

Projektdokumentation

Klassifikationsverfahren im Data Mining (Sommersemester 2018)

Arish Saeed

Michael Theis

Maximilian Kurth

19.04.2018

Inhaltsverzeichnis

1	Business Understanding	1
2	Data Understanding	1
3	Data Preparation	11
4	Modeling	25
5	Evaluation	28
6	Deployment	30
7	Ergebnisse Cart: Accuracy = 0.9354, AUC = 0.65	30
8	Ergebnisse rf: Accuracy = 0.9520 , AUC = 0.94	30

1 Business Understanding

Eine Bank möchte vorhersehen können, ob ein Kunde eine Kreditrate mehr als 90 Tage nicht begleicht. Dafür möchten sie eine Data Mining Lösung entwickelt haben. Für die Entwicklung des Modells bekamen wir 120.000 Datensätze mit jeweils 11 Attributen gestellt, deren Bedeutung uns anhand eines Data-Dictionarys erklärt wurden. Die Bank möchte jetzt das Attribut DefaultLast2Years anhand der restlichen Attribute vorhersagen.

2 Data Understanding

-Einlesen der Daten und überprüfen, ob korrekt eingelesen wurde.

```
#dat = read.csv("C:/Users/Lenovo/Downloads/trainset.csv", sep = ",", dec = ".", row.names = "X")
dat=read.csv("C:\\\\Users\\\\ARISH\\\\Desktop\\\\Klassifikation im Data Mining\\\\Projekt\\\\trainset.csv",sep=",",
str(dat)

## 'data.frame': 120000 obs. of 11 variables:
## $ DefaultLast2Years      : int 0 0 0 0 0 0 0 0 0 ...
## $ ProportionOfUnsecuredLines: num 0.665 0.253 0.882 0.018 1 ...
## $ Age                     : int 44 26 46 52 31 63 42 54 73 28 ...
## $ NumberOfTime30.59DaysPast : int 0 0 0 0 0 0 1 0 0 0 ...
## $ DebtRatio                : num 0.42 0.117 0.538 0.721 0 ...
## $ MonthlyIncome             : int 2000 4639 6741 8667 0 3883 1 8375 10097 0 ...
## $ OpenCreditLinesAndLoans   : int 2 9 20 16 0 9 6 6 21 0 ...
## $ X90DaysLate               : int 0 0 0 0 0 0 0 0 0 4 ...
## $ RealEstateLoansOrLines    : int 1 0 1 3 0 0 2 1 4 0 ...
## $ X60.89DaysPast             : int 0 0 0 0 0 0 0 0 0 0 ...
## $ Dependents                 : int 3 0 0 0 0 0 3 0 0 0 ...

summary(dat)

## DefaultLast2Years ProportionOfUnsecuredLines      Age
## Min. :0.00000   Min.   : 0.00          Min.   : 0.0
## 1st Qu.:0.00000  1st Qu.: 0.03          1st Qu.: 41.0
## Median :0.00000  Median : 0.15          Median : 52.0
## Mean   :0.06695  Mean   : 6.30          Mean   : 52.3
## 3rd Qu.:0.00000  3rd Qu.: 0.56          3rd Qu.: 63.0
## Max.  :1.00000   Max.   :50708.00        Max.   :109.0
##
## NumberOfTime30.59DaysPast  DebtRatio      MonthlyIncome
## Min.   : 0.0000  Min.   : 0.0  Min.   :     0
## 1st Qu.: 0.0000  1st Qu.: 0.2  1st Qu.: 3400
## Median : 0.0000  Median : 0.4  Median : 5400
## Mean   : 0.4286  Mean   : 355.3 Mean   : 6669
## 3rd Qu.: 0.0000  3rd Qu.: 0.9  3rd Qu.: 8250
## Max.   :98.0000  Max.   :329664.0 Max.   :3008750
## NA's   :23792
##
## OpenCreditLinesAndLoans  X90DaysLate  RealEstateLoansOrLines
## Min.   : 0.000  Min.   : 0.0000  Min.   : 0.000
## 1st Qu.: 5.000  1st Qu.: 0.0000  1st Qu.: 0.000
## Median : 8.000  Median : 0.0000  Median : 1.000
## Mean   : 8.461  Mean   : 0.2742  Mean   : 1.021
## 3rd Qu.:11.000  3rd Qu.: 0.0000  3rd Qu.: 2.000
## Max.   :58.000  Max.   :98.0000  Max.   :54.000
```

```

## X60.89DaysPast Dependents
## Min. : 0.000 Min. : 0.000
## 1st Qu.: 0.000 1st Qu.: 0.000
## Median : 0.000 Median : 0.000
## Mean : 0.248 Mean : 0.758
## 3rd Qu.: 0.000 3rd Qu.: 1.000
## Max. :98.000 Max. :13.000
## NA's :3179

head(dat)

## DefaultLast2Years ProportionOfUnsecuredLines Age
## 77315 0 0.66520616 44
## 59212 0 0.25298626 26
## 71951 0 0.88230887 46
## 135632 0 0.01799581 52
## 129226 0 0.99999990 31
## 64958 0 0.61210221 63
## Numberoftime30.59DaysPast DebtRatio MonthlyIncome
## 77315 0 0.4202899 2000
## 59212 0 0.1170259 4639
## 71951 0 0.5375260 6741
## 135632 0 0.7209275 8667
## 129226 0 0.0000000 0
## 64958 0 0.1390319 3883
## OpenCreditLinesAndLoans X90DaysLate RealEstateLoansOrLines
## 77315 2 0 1
## 59212 9 0 0
## 71951 20 0 1
## 135632 16 0 3
## 129226 0 0 0
## 64958 9 0 0
## X60.89DaysPast Dependents
## 77315 0 3
## 59212 0 0
## 71951 0 0
## 135632 0 0
## 129226 0 0
## 64958 0 0

```

- Erste auffälligkeiten: viele NAs bei MonthlyIncome und viele Ausreißer zwischen 3. Quantil und Maximum, extrem hohe Maximalwerte
- Einbinden der benötigten Libraries

```

library(rpart.plot)

## Loading required package: rpart
library(rpart)
library(fifer)

## Warning: package 'fifer' was built under R version 3.4.4
## Loading required package: MASS

```

```

library(rpart)
library(ROCR)

## Warning: package 'ROCR' was built under R version 3.4.4
## Loading required package: gplots
## Warning: package 'gplots' was built under R version 3.4.4
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##      lowess

library(randomForest)

## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
library(pmml)

## Loading required package: XML
library(mice)

## Loading required package: lattice
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##      combine

## The following object is masked from 'package:MASS':
##      select

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

## The following objects are masked from 'package:base':
##      intersect, setdiff, setequal, union
• Ändern der falsch klassifizierten Attribute

dat$DefaultLast2Years = as.factor(dat$DefaultLast2Years) #zu Factor ändern, da ja nein
str(dat)

## 'data.frame': 120000 obs. of 11 variables:
##   $ DefaultLast2Years : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 ...
##   $ ProportionOfUnsecuredLines: num 0.665 0.253 0.882 0.018 1 ...
##   $ Age : int 44 26 46 52 31 63 42 54 73 28 ...
##   $ NumberOfTime30.59DaysPast : int 0 0 0 0 0 1 0 0 0 ...
##   $ DebtRatio : num 0.42 0.117 0.538 0.721 0 ...

