- Karthik Varanasi (vvaran3@uic.edu)
- Mrinal Dhawan (mdhawa3@uic.edu)

Problem 1 –

Status	Department	Age	Salary
Senior	Sales	31-35	46K-50K
Junior	Sales	26-30	26K-30K
Junior	Sales	31-35	31K-35K
Junior	Systems	21-25	46K-50K
Senior	Systems	36-40	66K-70K
Junior	Systems	26-30	66K-70K
Senior	Systems	41-45	66K-70K
Senior	Marketing	36-40	46K-50K
Junior	Marketing	31-35	41K-45K
Senior	Secretary	46-50	36K-40K
Junior	Secretary	26-30	26K-30K

Solutions

(a) p(status):
p(Junior) = 6/11
P(Senior) = 5/11

(b) P(department|status)

Department	
P(Sales/Junior)	1/3
P(Sales/Senior)	1/6
P(Systems/Junior)	1/3
P(Systems/Senior)	2/5
P(Marketing/Junior)	1/6
P(Marketing/senior)	1/5
P(Secretary/Junior)	1/6
P(Secretary/Senior)	1/5

P(age|status)

Age	
P(21-25/Junior)	1/6
P(21-25/senior)	0
P(26-30/junior)	1/2
P(26-30/senior)	0
P(31-35/junior)	1/3
P(31-35/senior)	1/5
P(36-40/junior)	0
P(36-40/senior)	2/5
P(41-45/Junior)	0
P(41-45/senior)	1/5
P(46-50/senior)	1/5
P(46-50/junior)	0

P(Salary|status)

Salary	
P(26k-30k/Junior)	1/3
P(26k-30k/senior)	0
P(31k - 35k/ junior)	1/6
P(31k - 35k/ senior)	0
P(36k - 40k/ senior)	1/5
P(36k - 40k/ junior)	0
P(41k - 45k/ junior)	1/6
P(41k - 45k/ senior)	0
P(46k - 50k/ junior)	1/6
P(46k - 50k/ junior)	2/5
P(66k-70k/junior)	1/6
P(66k-70k/senior)	2/5

(c) A= *{*Marketing, 31-35, 46K-50K*}* and B= *{*Sales, 31-35, 66K-70K}

P(A) = P(Marketing, 31-35, 46K-50K)*P(Junior)+ P(Marketing, 31-35, 46K-50K)*P(senior)

= 1/11*6/11 +1/11*5/11 = 0.0123

P(B) = P(Sales, 31-35, 66K-70K)*P(Junior) + P(Sales, 31-35, 66K-70K)*P(Senior)

= 0.017

P(Junior|A) = P(A|Junior)*P(Junior)/p(A) = P(Marketing|Junior)*P(31-35|Junior)*P(46k-50k|junior)*P(Junior)/P(A)

= (1/6*1/5*1/6*6/11) = 0.24

P(Senior|A) = P(A|Senior)*P(Senior)/p(A) = P(Marketing| Senior)*P(31-35| Senior)*P(46k-50k| Senior)*P(Senior)/P(A)

= (1/5*1/5*2/5*5/11)/0.0123 = .42

P(Junior|B) = P(B|Junior)*P(Junior)/p(B) = P(Sales | Junior)*P(31-35| Junior)*P(66K-70K | Junior)*P(Junior)/P(B)

= (1/3*1/3*1/6*6/11)/0.017 = 0.59

P(Senior|B) = P(B| Senior)*P(Senior)/p(B) = P(Sales | Senior)*P(31-35| Senior)*P(66K-70K | Senior)*P(Senior)/P(B)

= (1/5*1/5*2/5*5/11)/0.017 = 0.42

(d) New Table has Salary duplicate column:

Status	Department	Age	Salary	Salary_duplicate
Senior	Sales	31-35	46K-50K	46K-50K
Junior	Sales	26-30	26K-30K	26K-30K
Junior	Sales	31-35	31K-35K	31K-35K
Junior	Systems	21-25	46K-50K	46K-50K
Senior	Systems	36-40	66K-70K	66K-70K
Junior	Systems	26-30	66K-70K	66K-70K
Senior	Systems	41-45	66K-70K	66K-70K
Senior	Marketing	36-40	46K-50K	46K-50K
Junior	Marketing	31-35	41K-45K	41K-45K
Senior	Secretary	46-50	36K-40K	36K-40K
Junior	Secretary	26-30	26K-30K	26K-30K

Now the new A and B are

A= *{*Marketing, 31-35, 46K-50K,46K-50K*}* and B= *{*Sales, 31-35, 66K-70K, 66K-70K }

P(A)= P(A|Junior)*P(Junior) + P(A|Senior)*P(Senior) =
= 0.00088 + 0.00288
= 0.00376

P(B)= P(B|Junior)*P(Junior) + P(B|Senior)*P(Senior) =
= 0.0017 + 0.00288
= 0.00458

(e) c, d gives different results because there is an extra factor of the salary duplicate which gets multiplied to the probability. Hence we get a different result.

Drawing from this mathematical formula we can say that the extra factor of consideration that comes to the play with the addition of the salary which we consider as an attribute/condition to satisfy for constructing a classification model.

Problem 2 –

(a) For the analysis of the Data, we followed the following steps:

1. Get the attributes of the data.

Code:

```
#Check the dimensions of Data
dim(German.Credit)
# Check the Variabls of the Data
names(German.Credit)
str(German.Credit)
```

Output/Inference:

The given data has 100 records in total and 32 columns (NOT considering the OBS field)

The names off the fields in the data are:

```
[1] "OBS"          "CHK_ACCT"       "DURATION"      "HISTORY"         "NEW_CAR"
[6] "USED_CAR"    "FURNITURE"      "RADIO.TV"      "EDUCATION"       "RETRAINING"
[11] "AMOUNT"      "SAV_ACCT"        "EMPLOYMENT"    "INSTALL_RATE"    "MALE_DIV"
[16] "MALE_SINGLE" "MALE_MAR_or_WID" "CO.APPLICANT" "GUARANTOR"   "PRESENT_RES
IDENT"
[21] "REAL_ESTATE"    "PROP_UNKN_NONE"    "AGE"     "OTHER_INSTALL"     "RENT"
[26] "OWN_RES"     "NUM_CREDITS" "JOB"      "NUM_DEPENDENTS"    "TELEPHONE"
[31] "FOREIGN"           "RESPONSE"          "MALE_MAR_WID"
```

2. **Check the data and how it is organized:**

   To check the data, we look at the actual records of the data using "head".

   Code:

   Head(German.Credit) [German.Credit is the name of the dataset that we imported from the file)

3. **Get the Summary Statistics of the data:**

   Snippet: *summary(German.Credit)*

   Output:

```
> summary(German.Credit)
      OBS           CHK_ACCT        DURATION        HISTORY         NEW_CAR         USED_CAR        FURNITURE        RADIO.TV       EDUCATION       RETRAINING
 Min.   :   1.0   Min.   :0.000   Min.   : 4.0   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.00   Min.   :0.00   Min.   :0.000
 1st Qu.: 250.8   1st Qu.:0.000   1st Qu.:12.0   1st Qu.:2.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:0.000
 Median : 500.5   Median :1.000   Median :18.0   Median :2.000   Median :0.000   Median :0.000   Median :0.000   Median :0.00   Median :0.00   Median :0.000
 Mean   : 500.5   Mean   :1.577   Mean   :20.9   Mean   :2.545   Mean   :0.234   Mean   :0.103   Mean   :0.181   Mean   :0.28   Mean   :0.05   Mean   :0.097
 3rd Qu.: 750.2   3rd Qu.:3.000   3rd Qu.:24.0   3rd Qu.:4.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:1.00   3rd Qu.:0.00   3rd Qu.:0.000
 Max.   :1000.0   Max.   :3.000   Max.   :72.0   Max.   :4.000   Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.00   Max.   :1.00   Max.   :1.000
     AMOUNT         SAV_ACCT       EMPLOYMENT     INSTALL_RATE      MALE_DIV      MALE_SINGLE    MALE_MAR_or_WID  CO.APPLICANT      GUARANTOR     PRESENT_RESIDENT
 Min.   :  250   Min.   :0.000   Min.   :0.000   Min.   :1.000   Min.   :0.00   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :1.000
 1st Qu.: 1366   1st Qu.:0.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:2.000
 Median : 2320   Median :0.000   Median :2.000   Median :3.000   Median :0.00   Median :1.000   Median :0.000   Median :0.000   Median :0.000   Median :3.000
 Mean   : 3271   Mean   :1.105   Mean   :2.384   Mean   :2.973   Mean   :0.05   Mean   :0.548   Mean   :0.092   Mean   :0.041   Mean   :0.052   Mean   :2.845
 3rd Qu.: 3972   3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:0.00   3rd Qu.:1.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:4.000
 Max.   :18424   Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :1.00   Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :4.000
  REAL_ESTATE    PROP_UNKN_NONE        AGE       OTHER_INSTALL        RENT           OWN_RES       NUM_CREDITS         JOB        NUM_DEPENDENTS    TELEPHONE
 Min.   :0.000   Min.   :0.000   Min.   :19.00   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :1.000   Min.   :0.000   Min.   :1.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:27.00   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:0.000
 Median :0.000   Median :0.000   Median :33.00   Median :0.000   Median :0.000   Median :1.000   Median :1.000   Median :2.000   Median :1.000   Median :0.000
 Mean   :0.282   Mean   :0.154   Mean   :35.55   Mean   :0.186   Mean   :0.179   Mean   :0.713   Mean   :1.407   Mean   :1.904   Mean   :1.155   Mean   :0.404
 3rd Qu.:1.000   3rd Qu.:0.000   3rd Qu.:42.00   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:1.000
 Max.   :1.000   Max.   :1.000   Max.   :75.00   Max.   :1.000   Max.   :1.000   Max.   :1.000   Max.   :4.000   Max.   :3.000   Max.   :2.000   Max.   :1.000
    FOREIGN         RESPONSE
 Min.   :0.000   Min.   :0.0
 1st Qu.:0.000   1st Qu.:0.0
 Median :0.000   Median :1.0
 Mean   :0.037   Mean   :0.7
 3rd Qu.:0.000   3rd Qu.:1.0
 Max.   :1.000   Max.   :1.0
```

4. Get the number of Missing values in the Data

   Code : *sapply(German.Credit, function(x) sum(is.na(x)))*

Result:

```
> sapply(German.Credit, function(x) sum(is.na(x)))
           OBS       CHK_ACCT       DURATION        HISTORY        NEW_CAR       USED_CAR      FURNITURE       RADIO.TV      EDUCATION     RETRAINING
             0              0              0              0              0              0              0              0              0              0
        AMOUNT       SAV_ACCT     EMPLOYMENT   INSTALL_RATE       MALE_DIV    MALE_SINGLE MALE_MAR_or_WID   CO.APPLICANT      GUARANTOR PRESENT_RESIDENT
             0              0              0              0              0              0              0              0              0              0
   REAL_ESTATE PROP_UNKN_NONE            AGE  OTHER_INSTALL           RENT        OWN_RES    NUM_CREDITS            JOB NUM_DEPENDENTS      TELEPHONE
             0              0              0              0              0              0              0              0              0              0
       FOREIGN       RESPONSE
             0              0
>
```

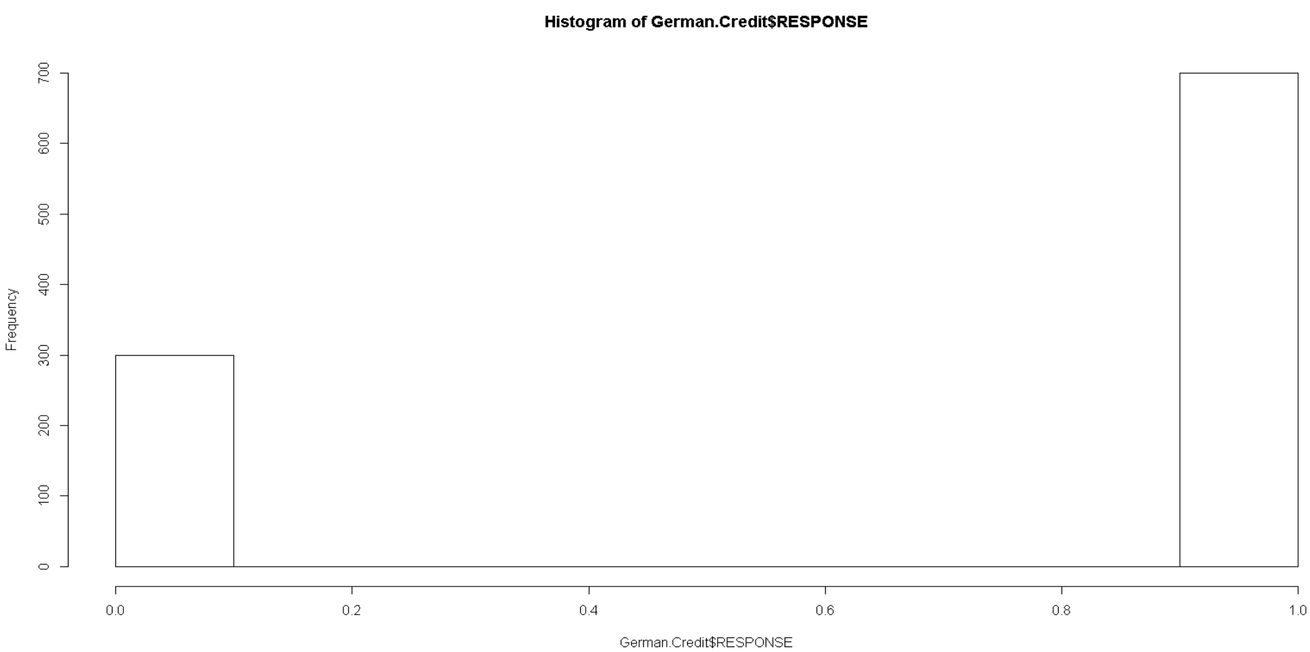
5. Get the Target variable details

Snippet:

#Get the summary of Good and Bad Creditors
summary(German.Credit$RESPONSE)
#plot the frequencies
hist(German.Credit$RESPONSE)
 # Get the number of good vs Bad credit risks
resp=table(German.Credit$RESPONSE)
t=as.data.frame(resp)
names(t)[1]='Credit Risk'
View(t)

Output:

```
> summary(German.Credit$RESPONSE)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     0.0     1.0     0.7     1.0     1.0
```



Tabular representation of the

	Credit Risk	Freq
1	**0**	**300**
2	**1**	**700**

Going deeper into the data, we analyzed the data in the following steps:

6. Aggregate the data based on the various variables and look at the distribution in each category

Account (CHK_ACCT)
Description: Checking account status
Type: Categorical
Description: The categories of the variable provided here are as follows -
0: < 0 DM
1: 0 < ...< 200 DM
2: => 200 DM
3: no checking account

Code:

```
#Aggregate of the table based on the type of account
agg_acct=table(German.Credit$CHK_ACCT)
#look at the aggregation
View(agg_acct)
```

	Type	Frequency
1	0	274
2	1	269
3	2	63
4	3	394

Code:

```
# Split the data based onthe account type
split_acct= split(German.Credit,German.Credit$CHK_ACCT)

#look at the split data
#View(split_acct)

#Look for the distribution of 'Response' in each category of Account
dist_acct_0 = table(split_acct$`0`$RESPONSE)
```

dist_acct_1 = table(split_acct$`1`$RESPONSE)
dist_acct_2 = table(split_acct$`2`$RESPONSE)
dist_acct_3 = table(split_acct$`3`$RESPONSE)

#Look at the distribution
View(dist_acct_0)
View(dist_acct_1)
View(dist_acct_2)
View(dist_acct_3)

Result:

Category:

Category	Frequency
0	274
1	269
2	63
3	394

Category 0 - CHK_ACCT< 0 DM

Response	Frequency
0	135
1	139

Category 1 - 0 < CHK_ACCT< 200 DM

Response	Frequency
0	105
1	164

Category 2 - CHK_ACCT => 200 DM

Response	Frequency
0	14
1	49

Category 3: - no checking account

Response	Frequency
0	46
1	348

Inference:

By looking at the data, we can say that a significant number of people do not have a checking account. The number of people in the categories 0 and 1 are almost the same. The response variable for category 0 is equally distributed between 0 and 1. The highest disproportion of response variable could be seen in Category 3.

HISTORY:

Description: Credit history

Type: Categorical

Description: The description of the categories of the variable provided here are

0: no credits taken

1: all credits at this bank paid back duly

2: existing credits paid back duly till now

3: delay in paying off in the past

4: critical account

Code:

```
#Aggregate of the table based on the History
agg_hist=table(German.Credit$HISTORY)
#look at the aggregation
View(agg_hist)
# Split the data based onthe history
split_hist= split(German.Credit,German.Credit$HISTORY)
#look at the split data
#View(split_hist)
#Look for the distribution of 'Response' in each category of history
dist_hist_0 = table(split_hist$`0`$RESPONSE)
dist_hist_1 = table(split_hist$`1`$RESPONSE)
dist_hist_2 = table(split_hist$`2`$RESPONSE)
dist_hist_3 = table(split_hist$`3`$RESPONSE)
dist_hist_4 = table(split_hist$`4`$RESPONSE)
#Look at the distribution / QA the distribution
View(dist_hist_0)
View(dist_hist_1)
View(dist_hist_2)
View(dist_hist_3)
View(dist_hist_4)
```

Results:

Category wise Aggregate:

Category	Frequency
0	40
1	49
2	530
3	88
4	293

Category 0:

Response	Frequency
0	25
1	15

Category 1:

Response	Frequency
0	28
1	21

Category 2:

Response	Frequency
0	169
1	361

Category 3:

Response	Frequency
0	28
1	60

Category 4:

Response	Frequency
0	50
1	243

Inference:

By looking at the data, we can say that a mostly all the existing credits are/were paid back duly till now. The number of people in the categories 0 and 1 are almost the same. A large portion (293) of the credit history is in 'Critical Account Status'.

EMPLOYMENT:

Description: Present employment since

Type: Categorical

Description: The categories of the variable provided here are as follows -

0: unemployed
1: < 1 year
2: 1 <= ... < 4 years
3: 4 <=... < 7 years
4: >= 7 years

Code:

```
#Aggregate of the table based on the Employment
agg_emp=table(German.Credit$EMPLOYMENT)
#look at the aggregation
```

```
View(agg_emp)
# Split the data based onthe history
split_emp= split(German.Credit,German.Credit$EMPLOYMENT)
#look at the split data
#View(split_emp)
#Look for the distribution of 'Response' in each category of EMPLOYMENT
dist_emp_0 = table(split_emp$`0`$RESPONSE)
dist_emp_1 = table(split_emp$`1`$RESPONSE)
dist_emp_2 = table(split_emp$`2`$RESPONSE)
dist_emp_3 = table(split_emp$`3`$RESPONSE)
dist_emp_4 = table(split_emp$`4`$RESPONSE)
#Look at the distribution / QA the distribution
View(dist_emp_0)
View(dist_emp_1)
View(dist_emp_2)
View(dist_emp_3)
View(dist_emp_4)
```

Category wise agg:

Category	Frequency
0	62
1	172
2	339
3	174
4	253

Category 0:

Response	Frequency
0	23
1	39

Category 1:

Response	Frequency
0	70
1	102

Category 2:

Response	Frequency
0	104
1	235

Category 3:

Response	Frequency
0	39

1	135

Category 4:

Response	Frequency
0	64
1	189

Inference:

By looking at the data, we can say that almost everyone is currently employed. Looking at the data above, we can infer that 938 are employed against 62 unemployed. Most number of employments are in the category 2 i.e. 1 <= ... < 4 years.

RESIDENCE STATUS (PRESENT_RESIDENT):

Description: Present resident since - years

Type: Categorical

Description: The categories of the variable provided here are as follows -

0: <= 1 year

1: <=2 years

2: 2<…<=3 years

3: > 4years

Code:

```
#Aggregate of the table based on the Present residence status
agg_res=table(German.Credit$PRESENT_RESIDENT)
#look at the aggregation
View(agg_res)
# Split the data based onthe history
split_res= split(German.Credit,German.Credit$PRESENT_RESIDENT)
#look at the split data
#View(split_res)
#Look for the distribution of 'Response' in each category of EMPLOYMENT
dist_res_0 = table(split_res$`0`$RESPONSE)
dist_res_1 = table(split_res$`1`$RESPONSE)
dist_res_2 = table(split_res$`2`$RESPONSE)
dist_res_3 = table(split_res$`3`$RESPONSE)
#Look at the distribution / QA the distribution
View(dist_res_0)
View(dist_res_1)
View(dist_res_2)
View(dist_res_3)
```

Results:

Category wise agg:

Category	Frequency
1	130
2	308
3	149
4	413

Category 1:

Response	Frequency
0	36
1	94

Category 2:

Response	Frequency
0	97
1	211

Category 3:

Response	Frequency
0	43
1	106

Category 4:

Response	Frequency
0	124
1	289

Inference:

By looking at the data, we can see that category 4 (> 4years) of residence has the most number of records, followed by the category 2 (<=2 years) of residence. Incidentally, the residence status of category 1 and category 3 is identical with values nearing ~150.