```

```

## $ MonthlyIncome : int 2000 4639 6741 8667 0 3883 1 8375 10097 0 ...
## $ OpenCreditLinesAndLoans : int 2 9 20 16 0 9 6 6 21 0 ...
## $ X90DaysLate : int 0 0 0 0 0 0 0 0 0 4 ...
## $ RealEstateLoansOrLines : int 1 0 1 3 0 0 2 1 4 0 ...
## $ X60.89DaysPast : int 0 0 0 0 0 0 0 0 0 0 ...
## $ Dependents : int 3 0 0 0 0 0 3 0 0 0 ...



- Verstehen der einzelnen Attribute und mögliche Änderungen vorschlagen



```
#ProportionOfUnsecuredLines
#Unsichere Kredite/ alle Kredite -> kann größer als 1 sein aber nicht größer 2
nrow(dat[dat$ProportionOfUnsecuredLines>2,]) # = 298 Anomalien
```



```
[1] 298
```



```
#Age
#Sollte nicht größer als 65, da wir Laufzeiten nicht kennen 65 = Rentenalter, keine
#großen Investitionen und nicht jünger als 18, da erst mit 18 Kredite genommen werden können
nrow(dat[dat$Age>65,]) # = 22863
```



```
[1] 22863
```



```
nrow(dat[dat$Age < 18,]) # = 1
```



```
[1] 1
```



```
#NumberOfType30.59DaysPast
#Obergrenze für letzte 2 Jahre -> höchstens 24(12,8) mal da monat = 30 Tage. Analog für die anderen bei
head(sort(dat$NumberOfType30.59DaysPast, decreasing = TRUE), n=10)
```



```
[1] 98 98 98 98 98 98 98 98 98 98
```



```
#DebtRatio
#nicht größer als eins, da sonst Einkommen < Forderungen
head(sort(dat$DebtRatio, decreasing = TRUE), n=10)
```



```
[1] 329664 326442 307001 220516 168835 110952 106885 101320 61907 60212
```



```
nrow(dat[dat$DebtRatio > 1,])
```



```
[1] 28136
```



```
nrow(dat[dat$DebtRatio > 1 & is.na(dat$MonthlyIncome) == TRUE,]) # offensichtlich hoher Zusammenhang zw
```



```
[1] 22335
```



```
nrow(dat[dat$DebtRatio < 1 & is.na(dat$MonthlyIncome) == TRUE,]) # Kreditbetrag berechnen?
```



```
[1] 1289
```



```
#MonthlyIncome
#Oberstes Quantil exorbitant hoch
siehe spiegel 2005 http://www.spiegel.de/wirtschaft/milliarden-microsoft-erhöht-gehalt-von-bill-ga
vorstand deutsche bank 3.000.000 im Jahr = 250.000 im Monat
head(sort(dat$MonthlyIncome, decreasing = TRUE), n=10)
```



```
[1] 3008750 1794060 1072500 835040 730483 702500 699530 649587
```



```
[9] 582369 562466
```



```
#OpenCreditLinesAndLoans zum Zeitpunkt der Einstellung über 11 ungewöhnlich (11 = 3. Quantil, ab da sta
head(sort(dat$OpenCreditLinesAndLoans, decreasing = TRUE), n=10)
```


```

```

## [1] 58 57 57 56 54 54 54 54 53 52
dat[dat$OpenCreditLinesAndLoans == 58,]

##      DefaultLast2Years ProportionOfUnsecuredLines Age
## 30588              0          0.003032193 53
##      NumberOfTime30.59DaysPast DebtRatio MonthlyIncome
## 30588              0  5.967504        8000
##      OpenCreditLinesAndLoans X90DaysLate RealEstateLoansOrLines
## 30588              58          0          54
##      X60.89DaysPast Dependents
## 30588              0          0
#X60 und 90 gleich hoch? 90 > 60 Anomalie, da man mindestens 60 Tage zu spät sein muss um 90 Tage zu sp
nrow(dat[dat$X60.89DaysPast,])

## [1] 6069
nrow(dat[dat$X90DaysLate,])

## [1] 6661
#RealEstateLoansOrLines muss kleiner sein als Opencreditlinesandloans, da echte Teilmenge
head(sort(dat$RealEstateLoansOrLines, decreasing = TRUE), n=10)

## [1] 54 29 25 25 25 23 23 21 20 19
dat[dat$RealEstateLoansOrLines == 54,]

##      DefaultLast2Years ProportionOfUnsecuredLines Age
## 30588              0          0.003032193 53
##      NumberOfTime30.59DaysPast DebtRatio MonthlyIncome
## 30588              0  5.967504        8000
##      OpenCreditLinesAndLoans X90DaysLate RealEstateLoansOrLines
## 30588              58          0          54
##      X60.89DaysPast Dependents
## 30588              0          0
dat[dat$RealEstateLoansOrLines > dat$OpenCreditLinesAndLoans,] #keine anomalien

## [1] DefaultLast2Years      ProportionOfUnsecuredLines
## [3] Age                  NumberOfTime30.59DaysPast
## [5] DebtRatio            MonthlyIncome
## [7] OpenCreditLinesAndLoans X90DaysLate
## [9] RealEstateLoansOrLines X60.89DaysPast
## [11] Dependents
## <0 rows> (or 0-length row.names)

#Dependents
head(sort(dat$Dependents, decreasing = TRUE), n=10) #keine auffälligkeiten

## [1] 13 10 10 10 10  9  9  9  9  8
nrow(dat[dat$Age > 65, ])

## [1] 22863
nrow(dat[dat$NumberOfTime30.59DaysPast == 98,])

## [1] 220

```

```

nrow(dat[dat$X60.89DaysPast == 98,])

## [1] 220
nrow(dat[dat$X90DaysLate == 98,])

## [1] 220
quantile(dat$X90DaysLate) #drittes quantil noch bei 0, viertes schon bei 98???
##    0%   25%   50%   75% 100%
##    0     0     0     0    98

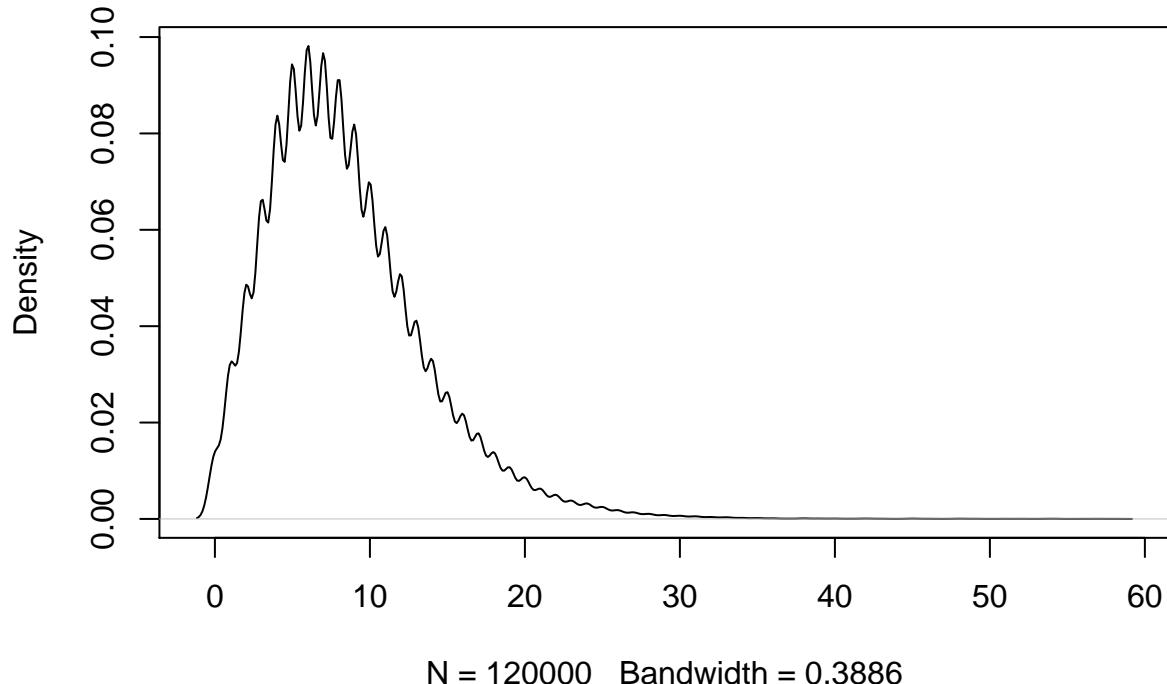
head(sort(dat$X90DaysLate, decreasing = TRUE)) # die ersten 10 nach 220 ausgeben und schauen auch für d
## [1] 98 98 98 98 98 98
nrow(dat[dat$DebtRatio > 1,])

## [1] 28136
nrow(dat[dat$Age>65 & dat$DebtRatio>1,]) # sehr alt und negatives einkommen -> unrealistisch
## [1] 6880

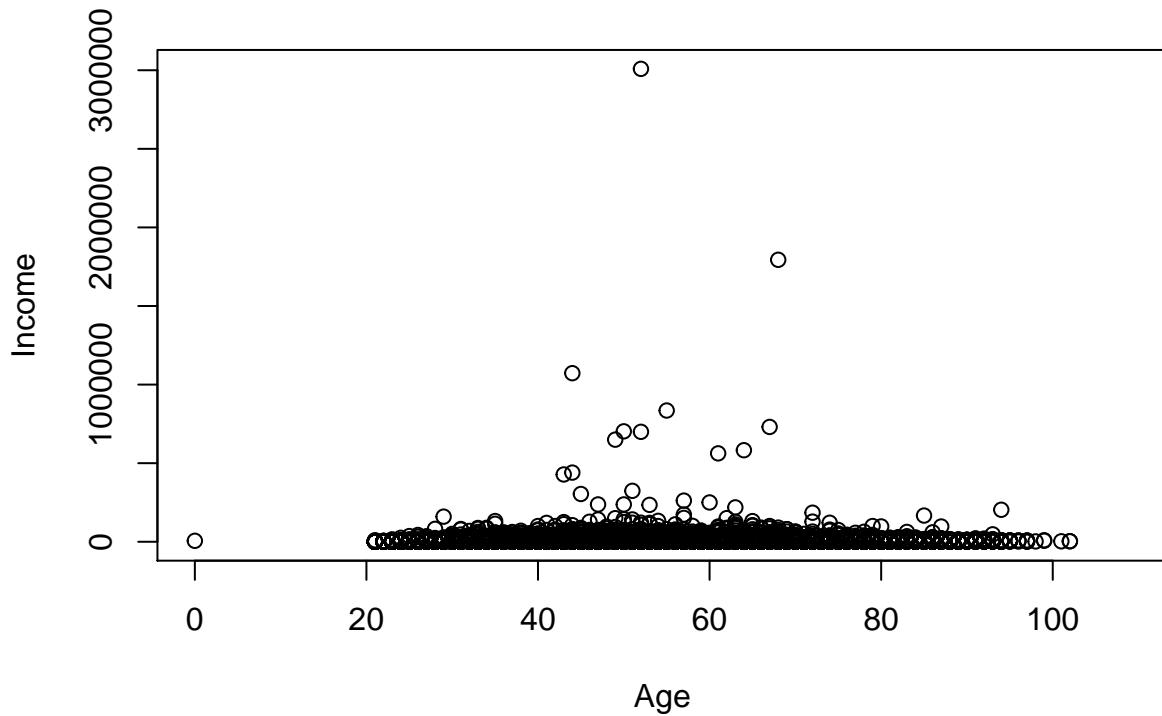
-
#Plots zur Veranschaulichung
plot(density(dat$OpenCreditLinesAndLoans)) #quasi Normalverteilt -> Mittelwert nehmen