JOB:

Description: Nature of job

Type: Categorical

Description: The categories of the variable provided here are as follows -

0: unemployed/unskilled/non-resident

1: unskilled-resident

2: skilled employee / official

3: management/self-employed/highly qualified employee/officer

Code:

```
#Aggregate of the table based on the Job
agg_job=table(German.Credit$JOB)
```

```
#look at the aggregation
View(agg_job)
# Split the data based onthe history
split_job= split(German.Credit,German.Credit$JOB)
#Look for the distribution of 'Response' in each category of
EMPLOYMENT
dist_job_0 = table(split_job$`0`$RESPONSE)
dist_job_1 = table(split_job$`1`$RESPONSE)
dist_job_2 = table(split_job$`2`$RESPONSE)
dist_job_3 = table(split_job$`3`$RESPONSE)
#Look at the distribution / QA the distribution
View(dist_job_0)
View(dist_job_1)
View(dist_job_2)
View(dist_job_3)
#Look at the distribution / QA the distribution
View(dist_res_0)
View(dist_res_1)
View(dist_res_2)
View(dist_res_3)
```

Results:

Category wise agg:

Category	Frequency
0	22
1	200
2	630
3	148

Category01:

Response	Frequency
0	7
1	15

Category 1:

Response	Frequency
0	56
1	144

Category 2:

Response	Frequency
0	186
1	444

Category 3:

Response	Frequency
0	51
1	97

Inference:

By looking at the data, we can infer that most number of jobs are of the category 2 (skilled employee / official) with number 630. There are very less number of people who fall in the category 0 i.e. unemployed/unskilled/non-resident (22). This is very less when compared to total of 978 total jobs.

FOREIGN:

Description: Foreign worker

Type: Binary

Description: The categories of the variable provided here are as follows -

0: No

1: Yes

Code:

```
#Aggregate of the table based on the Nationality
agg_frn=table(German.Credit$FOREIGN)
#look at the aggregation
View(agg_frn)
# Split the data based onthe history
split_frn= split(German.Credit,German.Credit$FOREIGN)
#Look for the distribution of 'Response' in each category of
EMPLOYMENT
dist_frn_0 = table(split_frn$`0`$RESPONSE)
dist_frn_1 = table(split_frn$`1`$RESPONSE)
#Look at the distribution / QA the distribution
View(dist_frn_0)
View(dist_frn_1)
```

Results:

Category wise agg:

Category	Frequency
0	963
1	37

Category01:

Response	Frequency
0	296
1	667

Category 1:

Response	Frequency
0	4
1	33

Inference:
By looking at the data, we can infer that for every 1000 people working, we have 963 people who are residents of the nation vs 37 people who are foreigners.

(b) The main variables that we seemed as important are:

"CHK_ACCT","DURATION","HISTORY", "OBS","SAV_ACCT", "OTHER_INSTALL"

The Misclassification error rate of using the following model is: 0.235 = 235 / 1000

As the variables are considered as continuous variables, before everything, we convert the variable into respective factors. This is achieved by using the following code:

Code:

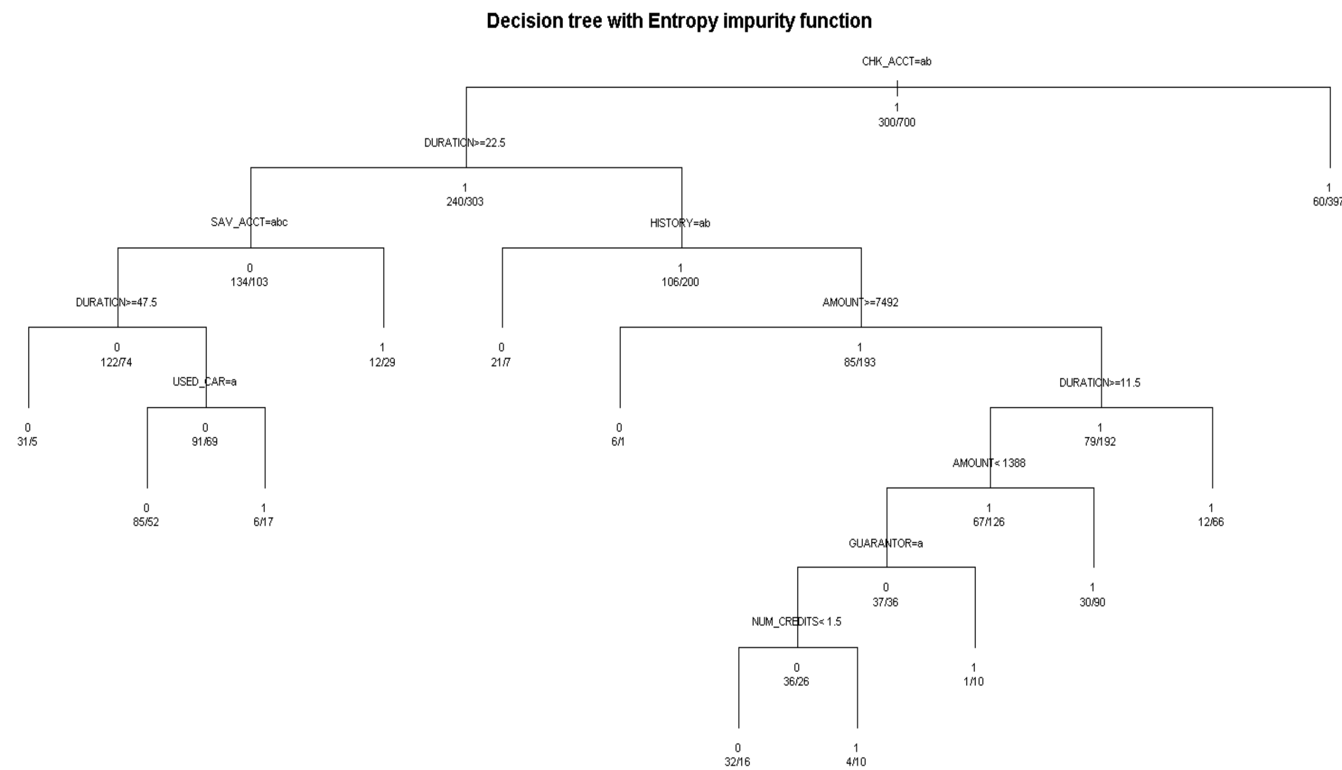
```
#Look at the data
str(German.Credit)
#observe that the values are taken as continous values but actually they are categorical.
#Converting them to categorical values
German.Credit$RESPONSE= factor(German.Credit$RESPONSE)
German.Credit$FOREIGN = factor(German.Credit$FOREIGN)
German.Credit$TELEPHONE = factor (German.Credit$TELEPHONE)
German.Credit$JOB = factor(German.Credit$JOB)
German.Credit$OWN_RES = factor(German.Credit$OWN_RES)
German.Credit$RENT = factor(German.Credit$RENT)
German.Credit$OTHER_INSTALL = factor(German.Credit$OTHER_INSTALL)
German.Credit$PROP_UNKN_NONE = factor(German.Credit$PROP_UNKN_NONE)
German.Credit$REAL_ESTATE = factor(German.Credit$REAL_ESTATE)
German.Credit$PRESENT_RESIDENT = factor(German.Credit$PRESENT_RESIDENT)
German.Credit$GUARANTOR = factor(German.Credit$GUARANTOR)
German.Credit$CO.APPLICANT = factor(German.Credit$CO.APPLICANT)
German.Credit$MALE_MAR_WID = factor(German.Credit$MALE_MAR_or_WID)
German.Credit$MALE_SINGLE = factor(German.Credit$MALE_SINGLE)
German.Credit$MALE_DIV = factor(German.Credit$MALE_DIV)
German.Credit$EMPLOYMENT = factor(German.Credit$EMPLOYMENT)
German.Credit$SAV_ACCT = factor(German.Credit$SAV_ACCT)
German.Credit$RETRAINING = factor(German.Credit$RETRAINING)
German.Credit$EDUCATION = factor(German.Credit$EDUCATION)
German.Credit$RADIO.TV = factor(German.Credit$RADIO.TV)
German.Credit$FURNITURE = factor(German.Credit$FURNITURE)
```

German.Credit$USED_CAR = factor(German.Credit$USED_CAR)
German.Credit$NEW_CAR = factor(German.Credit$NEW_CAR)
German.Credit$HISTORY = factor(German.Credit$HISTORY)
German.Credit$CHK_ACCT = factor(German.Credit$CHK_ACCT)

Plot the Decision Tree:

Code:
 # Plot the Decision Tree
fit = rpart(German.Credit$RESPONSE ~ . , method="class", data = German.Credit[,2:32])
plot(fit, uniform = T,main ='Decision tree with Entropy impurity function')
text(fit, use.n = T, all = T, cex = 0.7, xpd = T)
summary(fit)

Graph:



For the classification tree, the following are the summary statistics obtained from summary(fit)

```
rpart(formula = German.Credit$RESPONSE ~ ., data = German.Credit[,
    2:32], method = "class")
  n= 1000

          CP nsplit rel error    xerror       xstd
```

```
1 0.05166667      0 1.0000000 1.0000000 0.04830459
2 0.04666667      3 0.8400000 1.0033333 0.04835046
3 0.01833333      4 0.7933333 0.8666667 0.04623611
4 0.01666667      6 0.7566667 0.8366667 0.04570424
5 0.01111111      7 0.7400000 0.8200000 0.04539750
6 0.01000000     11 0.6866667 0.8466667 0.04588440

Variable importance
CHK_ACCT  DURATION  HISTORY    AMOUNT        SAV_ACCT        USED_CAR
GUARANTOR
   33       15          12      12              10                4
3

NUM_CREDITS PRESENT_RESIDENT   PROP_UKN_NONE   RADIO.TV  EMPLOYMENT AGE  REA
L_ESTATE
         2                2                2       2            1     1
1
             JOB
               1

Node number 1: 1000 observations,  complexity param=0.05166667
  predicted class=1  expected loss=0.3  P(node) =1
    class counts:   300   700
   probabilities: 0.300 0.700
  left son=2 (543 obs) right son=3 (457 obs)
  Primary splits:
      CHK_ACCT splits as  LLRR,        improve=47.90962, (0 missing)
      HISTORY  splits as  LLRRR,       improve=17.06212, (0 missing)
      SAV_ACCT splits as  LLRRR,       improve=14.80642, (0 missing)
      DURATION < 34.5    to the right, improve=13.62155, (0 missing)
      AMOUNT   < 3913.5  to the right, improve=11.32017, (0 missing)
  Surrogate splits:
      SAV_ACCT         splits as  LLRRR, agree=0.611, adj=0.149, (0 split)
      HISTORY          splits as  LLLLR, agree=0.592, adj=0.107, (0 split)
      PRESENT_RESIDENT splits as  LRLL,  agree=0.567, adj=0.053, (0 split)
      RADIO.TV         splits as  LR,    agree=0.565, adj=0.048, (0 split)
      EMPLOYMENT       splits as  LLLLR, agree=0.554, adj=0.024, (0 split)

Node number 2: 543 observations,    complexity param=0.05166667
  predicted class=1  expected loss=0.441989  P(node) =0.543
    class counts:   240   303
   probabilities: 0.442 0.558
  left son=4 (237 obs) right son=5 (306 obs)
  Primary splits:
      DURATION    < 22.5    to the right, improve=12.810640, (0 missing)
      HISTORY     splits as  LLRRR,       improve= 9.653787, (0 missing)
      REAL_ESTATE splits as  LR,          improve= 9.181363, (0 missing)
      SAV_ACCT    splits as  LLRRR,       improve= 8.890786, (0 missing)
      AMOUNT      < 8079    to the right, improve= 6.601270, (0 missing)
  Surrogate splits:
      AMOUNT         < 2805.5  to the right, agree=0.748, adj=0.422, (0 split
)
      PROP_UNKN_NONE splits as  RL,          agree=0.643, adj=0.181, (0 split
)
      HISTORY        splits as  LLRLR,       agree=0.610, adj=0.105, (0 split
)
      USED_CAR       splits as  RL,          agree=0.599, adj=0.080, (0 split
)
      REAL_ESTATE    splits as  LR,          agree=0.597, adj=0.076, (0 split
)

Node number 3: 457 observations
  predicted class=1  expected loss=0.131291  P(node) =0.457
    class counts:    60   397
```

```
   probabilities: 0.131 0.869

Node number 4: 237 observations,    complexity param=0.05166667
  predicted class=0  expected loss=0.4345992  P(node) =0.237
    class counts:   134   103
   probabilities: 0.565 0.435
  left son=8 (196 obs) right son=9 (41 obs)
  Primary splits:
      SAV_ACCT     splits as  LLLRR,       improve=7.374515, (0 missing)
      USED_CAR     splits as  LR,          improve=4.129437, (0 missing)
      AMOUNT       < 1381.5  to the left,  improve=3.289316, (0 missing)
      INSTALL_RATE < 2.5     to the right, improve=3.067516, (0 missing)
      DURATION     < 43.5    to the right, improve=2.564920, (0 missing)

Node number 5: 306 observations,    complexity param=0.04666667
  predicted class=1  expected loss=0.3464052  P(node) =0.306
    class counts:   106   200
   probabilities: 0.346 0.654
  left son=10 (28 obs) right son=11 (278 obs)
  Primary splits:
      HISTORY     splits as  LLRRR,       improve=10.040510, (0 missing)
      REAL_ESTATE splits as  LR,          improve= 5.585685, (0 missing)
      GUARANTOR   splits as  LR,          improve= 3.782059, (0 missing)
      DURATION    < 11.5    to the right, improve= 3.766531, (0 missing)
      AMOUNT      < 7491.5  to the right, improve= 3.737438, (0 missing)

Node number 8: 196 observations,    complexity param=0.01833333
  predicted class=0  expected loss=0.377551  P(node) =0.196
    class counts:   122    74
   probabilities: 0.622 0.378
  left son=16 (36 obs) right son=17 (160 obs)
  Primary splits:
      DURATION         < 47.5    to the right, improve=5.023838, (0 missing)
      USED_CAR         splits as  LR,          improve=4.598639, (0 missing)
      INSTALL_RATE     < 2.5     to the right, improve=2.682485, (0 missing)
      PRESENT_RESIDENT splits as  RLRL,        improve=2.610062, (0 missing)
      AMOUNT           < 11788   to the right, improve=2.516732, (0 missing)
  Surrogate splits:
      AMOUNT < 13319.5 to the right, agree=0.837, adj=0.111, (0 split)

Node number 9: 41 observations
  predicted class=1  expected loss=0.2926829  P(node) =0.041
    class counts:    12    29
   probabilities: 0.293 0.707

Node number 10: 28 observations
  predicted class=0  expected loss=0.25  P(node) =0.028
    class counts:    21     7
   probabilities: 0.750 0.250

Node number 11: 278 observations,    complexity param=0.01666667
  predicted class=1  expected loss=0.3057554  P(node) =0.278
    class counts:    85   193
   probabilities: 0.306 0.694
  left son=22 (7 obs) right son=23 (271 obs)
  Primary splits:
      AMOUNT      < 7491.5  to the right, improve=4.366338, (0 missing)
      DURATION    < 11.5    to the right, improve=3.840775, (0 missing)
      REAL_ESTATE splits as  LR,          improve=3.589042, (0 missing)
      EMPLOYMENT  splits as  LLLRL,       improve=3.449347, (0 missing)
      HISTORY     splits as  --LRR,       improve=2.954088, (0 missing)

Node number 16: 36 observations
  predicted class=0  expected loss=0.1388889  P(node) =0.036
```

```
    class counts:    31     5
   probabilities: 0.861 0.139

Node number 17: 160 observations,    complexity param=0.01833333
  predicted class=0  expected loss=0.43125  P(node) =0.16
    class counts:    91    69
   probabilities: 0.569 0.431
  left son=34 (137 obs) right son=35 (23 obs)
  Primary splits:
      USED_CAR     splits as  LR,           improve=5.092387, (0 missing)
      AMOUNT       < 2313    to the left,  improve=3.402464, (0 missing)
      INSTALL_RATE < 2.5     to the right, improve=2.374236, (0 missing)
      NEW_CAR      splits as  RL,           improve=2.000321, (0 missing)
      AGE          < 57.5    to the left,  improve=1.711184, (0 missing)
  Surrogate splits:
      AGE < 62      to the left,  agree=0.862, adj=0.043, (0 split)

Node number 22: 7 observations
  predicted class=0  expected loss=0.1428571  P(node) =0.007
    class counts:     6     1
   probabilities: 0.857 0.143

Node number 23: 271 observations,    complexity param=0.01111111
  predicted class=1  expected loss=0.2915129  P(node) =0.271
    class counts:    79   192
   probabilities: 0.292 0.708
  left son=46 (193 obs) right son=47 (78 obs)
  Primary splits:
      DURATION    < 11.5    to the right, improve=4.151402, (0 missing)
      AMOUNT      < 1373    to the left,  improve=3.770882, (0 missing)
      EDUCATION   splits as  RL,           improve=3.465097, (0 missing)
      EMPLOYMENT  splits as  LLLRL,        improve=2.956763, (0 missing)
      REAL_ESTATE splits as  LR,           improve=2.672491, (0 missing)
  Surrogate splits:
      AMOUNT  < 527.5   to the right, agree=0.742, adj=0.103, (0 split)
      FOREIGN splits as  LR,          agree=0.723, adj=0.038, (0 split)
      AGE     < 66.5    to the left,  agree=0.720, adj=0.026, (0 split)

Node number 34: 137 observations
  predicted class=0  expected loss=0.379562  P(node) =0.137
    class counts:    85    52
   probabilities: 0.620 0.380

Node number 35: 23 observations
  predicted class=1  expected loss=0.2608696  P(node) =0.023
    class counts:     6    17
   probabilities: 0.261 0.739

Node number 46: 193 observations,    complexity param=0.01111111
  predicted class=1  expected loss=0.3471503  P(node) =0.193
    class counts:    67   126
   probabilities: 0.347 0.653
  left son=92 (73 obs) right son=93 (120 obs)
  Primary splits:
      AMOUNT     < 1387.5  to the left,  improve=5.988715, (0 missing)
      CHK_ACCT   splits as  LR--,        improve=2.224992, (0 missing)
      EMPLOYMENT splits as  LLLRL,       improve=2.084052, (0 missing)
      GUARANTOR  splits as  LR,          improve=1.966915, (0 missing)
      SAV_ACCT   splits as  LLRRL,       improve=1.963817, (0 missing)
  Surrogate splits:
      INSTALL_RATE < 3.5     to the right, agree=0.658, adj=0.096, (0 split)
      JOB          splits as  RLRR,        agree=0.658, adj=0.096, (0 split)
      AGE          < 21.5    to the left,  agree=0.653, adj=0.082, (0 split)
      DURATION     < 12.5    to the left,  agree=0.648, adj=0.068, (0 split)
```

```
      EDUCATION    splits as  RL,          agree=0.642, adj=0.055, (0 split)

Node number 47: 78 observations
  predicted class=1  expected loss=0.1538462  P(node) =0.078
    class counts:    12    66
   probabilities: 0.154 0.846

Node number 92: 73 observations,    complexity param=0.01111111
  predicted class=0  expected loss=0.4931507  P(node) =0.073
    class counts:    37    36
   probabilities: 0.507 0.493
  left son=184 (62 obs) right son=185 (11 obs)
  Primary splits:
      GUARANTOR   splits as  LR,          improve=4.481420, (0 missing)
      REAL_ESTATE splits as  LR,          improve=4.304126, (0 missing)
      NUM_CREDITS < 1.5     to the left,  improve=3.050656, (0 missing)
      NEW_CAR     splits as  RL,          improve=2.394020, (0 missing)
      JOB         splits as  LRLR,        improve=2.001678, (0 missing)

Node number 93: 120 observations
  predicted class=1  expected loss=0.25  P(node) =0.12
    class counts:    30    90
   probabilities: 0.250 0.750

Node number 184: 62 observations,    complexity param=0.01111111
  predicted class=0  expected loss=0.4193548  P(node) =0.062
    class counts:    36    26
   probabilities: 0.581 0.419
  left son=368 (48 obs) right son=369 (14 obs)
  Primary splits:
      NUM_CREDITS      < 1.5     to the left,  improve=3.145929, (0 missing)
      REAL_ESTATE      splits as  LR,          improve=2.621642, (0 missing)
      HISTORY          splits as  --LRR,       improve=2.451005, (0 missing)
      PRESENT_RESIDENT splits as  LLLR,        improve=2.105829, (0 missing)
      OTHER_INSTALL    splits as  RL,          improve=2.000676, (0 missing)
  Surrogate splits:
      HISTORY      splits as  --LRR,       agree=0.887, adj=0.500, (0 split)
      AMOUNT       < 612     to the right, agree=0.823, adj=0.214, (0 split)
      CO.APPLICANT splits as  LR,          agree=0.790, adj=0.071, (0 split)
      AGE          < 54.5    to the left,  agree=0.790, adj=0.071, (0 split)
      JOB          splits as  RLLL,        agree=0.790, adj=0.071, (0 split)

Node number 185: 11 observations
  predicted class=1  expected loss=0.09090909  P(node) =0.011
    class counts:     1    10
   probabilities: 0.091 0.909

Node number 368: 48 observations
  predicted class=0  expected loss=0.3333333  P(node) =0.048
    class counts:    32    16
   probabilities: 0.667 0.333

Node number 369: 14 observations
  predicted class=1  expected loss=0.2857143  P(node) =0.014
    class counts:     4    10
   probabilities: 0.286 0.714
```

Code:

confusionMatrix(predict(fit, German.Credit, type="class"), German.Credit$RESPONSE, dnn=c("Predictions", "Actual Values"), positive = "1")

The confusion matrix of the given data is:

```
Confusion Matrix and Statistics

           Actual Values
Predictions   0   1
          0 172  73
          1 128 627

               Accuracy : 0.799
                 95% CI : (0.7728, 0.8234)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 8.578e-13

                  Kappa : 0.495
 Mcnemar's Test P-Value : 0.0001396

            Sensitivity : 0.8957
            Specificity : 0.5733
         Pos Pred Value : 0.8305
         Neg Pred Value : 0.7020
             Prevalence : 0.7000
         Detection Rate : 0.6270
   Detection Prevalence : 0.7550
      Balanced Accuracy : 0.7345

       'Positive' Class : 1
```

The accuracy of the model is 0.799.