```

density.default(x = dat\$OpenCreditLinesAndLoans)

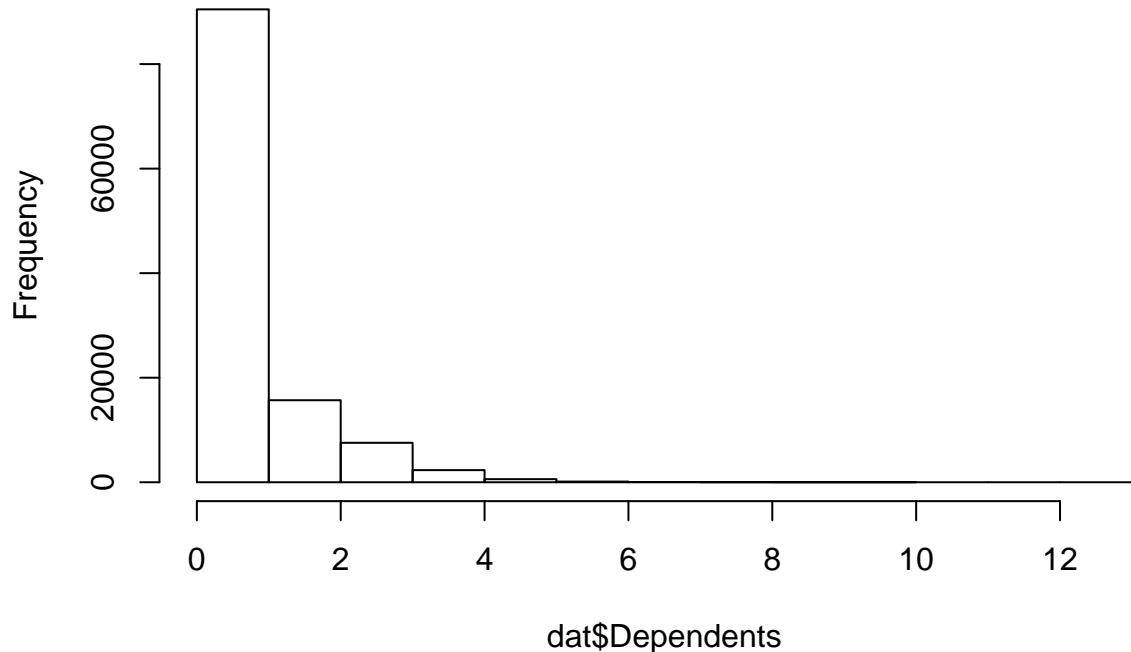


```
plot(dat$Age, dat$MonthlyIncome, xlab = "Age", ylab = "Income") #Nur sehr wenige Ausreißer
```



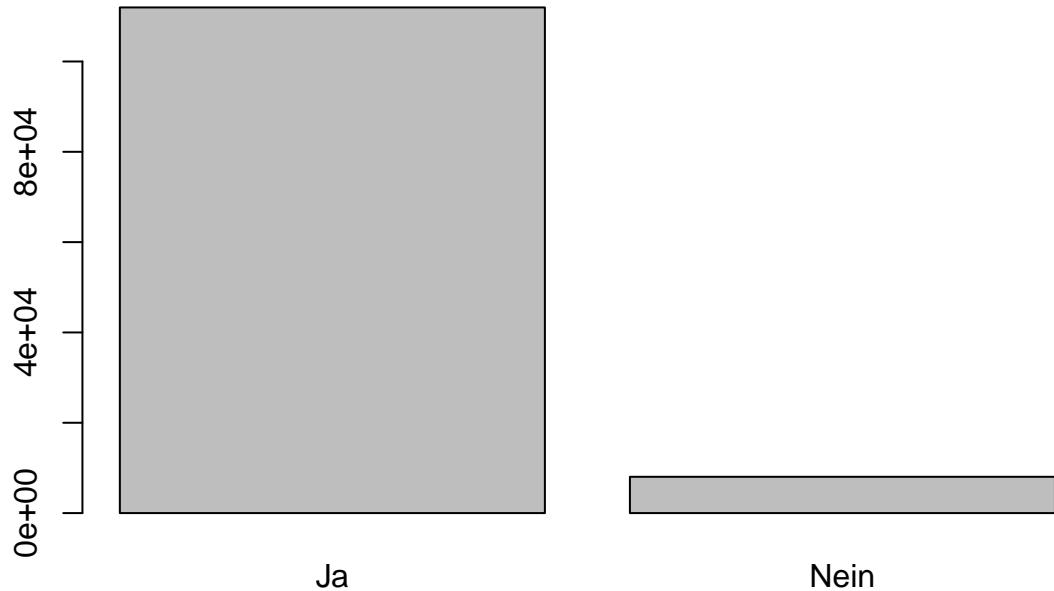
```
hist(dat$Dependents) #wenige Dependents über 6
```

Histogram of dat\$Dependents

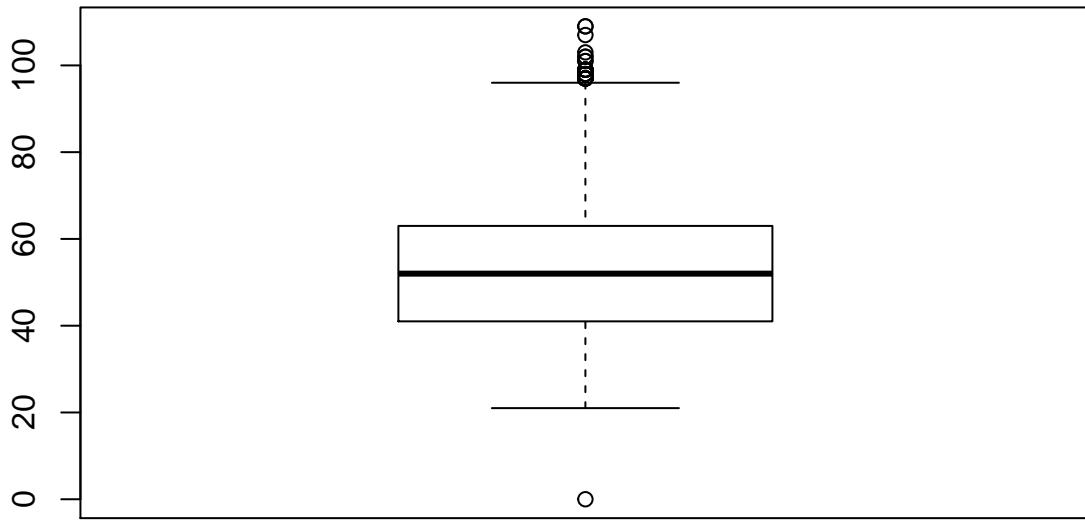


```
counts = table(dat$DefaultLast2Years)
barplot(counts, main = "Bezahlt", names.arg=c("Ja", "Nein")) # Viel mehr Leute haben bezahlt als nicht
```

Bezahlt



```
boxplot(dat$Age) # Nochmal visualisieren, dass die meisten Leute zwischen 40 und 60 sind.
```



Korrelationen suchen

```
#Korrelationen suchen zwecks Regressionsanalyse
cor(dat$MonthlyIncome, dat$NumberOfTime30.59DaysPast, use = "pairwise.complete.obs")

## [1] -0.010377
cor(dat$MonthlyIncome, dat$X60.89DaysPast, use = "pairwise.complete.obs")

## [1] -0.01131145
cor(dat$MonthlyIncome, dat$X90DaysLate, use = "pairwise.complete.obs")

## [1] -0.01292171
cor(dat$X90DaysLate, dat$NumberOfTime30.59DaysPast, use = "pairwise.complete.obs")

## [1] 0.9841773
cor(dat$X60.89DaysPast, dat$X90DaysLate, use = "pairwise.complete.obs")

## [1] 0.9930968
cor(dat$X60.89DaysPast, dat$NumberOfTime30.59DaysPast, use = "pairwise.complete.obs")

## [1] 0.9875823
cor(dat$X60.89DaysPast, dat$RealEstateLoansOrLines, use = "pairwise.complete.obs")

## [1] -0.04057814
```

```

cor(dat$X60.89DaysPast, dat$ProportionOfUnsecuredLines, use = "pairwise.complete.obs")
## [1] -0.001029492
cor(dat$RealEstateLoansOrLines, dat$ProportionOfUnsecuredLines, use = "pairwise.complete.obs")
## [1] 0.007125576
cor(dat$Age, dat$Dependents, use = "pairwise.complete.obs")
## [1] -0.2146264
cor(dat$MonthlyIncome, dat$Dependents, use = "pairwise.complete.obs")
## [1] 0.05991752
#Ergebnis ernüchternd

```

3 Data Preparation

```

#Preparation

#Unsecured auf NA

for(i in 1:nrow(dat)){
  if(dat$ProportionOfUnsecuredLines[i] >2){ #kann nicht > 2 sein
    dat$ProportionOfUnsecuredLines[i] = NA
  }
}

#Age auf NA
for(i in 1:nrow(dat)){
  if(dat$Age[i] >=65 | dat$Age[i] < 18){ #Rentenalter! Laufzeit unbekannt
    dat$Age[i] = NA
  }
}

#Obergrenze für letzte 2 Jahre -> höchstens 24(12,8) mal da monat = 30 Tage. Analog für die anderen bei

for(i in 1:nrow(dat)){
  if(dat$NumberOfTime30.59DaysPast[i] >24){
    dat$NumberOfTime30.59DaysPast[i] = NA
  }
}

for(i in 1:nrow(dat)){
  if(dat$X60.89DaysPast[i] >12){
    dat$X60.89DaysPast[i] = NA
  }
}

for(i in 1:nrow(dat)){
  if(dat$X90DaysLate[i] >=8){
    dat$X90DaysLate[i] = NA
  }
}

```

```

    }

}

#DebtRatio
for(i in 1:nrow(dat)){
  if(dat$DebtRatio[i] > 1){
    dat$DebtRatio[i] = NA
  }
}

#Monthly < 250.000 & monthly > 416 für HartzIV
for(i in 1:nrow(dat)){
  if(!is.na(dat$MonthlyIncome[i]) & dat$MonthlyIncome[i] > 250000){
    dat$MonthlyIncome[i] = NA
  }
}

for(i in 1:nrow(dat)){
  if(!is.na(dat$MonthlyIncome[i]) & dat$MonthlyIncome[i] < 416){
    dat$MonthlyIncome[i] = NA
  }
}

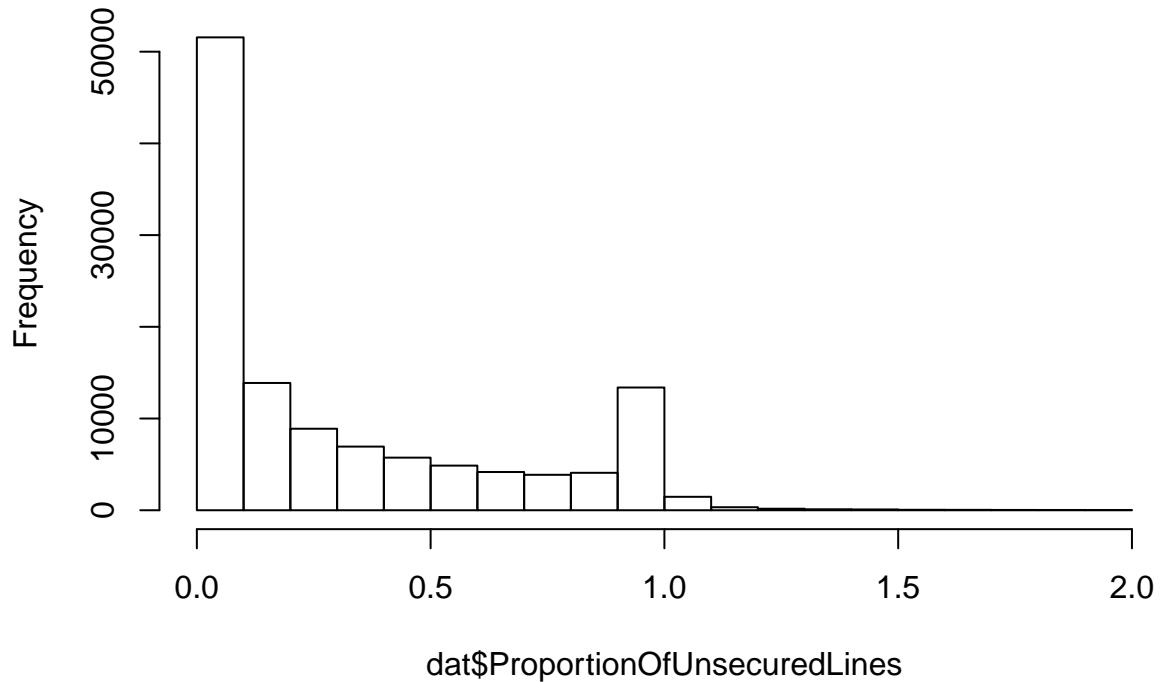
for(i in 1:nrow(dat)){
  if(!is.na(dat$OpenCreditLinesAndLoans[i]) & dat$OpenCreditLinesAndLoans[i] > 11){
    dat$OpenCreditLinesAndLoans[i] = NA
  }
}

#Setzen der NAs auf Mittelwert/Median, je nachdem ob Histogramm Normalverteilt ist

hist(dat$ProportionOfUnsecuredLines)

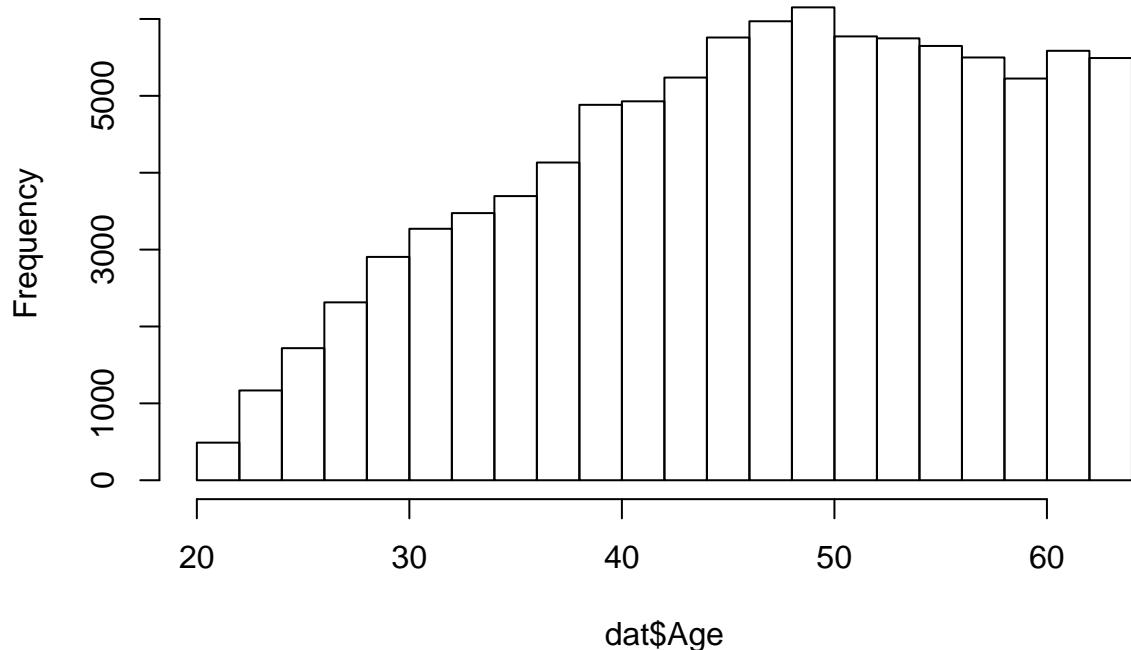
```