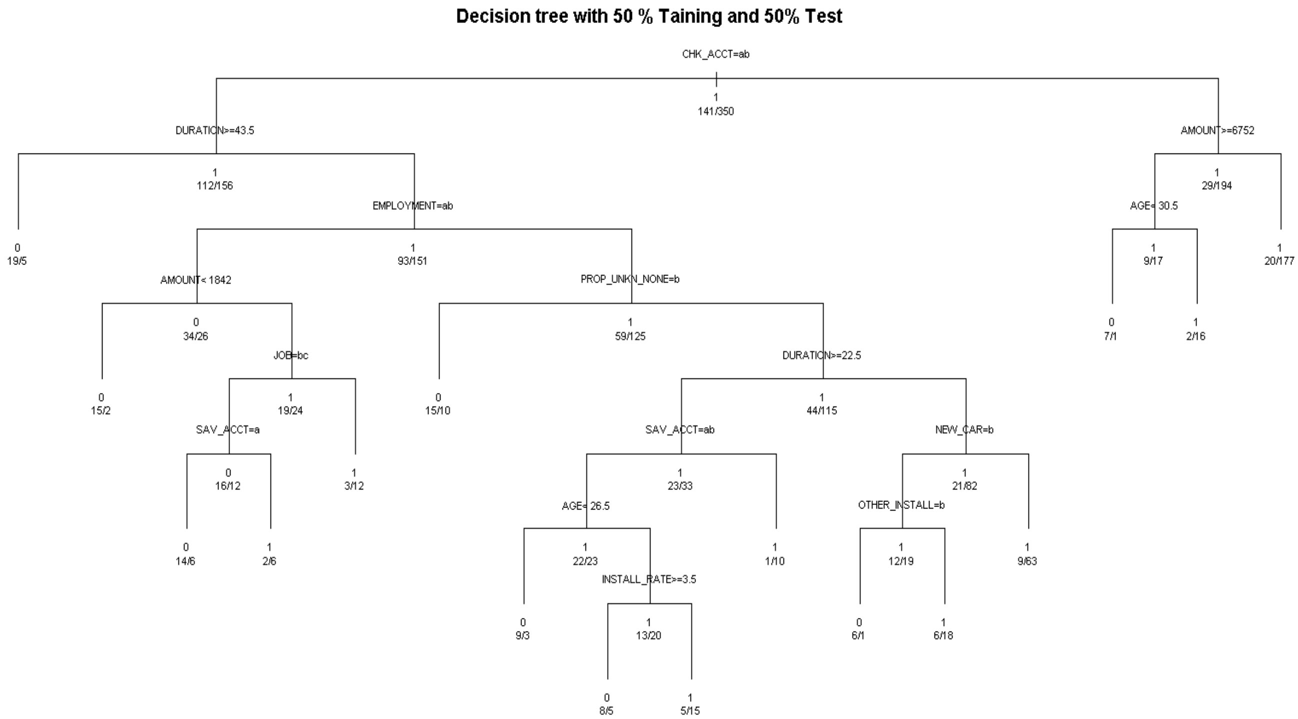
Here the P- Value is very less and also the sensitivity and specificity for the model are ina good stand. Looking at the data, we can conclude that the model is a reliable model.

(c) Building a tree for 50% Test and 50 % Training dataset.

Code:

```
ind = sample(2, nrow(German.Credit), replace = T, prob = c(0.5, 0.5))
GCTrain=German.Credit[ind==1,]
GCTest=German.Credit[ind==2,]
GCTrain4rpart = GCTrain[,2:32]

formula=as.formula(German.Credit$RESPONSE~DURATION+HISTORY+AMOUNT+SAV_A
CCT+USED_CAR+GUARANTOR+NUM_CREDITS+PRESENT_RESIDENT+PROP_UNKN_NONE
+RADIO.TV+EMPLOYMENT+AGE+REAL_ESTATE+JOB)

fit = rpart(RESPONSE ~ ., method="class", data=GCTrain[,2:33])
plot(fit, uniform = T,main ='Decision tree with 50 % Taining and 50% Test')
text(fit, use.n = T, all = T, cex = 0.7, xpd = T)
printcp(fit)
```

Graph:



Output:

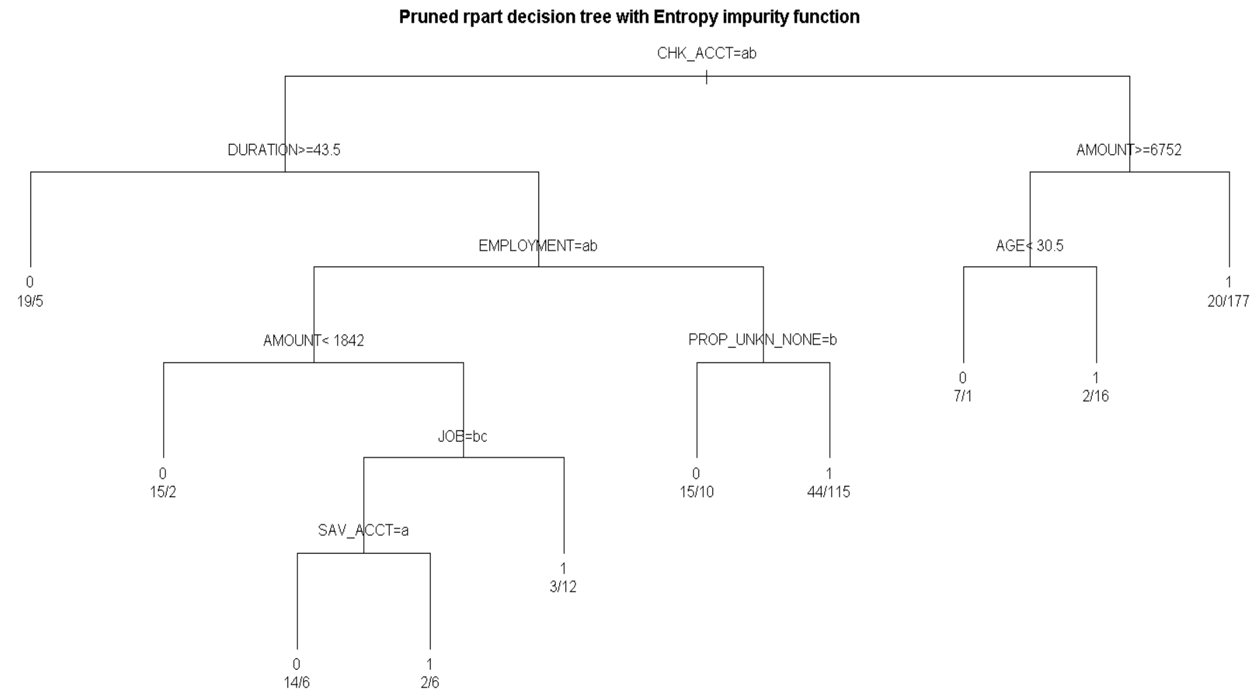
```
printcp(fit)

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE            AMOUNT         CHK_ACCT       DURATION       EMPLOYMENT     INSTALL_R
 [9] OTHER_INSTALL  PROP_UNKN_NONE SAV_ACCT

Root node error: 141/491 = 0.28717

n= 491

        CP nsplit rel error  xerror     xstd
1 0.049645      0   1.00000 1.00000 0.071102
2 0.035461      3   0.84397 1.04965 0.072114
3 0.028369      5   0.77305 1.02837 0.071691
4 0.021277      7   0.71631 1.00709 0.071252
5 0.015603      9   0.67376 0.97163 0.070486
6 0.010000     15   0.57447 0.99291 0.070951
```

After Pruning:

Code:

opt = which.min(fit$cptable[,"xerror"])
cp = fit$cptable[opt, "CP"]
tree.prune = prune(fit, cp = cp)
plot(tree.prune, uniform = T, main = "Pruned rpart decision tree with Entropy impurity fu nction")
text(tree.prune, use.n = T, xpd = T)

Plot:



Output:

```
printcp(fit)

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE            AMOUNT         CHK_ACCT       DURATION       EMPLOYMENT
INSTALL_RATE    JOB            NEW_CAR
 [9] OTHER_INSTALL  PROP_UNKN_NONE SAV_ACCT

Root node error: 141/491 = 0.28717

n= 491

        CP nsplit rel error  xerror     xstd
1 0.049645      0   1.00000 1.00000 0.071102
2 0.035461      3   0.84397 1.04965 0.072114
3 0.028369      5   0.77305 1.02837 0.071691
4 0.021277      7   0.71631 1.00709 0.071252
5 0.015603      9   0.67376 0.97163 0.070486
```

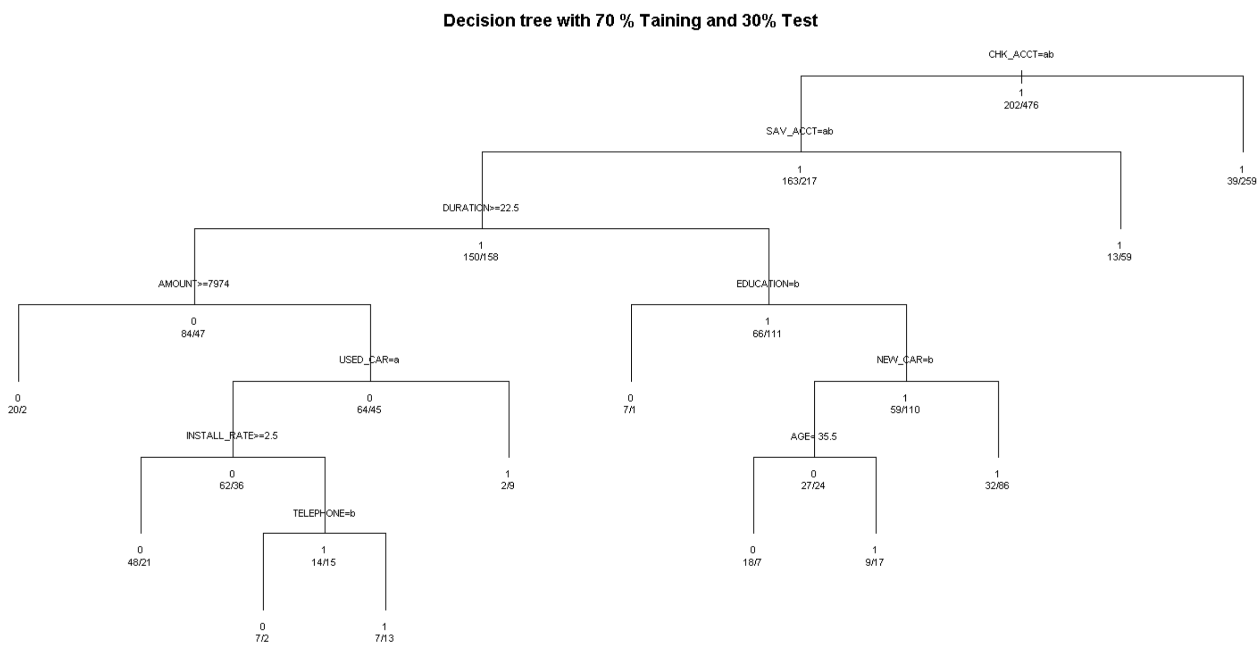
```
6 0.010000      15    0.57447 0.99291 0.070951
```