Histogram of dat\$ProportionOfUnsecuredLines



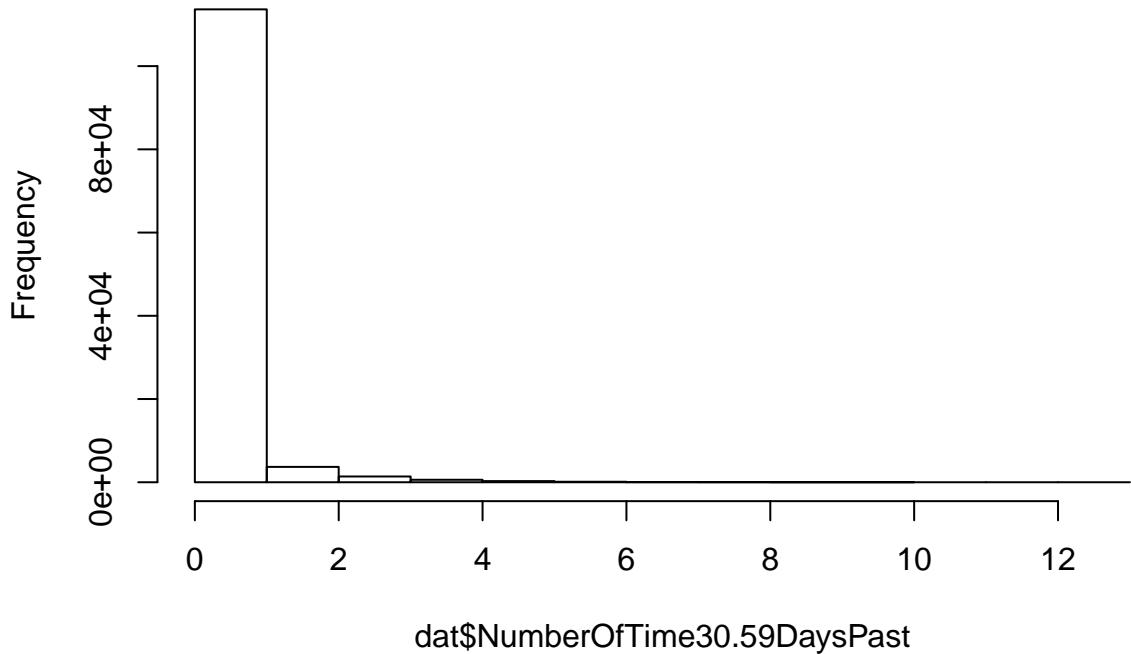
```
hist(dat$Age)
```

Histogram of dat\$Age



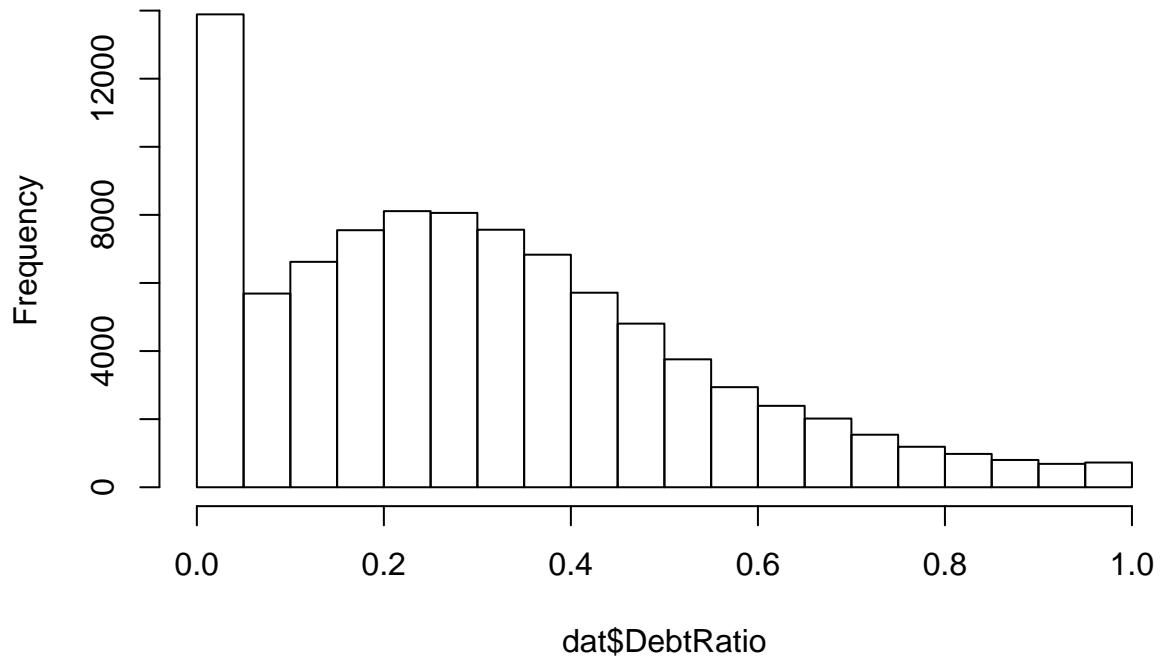
```
hist(dat$Number0fTime30.59DaysPast)
```

Histogram of dat\$NumberOfTime30.59DaysPast



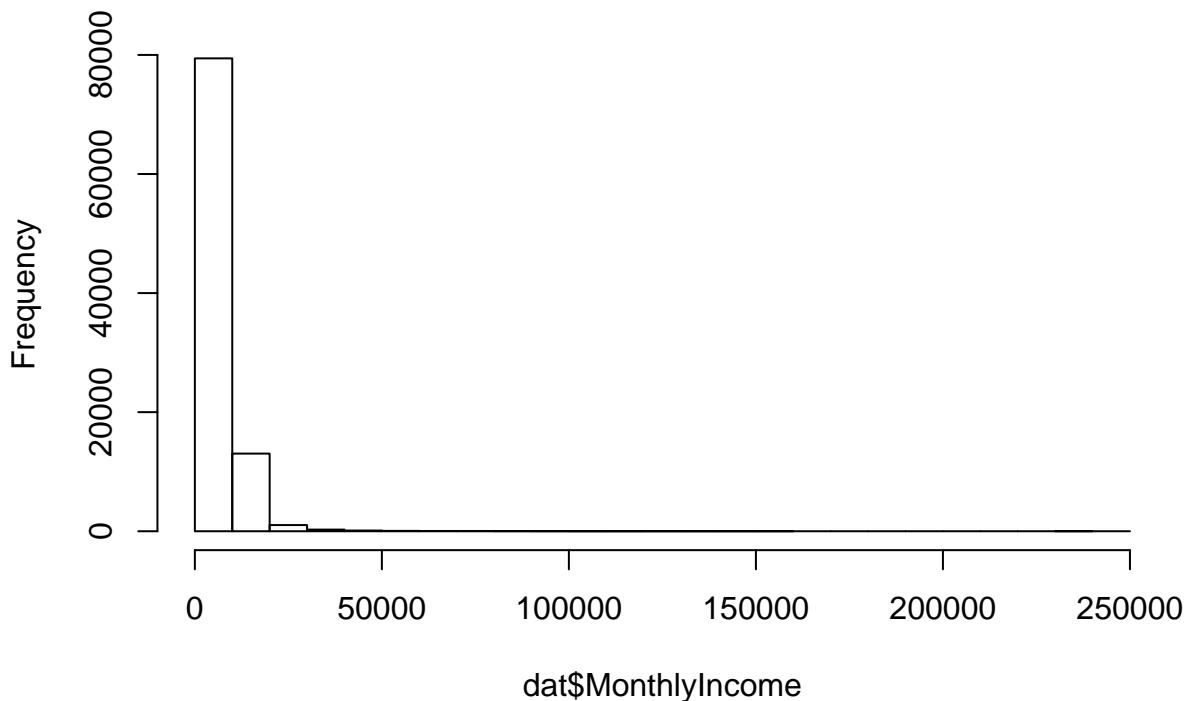
```
hist(dat$DebtRatio)
```

Histogram of dat\$DebtRatio



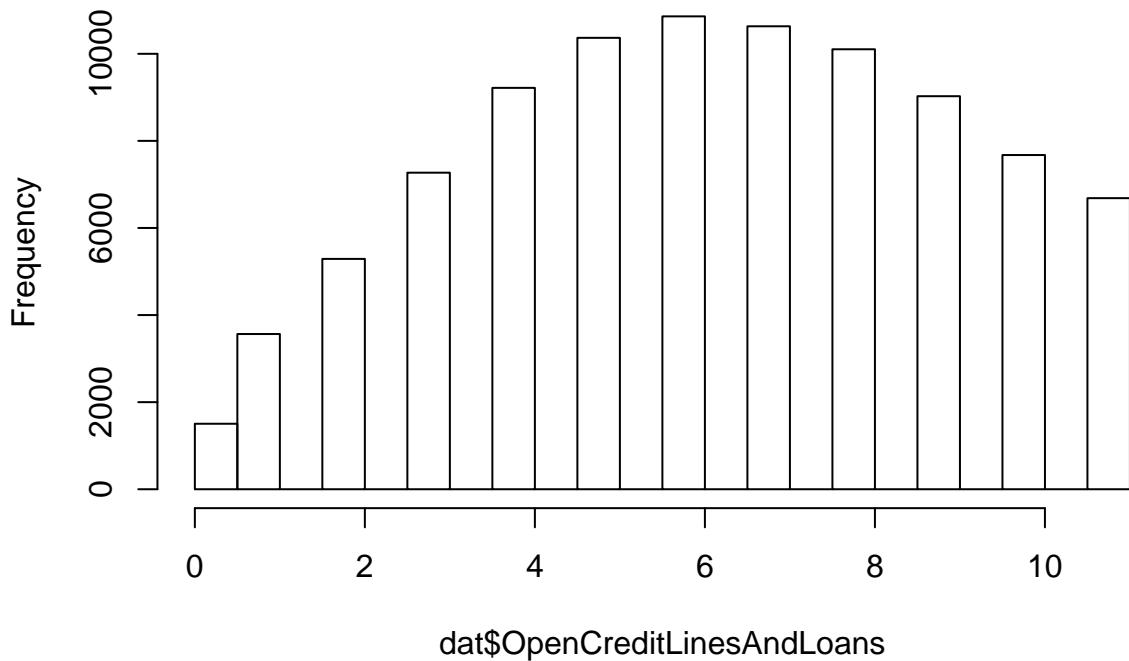
```
hist(dat$MonthlyIncome)
```

Histogram of dat\$MonthlyIncome



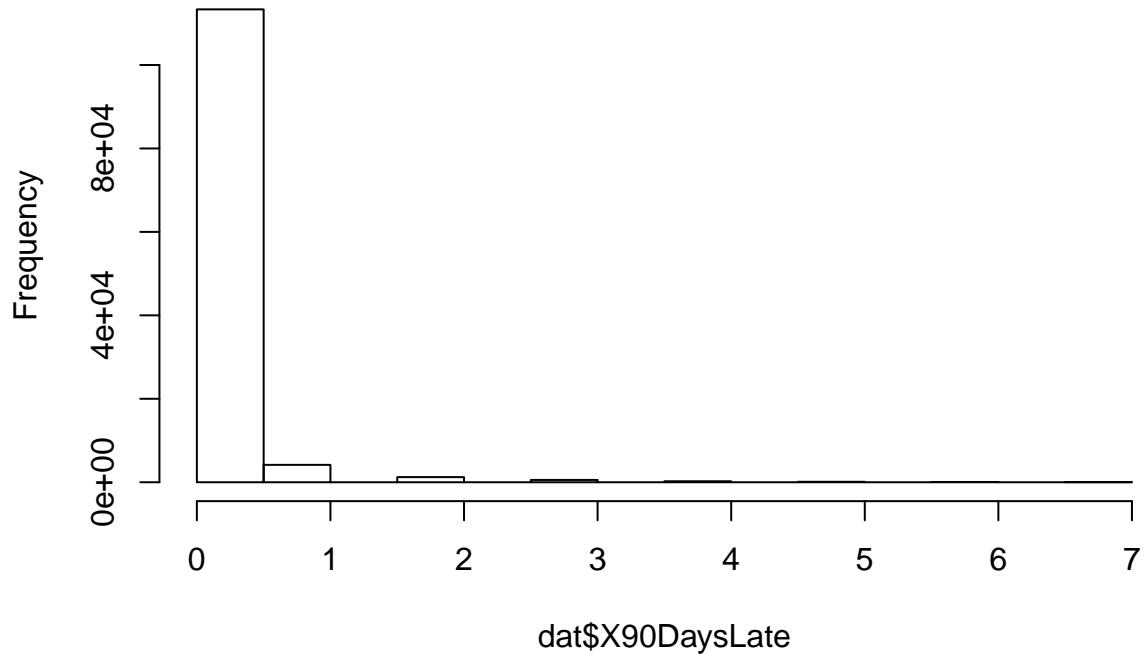
```
hist(dat$OpenCreditLinesAndLoans)
```