With Training= 70 % and Testing = 30 %

Code:

```
# Training = 70% testing=30%
ind = sample(2, nrow(German.Credit), replace = T, prob = c(0.7, 0.3))
GCTrain=German.Credit[ind==1,]
GCTest=German.Credit[ind==2,]
fit = rpart(RESPONSE ~ ., method="class", data = GCTrain[,2:33])
plot(fit, uniform = T,main ='Decision tree with 70 % Taining and 30% Test')
text(fit, use.n = T, all = T, cex = 0.7, xpd = T)
printcp(fit)
```

Plot:



Output:

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE AMOUNT CHK_ACCT DURATION EDUCATION INSTALL_RATE NEW_CAR SAV_ACCT TELEPHONE
[10] USED_CAR

Root node error: 202/678 = 0.29794

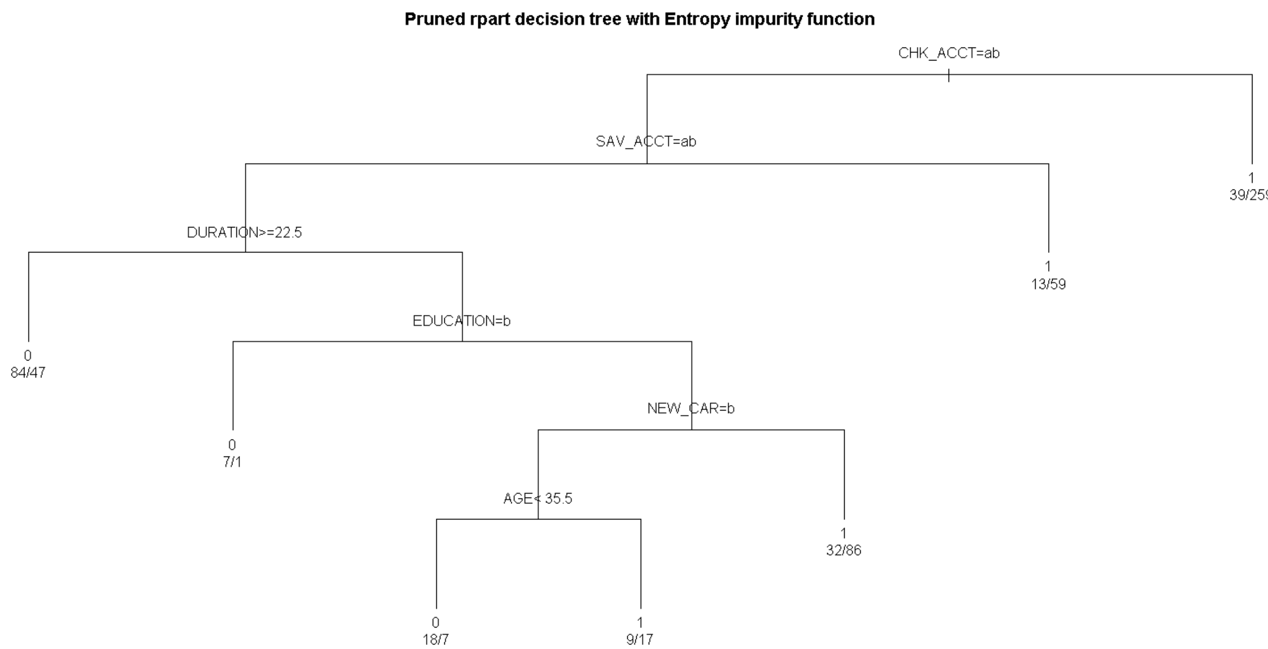
```
        n= 678

CP nsplit      rel  error   xerror     xstd
1 0.061056      0   1.00000 1.00000 0.058954
2 0.029703      3   0.81683 0.91089 0.057320
3 0.027228      4   0.78713 0.88614 0.056821
4 0.017327      6   0.73267 0.88119 0.056719
5 0.014851      8   0.69802 0.91584 0.057417
6 0.010000     10   0.66832 0.93564 0.057799
```

Pruning:

Code:

```
#Pruning

    opt = which.min(fit$cptable[,"xerror"])
    cp = fit$cptable[opt, "CP"]
    tree.prune = prune(fit, cp = cp)
    plot(tree.prune, uniform = T, main = "Pruned rpart decision tree
    with Entropy impurity function")
    text(tree.prune, use.n = T, xpd = T )
    printcp(fit)
```

Plot:



Results:

```
printcp(fit)

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")
```

```
Variables actually used in tree construction:
 [1] AGE          AMOUNT       CHK_ACCT     DURATION     EDUCATION
INSTALL_RATE NEW_CAR      SAV_ACCT     TELEPHONE
[10] USED_CAR

Root node error: 202/678 = 0.29794

n= 678

        CP nsplit rel error  xerror     xstd
1 0.061056      0   1.00000 1.00000 0.058954
2 0.029703      3   0.81683 0.91089 0.057320
3 0.027228      4   0.78713 0.88614 0.056821
4 0.017327      6   0.73267 0.88119 0.056719
5 0.014851      8   0.69802 0.91584 0.057417
6 0.010000     10   0.66832 0.93564 0.057799
```

```
printcp(fit)
```

```
Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE          AMOUNT       CHK_ACCT     DURATION     EDUCATION    INSTALL_RATE NEW_CA
[10] USED_CAR

Root node error: 202/678 = 0.29794

n= 678

        CP nsplit rel error  xerror     xstd
1 0.061056      0   1.00000 1.00000 0.058954
2 0.029703      3   0.81683 0.91089 0.057320
3 0.027228      4   0.78713 0.88614 0.056821
4 0.017327      6   0.73267 0.88119 0.056719
5 0.014851      8   0.69802 0.91584 0.057417
6 0.010000     10   0.66832 0.93564 0.057799
```

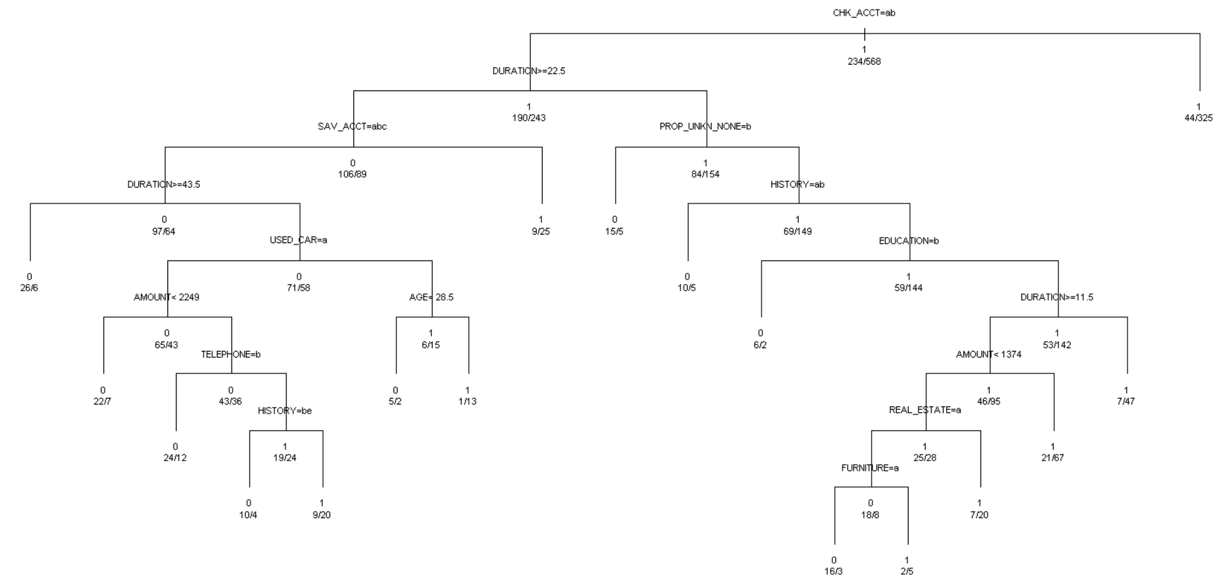
80 % training and 20 % test:

Code:

```
# Training = 80% testing=20%
ind = sample(2, nrow(German.Credit), replace = T, prob = c(0.8, 0.2))
GCTrain=German.Credit[ind==1,]
GCTest=German.Credit[ind==2,]
fit = rpart(RESPONSE ~ ., method="class",data=GCTrain[,2:33])
plot(fit, uniform = T,main ='Decision tree with 80 % Taining and 20% Test')
text(fit, use.n = T, all = T, cex = 0.7, xpd = T)
printcp(fit)
```

Plot:

Decision tree with 80 % Taining and 20% Test



Output:

```
printcp(fit)

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE            AMOUNT         CHK_ACCT       DURATION       EDUCATION
FURNITURE      HISTORY        PROP_UNKN_NONE
 [9] REAL_ESTATE    SAV_ACCT       TELEPHONE      USED_CAR

Root node error: 234/802 = 0.29177

n= 802

        CP nsplit rel error  xerror     xstd
1 0.036325      0   1.00000 1.00000 0.055015
2 0.021368      4   0.81624 0.89316 0.053125
3 0.019231      5   0.79487 0.92735 0.053766
4 0.017094      7   0.75641 0.91880 0.053609
5 0.015670      8   0.73932 0.91880 0.053609
6 0.014245     11   0.69231 0.91026 0.053450
7 0.012821     14   0.64957 0.93590 0.053920
8 0.010000     16   0.62393 0.95299 0.054224
```

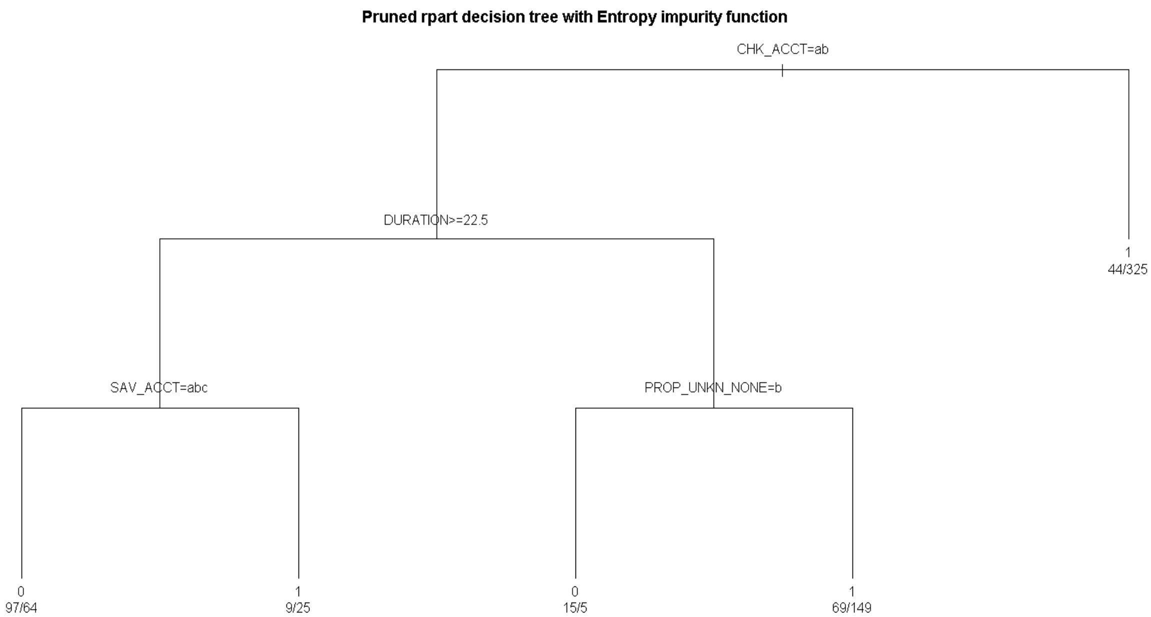
Pruning:

Code:

#Pruning

opt = which.min(fit$cptable[,"xerror"])
cp = fit$cptable[opt, "CP"]
tree.prune = prune(fit, cp = cp)
plot(tree.prune, uniform = T, main = "Pruned rpart decision tree with Entropy impurity function")
text(tree.prune, use.n = T, xpd = T)
printcp(fit)

Plot:



Output:

```
printcp(fit)

Classification tree:
rpart(formula = RESPONSE ~ ., data = GCTrain[, 2:33], method = "class")

Variables actually used in tree construction:
 [1] AGE            AMOUNT         CHK_ACCT       DURATION       EDUCATION
 FURNITURE      HISTORY        PROP_UNKN_NONE
 [9] REAL_ESTATE    SAV_ACCT       TELEPHONE      USED_CAR

Root node error: 234/802 = 0.29177

n= 802
```

```
         CP nsplit rel error  xerror     xstd
1 0.036325      0   1.00000 1.00000 0.055015
2 0.021368      4   0.81624 0.89316 0.053125
3 0.019231      5   0.79487 0.92735 0.053766
4 0.017094      7   0.75641 0.91880 0.053609
5 0.015670      8   0.73932 0.91880 0.053609
6 0.014245     11   0.69231 0.91026 0.053450
7 0.012821     14   0.64957 0.93590 0.053920
8 0.010000     16   0.62393 0.95299 0.054224
```

We can observe that as we increase the percentage of data for testing, the decision tree becomes more and more compact and the amount of root node error decreases. This shows that a better model can be constructed with more data for training.

(d)
As explained in the given question it is not always conceivable that the data will always have values or will be carrying reliable data. In such cases we have to look at the abnormal data and have to take a judgement call on how to handle the data.

For Missing data, we can use Multiple Imputation Techniques to plug the data with respective values. These values can be the mean of the existing values or any different values based on the requirements of the model and the way the variables are defined.

Unreliable data may lead us to false conclusions which are not true for the model. For the unreliable data, we can follow best practices in cleaning the data and using only the cleaned data for the analysis purposes. Ideally in a real time project 60 % of time goes in to cleaning and getting the appropriate data.

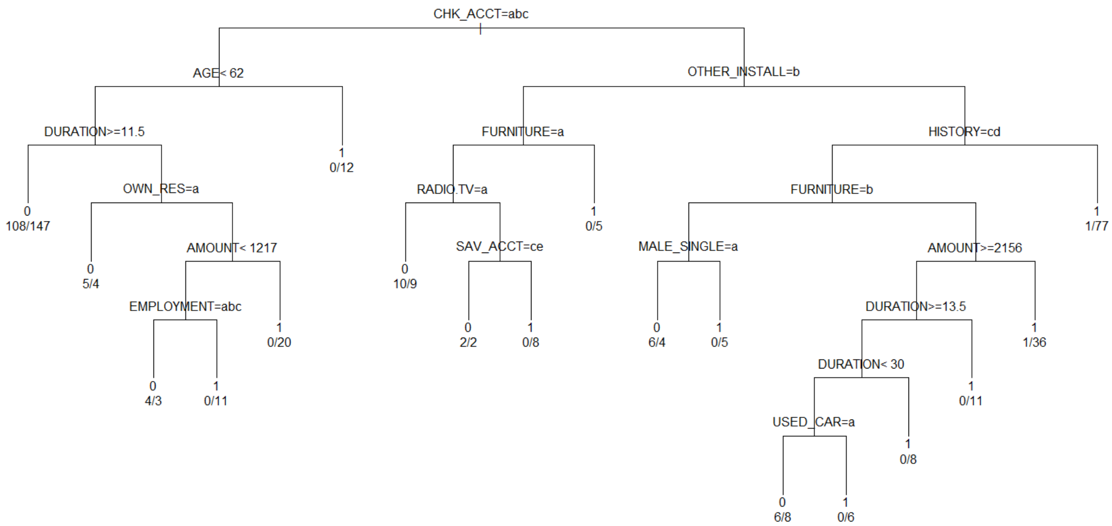
(e) Using misclassification cost in obtaining a model

- 50%-50% for training and test

  Code:

```
Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.5, 0.5))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10),
parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)
```

Plot:



Code for test data:

table(predict(LC, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"))

Output:

```
            Actual Values
Predictions   0   1
          0 134 166
          1  23 158
```

Accuracy = 292/ 481 = 60%

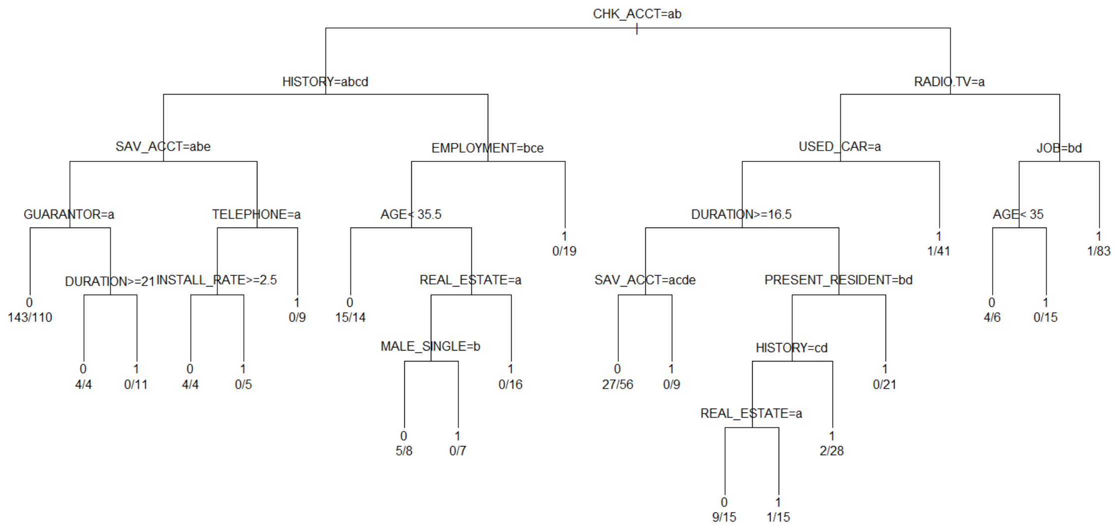
Error = 40%

- 70% for Train and 30% for test

Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.7, 0.3))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10), parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)

Plot:



Code for test data:

table(predict(LC, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"))

Output:

```
            Actual Values
Predictions   0   1
          0  64 109
          1  20  95
```

Accuracy = 159/288 = 55%

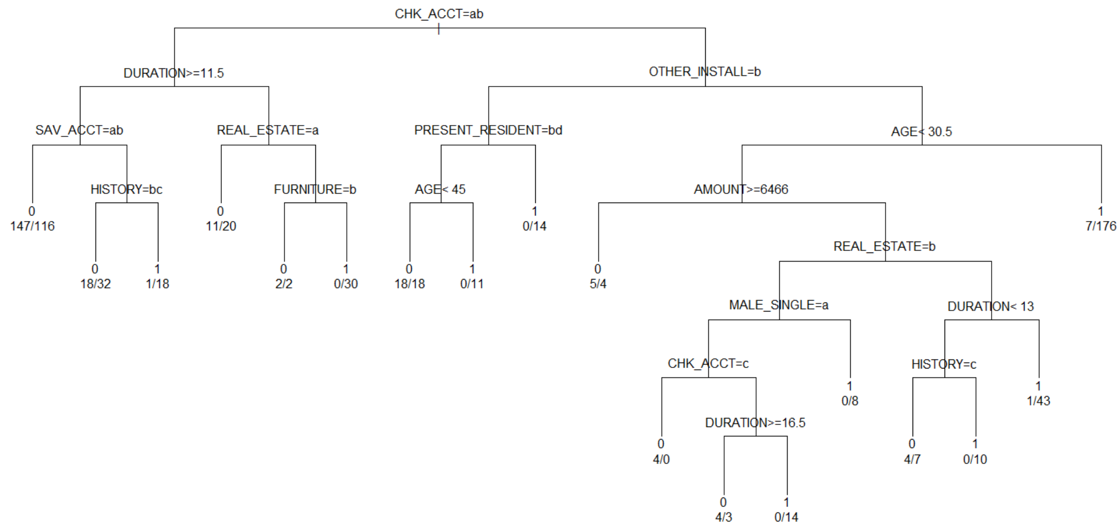
Error = 45%

- 80% for Train and 20% for test

Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.8, 0.3))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10), parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)

Plot:



Code for test data:

table(predict(LC, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"))

Output:

```
            Actual Values
Predictions  0  1
          0 62 83
          1 16 91
```

Accuracy = 153/252 = 60.7%

Error = 39.3%

Preferred model without using the misclassification

	Accuracy
50/50	**72**
70/30	70.3
80/20	70.9

Preferred model with using the misclassification

	Accuracy
50/50	60
70/30	55
80/20	**60.7**

The 50-50 model without misclassification is preferred as the accuracy of it is the highest. On the other hand, for the case with misclassification, the 80-20 model is preferred. The table above summarizes the different accuracy levels.