Histogram of dat\$OpenCreditLinesAndLoans



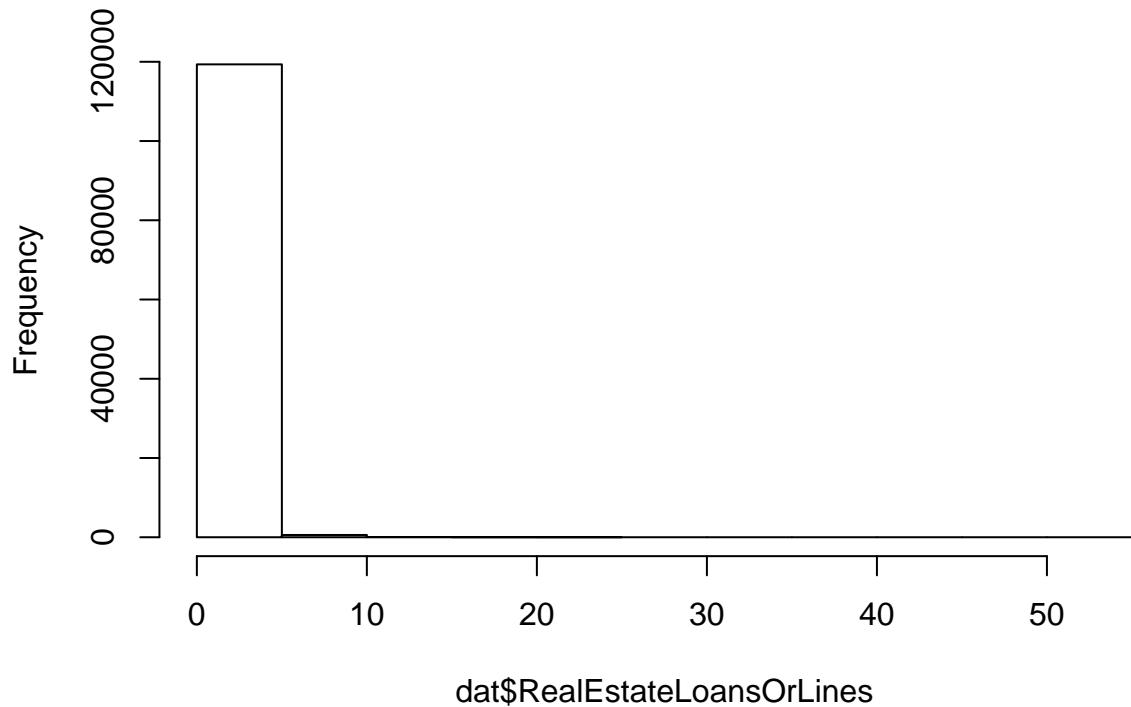
```
hist(dat$X90DaysLate)
```

Histogram of dat\$X90DaysLate



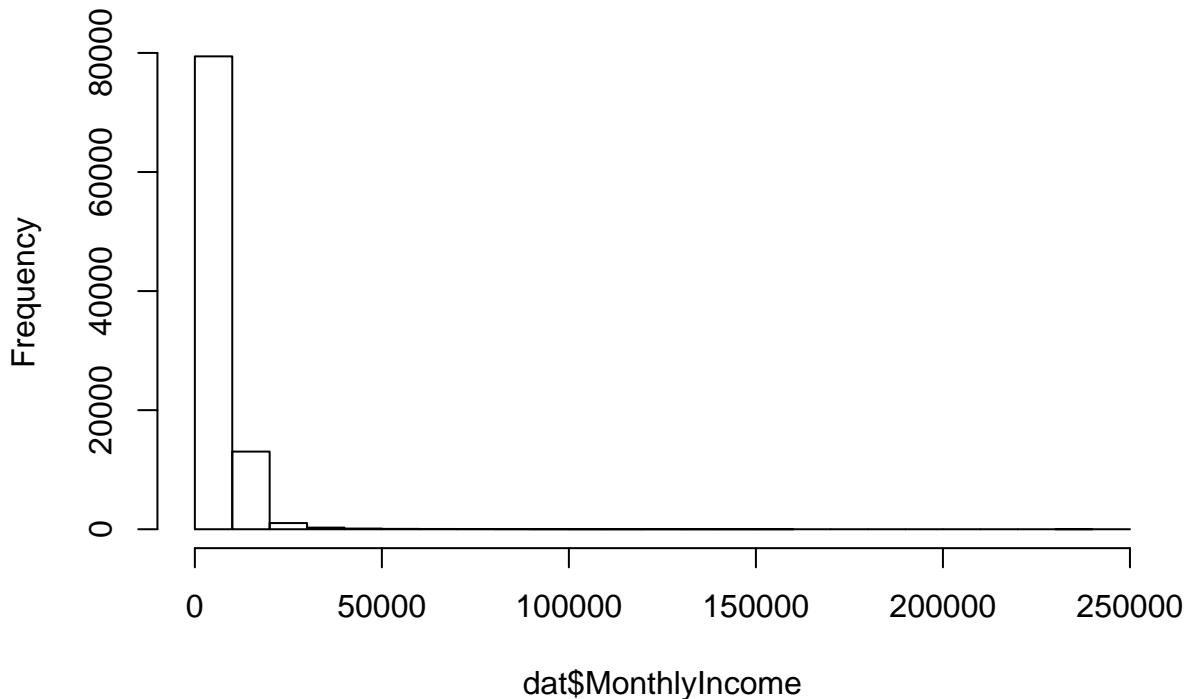
```
hist(dat$RealEstateLoansOrLines)
```

Histogram of dat\$RealEstateLoansOrLines



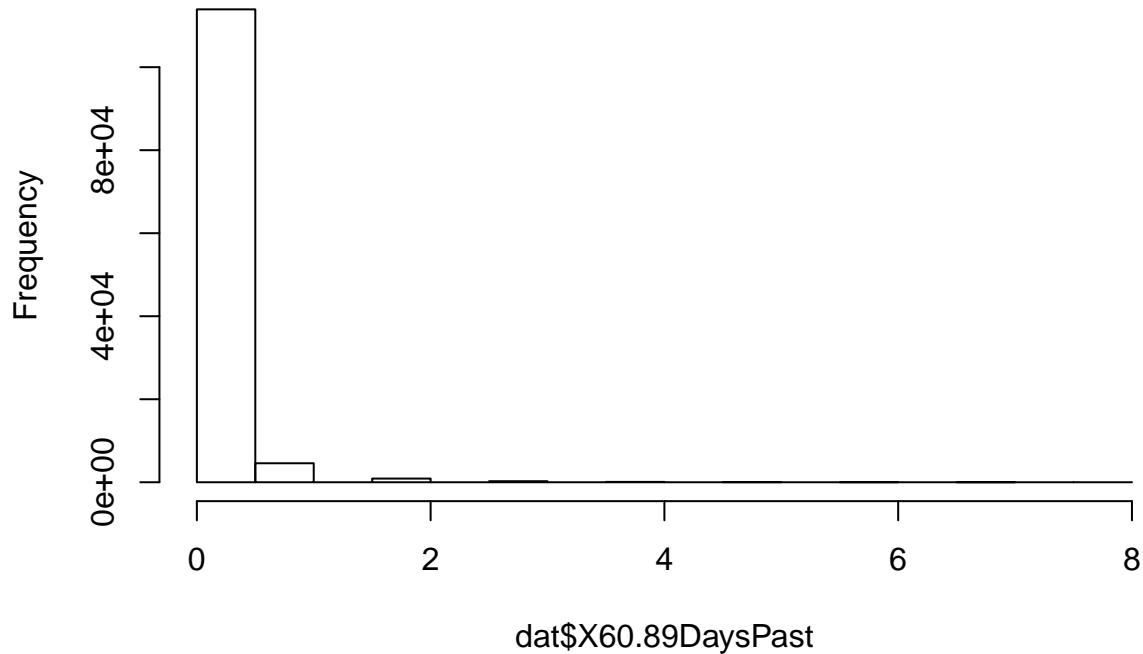
```
hist(dat$MonthlyIncome)
```

Histogram of dat\$MonthlyIncome



```
hist(dat$X60.89DaysPast)
```

Histogram of dat\$X60.89DaysPast



```

dat$ProportionOfUnsecuredLines[is.na(dat$ProportionOfUnsecuredLines)] = median(dat$ProportionOfUnsecuredLines)
dat$Age[is.na(dat$Age)] = mean(dat$Age[!is.na(dat$Age)])
dat$DebtRatio[is.na(dat$DebtRatio)] = mean(dat$DebtRatio[!is.na(dat$DebtRatio)])
dat$OpenCreditLinesAndLoans[is.na(dat$OpenCreditLinesAndLoans)] = mean(dat$OpenCreditLinesAndLoans[!is.na(dat$OpenCreditLinesAndLoans)])
dat$X90DaysLate[is.na(dat$X90DaysLate)] = median(dat$X90DaysLate[!is.na(dat$X90DaysLate)])
dat$NumberOfTime30.59DaysPast[is.na(dat$NumberOfTime30.59DaysPast)] = median(dat$NumberOfTime30.59DaysPast)
dat$RealEstateLoansOrLines[is.na(dat$RealEstateLoansOrLines)] = median(dat$RealEstateLoansOrLines[!is.na(dat$RealEstateLoansOrLines)])
dat$X60.89DaysPast[is.na(dat$X60.89DaysPast)] = median(dat$X60.89DaysPast[!is.na(dat$X60.89DaysPast)])
dat$Dependents[is.na(dat$Dependents)] = median(dat$Dependents[!is.na(dat$Dependents)])
dat$MonthlyIncome[is.na(dat$MonthlyIncome)] = median(dat$MonthlyIncome[!is.na(dat$MonthlyIncome)])

#Regressions Baum, nur wenn MonthlyIncome noch NAs hat

dataset = na.omit(dat)

set.seed(1)
train = sample_frac(dataset, 0.7)
sid = as.numeric(rownames(train)) # weil rownames() character zurück liefert
test = dataset[-sid,]

#Modell des Baums
mod_rpart = rpart(MonthlyIncome ~ .-DefaultLast2Years, data = train, method = "anova", control = rpart.control(minsplit=2))

https://stats.stackexchange.com/questions/5792/r-square-from-rpart-model?utm\_medium=organic&utm\_source=referral