Benefits of misclassification costs:

It applies weights to the model to specific outcomes hence takes into account the opportunity cost. Thus it can prevent costly mistakes.

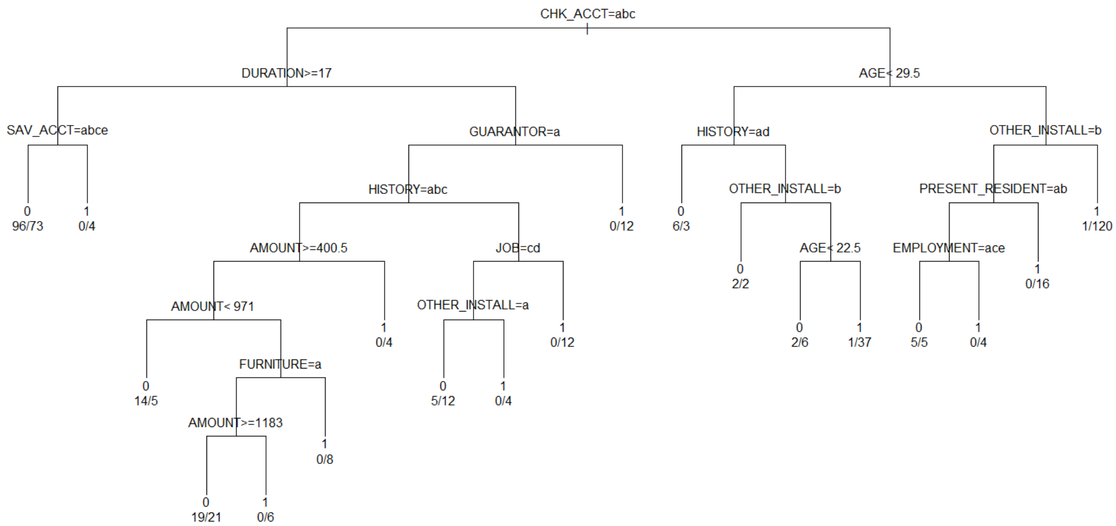
(f) Pruning the trees -

- 50-50% data

```
Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.5, 0.5))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10),
parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)
```

```
opt = which.min(LC$cptable[,"xerror"])
cp = LC$cptable[opt, "CP"]
LC_prune = prune(LC, cp = cp)
plot(LC_prune, uniform=TRUE)
text(LC_prune, use.n=T, xpd=T)
```

Plot:



confusionMatrix(predict(LC_prune, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"), positive = "1")

```
Confusion Matrix and Statistics

           Actual Values
Predictions   0   1
          0 116 165
          1  33 181

               Accuracy : 0.6
                 95% CI : (0.5553, 0.6435)
    No Information Rate : 0.699
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2409
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5231
            Specificity : 0.7785
         Pos Pred Value : 0.8458
         Neg Pred Value : 0.4128
             Prevalence : 0.6990
         Detection Rate : 0.3657
   Detection Prevalence : 0.4323
      Balanced Accuracy : 0.6508

       'Positive' Class : 1
```

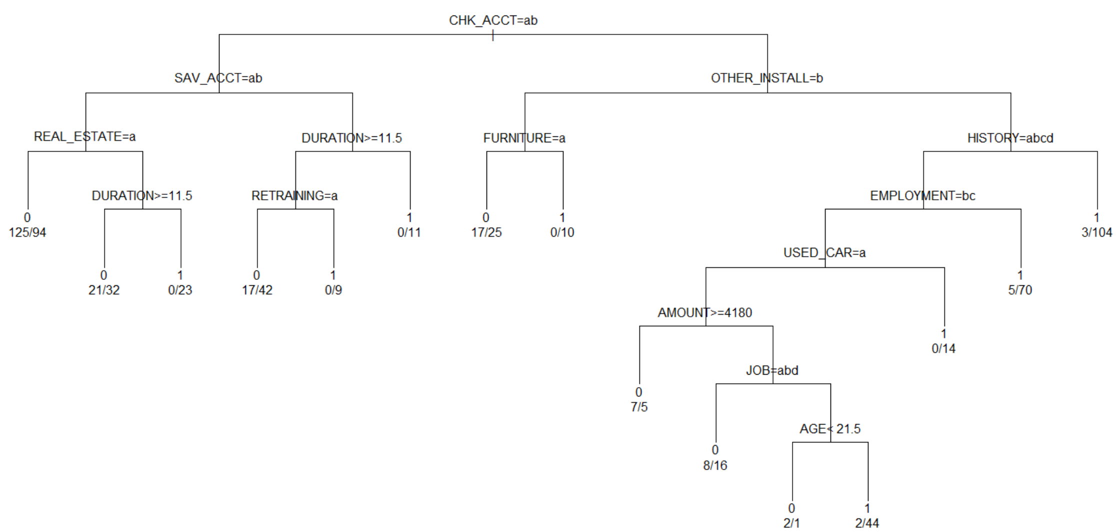
Accuracy = 60%

- 70-30% data

```
Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.7, 0.3))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10),
parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)
```

```
opt = which.min(LC$cptable[,"xerror"])
cp = LC$cptable[opt, "CP"]
LC_prune = prune(LC, cp = cp)
plot(LC_prune, uniform=TRUE)
text(LC_prune, use.n=T, xpd=T)
```

Plot:



confusionMatrix(predict(LC_prune, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"), positive = "1")

```
Confusion Matrix and Statistics

           Actual Values
Predictions   0   1
          0  79 105
          1  14  95
```

```
                Accuracy : 0.5939
                  95% CI : (0.5352, 0.6506)
     No Information Rate : 0.6826
     P-Value [Acc > NIR] : 0.9994

                   Kappa : 0.2572
  Mcnemar's Test P-Value : <2e-16

             Sensitivity : 0.4750
             Specificity : 0.8495
          Pos Pred Value : 0.8716
          Neg Pred Value : 0.4293
              Prevalence : 0.6826
          Detection Rate : 0.3242
    Detection Prevalence : 0.3720
       Balanced Accuracy : 0.6622

        'Positive' Class : 1
```

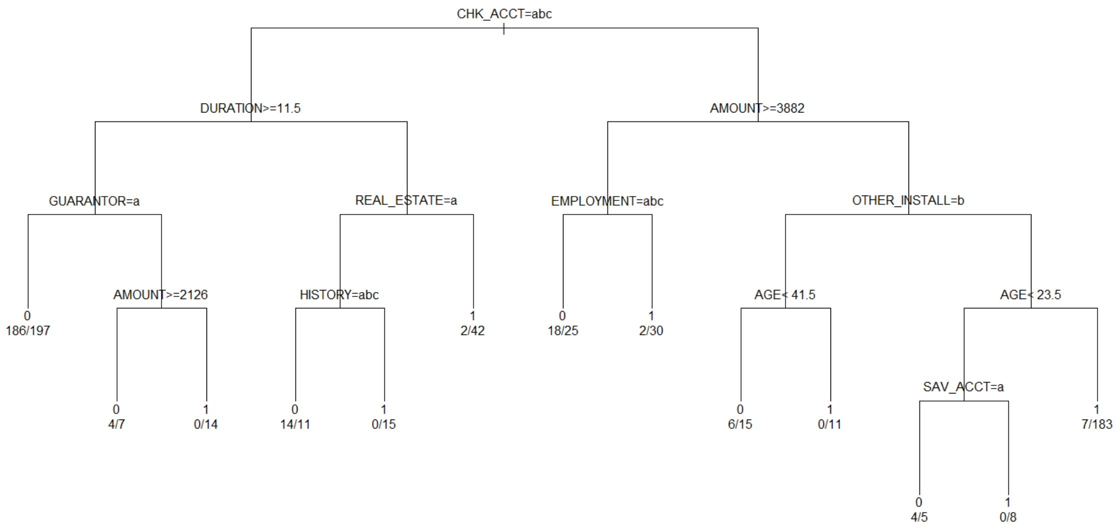
Accuracy = 59%

- 80-20 Train-Test data

```
Dat1 = sample(2, nrow(German.Credit), replace=TRUE, prob=c(0.8, 0.2))
trainData = German.Credit[Dat1==1,]
testData = German.Credit[Dat1==2,]

L <- matrix(c(0, 500, 100, 0), byrow=TRUE, ncol=2)
LC = rpart(RESPONSE~.,data=trainData[,-1], control = rpart.control(minsplit = 10),
parms=list(split="gini", loss=L), method = "class")
plot(LC, uniform=TRUE)
text(LC, use.n=T, xpd=T)
```

```
opt = which.min(LC$cptable[,"xerror"])
cp = LC$cptable[opt, "CP"]
LC_prune = prune(LC, cp = cp)
plot(LC_prune, uniform=TRUE)
text(LC_prune, use.n=T, xpd=T)
```

Plot:



confusionMatrix(predict(LC_prune, testData, type="class"), testData$RESPONSE, dnn=c("Predictions", "Actual Values"), positive = "1")

```
Confusion Matrix and Statistics

              Actual Values
Predictions  0  1
          0 47 67
          1 10 70

               Accuracy : 0.6031
                 95% CI : (0.5305, 0.6725)
    No Information Rate : 0.7062
    P-Value [Acc > NIR] : 0.9992

                  Kappa : 0.2597
 Mcnemar's Test P-Value : 1.75e-10

            Sensitivity : 0.5109
            Specificity : 0.8246
         Pos Pred Value : 0.8750
         Neg Pred Value : 0.4123
             Prevalence : 0.7062
         Detection Rate : 0.3608
   Detection Prevalence : 0.4124
      Balanced Accuracy : 0.6678

       'Positive' Class : 1
```

Before Pruning

	Accuracy
50/50	60
70/30	55
80/20	**60**

After Pruning

	Accuracy
50/50	60
70/30	59.9
80/20	**60.3**

As can be seen above, pruning has increased the accuracy of the model.

(g) Best nodes for classifying “good” applicants

- CHK_ACCT=012 and DURATION>=22 and SAV_ACCT = 0 then we get an optimal tree which performs better for 50% of training data.
- CHK_ACCT=01 and DURATION>+= 22 and SAV_ACCT = 0 and HISTORY = 01 is optimal for 70% of training data and 30% of testing data.

(h) Summary -

The best model is the one with 80-20 classification of train and test data without misclassification. Additionally, the accuracy of the model is higher without misclassification.

Pruning makes a significant difference on a tree. It increases the accuracy of the model.