```

```

tmp = printcp(mod_rpart)

##
## Regression tree:
## rpart(formula = MonthlyIncome ~ . - DefaultLast2Years, data = train,
##       method = "anova", control = rpart.control(minsplit = 30,
##                                                 minbucket = 1, cp = 0.001))
##
## Variables actually used in tree construction:
## [1] Age                      DebtRatio
## [3] Dependents               OpenCreditLinesAndLoans
## [5] ProportionOfUnsecuredLines RealEstateLoansOrLines
##
## Root node error: 2.4638e+12/84000 = 29330812
##
## n= 84000
##
##          CP nsplit rel error  xerror     xstd
## 1  0.0515622      0  1.00000 1.00003 0.053921
## 2  0.0371207      2  0.89688 0.91633 0.047815
## 3  0.0221266      3  0.85976 0.86880 0.045113
## 4  0.0162101      6  0.79025 0.80383 0.042294
## 5  0.0135838      7  0.77404 0.81730 0.045408
## 6  0.0120987      8  0.76045 0.79738 0.045926
## 7  0.0114635      9  0.74835 0.79788 0.045943
## 8  0.0108191     10  0.73689 0.79135 0.045840
## 9  0.0103505     11  0.72607 0.77751 0.045276
## 10 0.0067923     12  0.71572 0.76376 0.043908
## 11 0.0065242     13  0.70893 0.75875 0.043845
## 12 0.0055887     14  0.70240 0.75372 0.043749
## 13 0.0046241     15  0.69682 0.75871 0.044040
## 14 0.0045460     17  0.68757 0.75725 0.044103
## 15 0.0044450     18  0.68302 0.75359 0.043997
## 16 0.0041852     19  0.67858 0.75103 0.043934
## 17 0.0041412     20  0.67439 0.75144 0.043947
## 18 0.0035383     21  0.67025 0.74494 0.043759
## 19 0.0030656     22  0.66671 0.74543 0.044142
## 20 0.0026309     23  0.66365 0.74007 0.043844
## 21 0.0024194     24  0.66102 0.73972 0.043817
## 22 0.0023609     25  0.65860 0.74033 0.043830
## 23 0.0023134     26  0.65624 0.73929 0.043802
## 24 0.0022853     28  0.65161 0.73750 0.043719
## 25 0.0022060     29  0.64932 0.73537 0.043667
## 26 0.0021818     30  0.64712 0.73558 0.043669
## 27 0.0021748     36  0.63403 0.73512 0.043664
## 28 0.0020532     37  0.63185 0.73719 0.043986
## 29 0.0020065     38  0.62980 0.73652 0.043984
## 30 0.0019715     39  0.62779 0.73364 0.043991
## 31 0.0017096     41  0.62385 0.73193 0.044095
## 32 0.0016560     42  0.62214 0.72817 0.044036
## 33 0.0015713     43  0.62048 0.72782 0.043956
## 34 0.0015013     44  0.61891 0.72824 0.043964
## 35 0.0014334     45  0.61741 0.72850 0.043945
## 36 0.0014215     46  0.61598 0.72684 0.043931

```

```

## 37 0.0013588      47  0.61456 0.72627 0.043933
## 38 0.0012858      50  0.61048 0.72933 0.044016
## 39 0.0012313      51  0.60919 0.72844 0.043955
## 40 0.0011729      52  0.60796 0.72601 0.043389
## 41 0.0011559      53  0.60679 0.72643 0.043388
## 42 0.0011411      54  0.60563 0.72715 0.043604
## 43 0.0011373      60  0.59785 0.73064 0.043756
## 44 0.0011123      63  0.59444 0.73023 0.043749
## 45 0.0011104      64  0.59333 0.72950 0.043729
## 46 0.0010912      65  0.59222 0.72880 0.043722
## 47 0.0010747      66  0.59113 0.72854 0.043722
## 48 0.0010405      67  0.59005 0.72830 0.043790
## 49 0.0010162      68  0.58901 0.72773 0.043789
## 50 0.0010000      69  0.58800 0.72624 0.043739

rsq.val = 1-tmp[,c(3,4)]
rsq.val # r2 pro Anzahl der Splits = 1 - relativer Error am Split

```

```

##          rel.error        xerror
## 1  1.110223e-16 -2.552655e-05
## 2  1.031243e-01  8.367174e-02
## 3  1.402450e-01  1.311988e-01
## 4  2.097531e-01  1.961669e-01
## 5  2.259631e-01  1.827049e-01
## 6  2.395469e-01  2.026220e-01
## 7  2.516457e-01  2.021178e-01
## 8  2.631092e-01  2.086538e-01
## 9  2.739282e-01  2.224928e-01
## 10 2.842787e-01  2.362441e-01
## 11 2.910710e-01  2.412528e-01
## 12 2.975952e-01  2.462811e-01
## 13 3.031839e-01  2.412888e-01
## 14 3.124321e-01  2.427475e-01
## 15 3.169781e-01  2.464093e-01
## 16 3.214231e-01  2.489744e-01
## 17 3.256083e-01  2.485620e-01
## 18 3.297495e-01  2.550604e-01
## 19 3.332878e-01  2.545743e-01
## 20 3.363534e-01  2.599256e-01
## 21 3.389843e-01  2.602779e-01
## 22 3.414037e-01  2.596678e-01
## 23 3.437646e-01  2.607052e-01
## 24 3.483914e-01  2.624996e-01
## 25 3.506766e-01  2.646289e-01
## 26 3.528827e-01  2.644231e-01
## 27 3.659732e-01  2.648831e-01
## 28 3.681480e-01  2.628084e-01
## 29 3.702012e-01  2.634789e-01
## 30 3.722077e-01  2.663614e-01
## 31 3.761506e-01  2.680731e-01
## 32 3.778603e-01  2.718259e-01
## 33 3.795162e-01  2.721758e-01
## 34 3.810875e-01  2.717610e-01
## 35 3.825888e-01  2.714982e-01
## 36 3.840222e-01  2.731563e-01

```

```

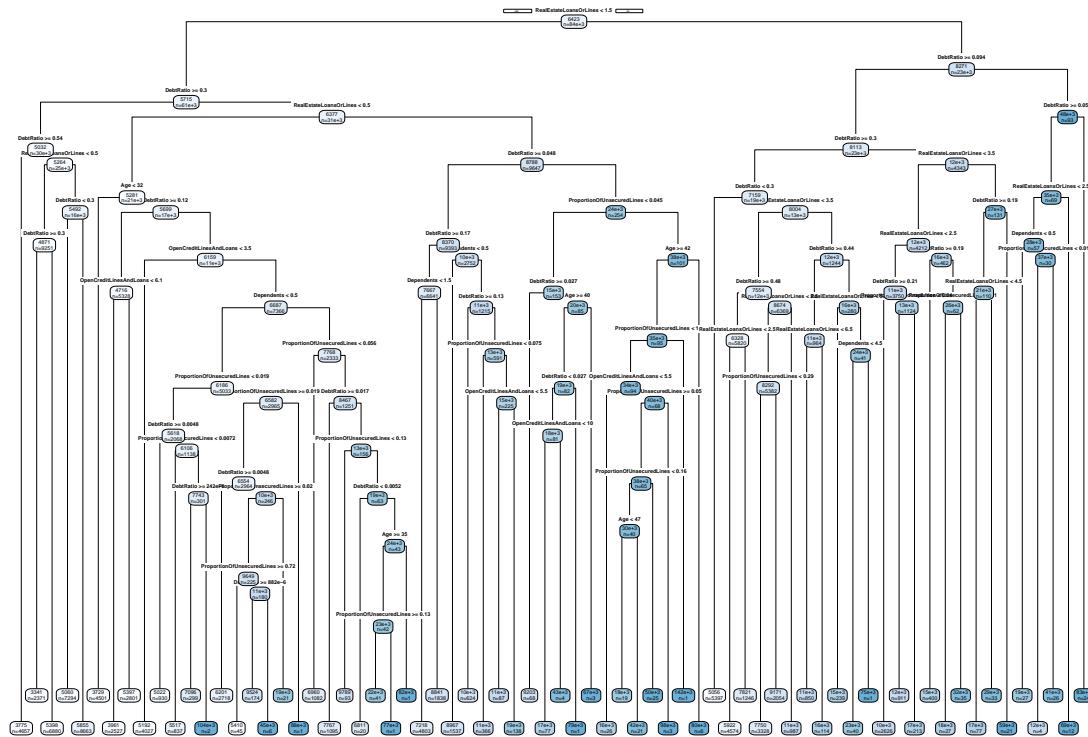
## 37 3.854437e-01 2.737268e-01
## 38 3.895199e-01 2.706680e-01
## 39 3.908058e-01 2.715553e-01
## 40 3.920371e-01 2.739888e-01
## 41 3.932100e-01 2.735651e-01
## 42 3.943659e-01 2.728527e-01
## 43 4.021467e-01 2.693619e-01
## 44 4.055587e-01 2.697742e-01
## 45 4.066710e-01 2.705015e-01
## 46 4.077815e-01 2.711973e-01
## 47 4.088726e-01 2.714648e-01
## 48 4.099473e-01 2.716958e-01
## 49 4.109879e-01 2.722733e-01
## 50 4.120041e-01 2.737640e-01

```

Plotten des Regression Baumes

```
rpart.plot(mod_rpart, type = 1, extra = 1)
```

Warning: labs do not fit even at cex 0.15, there may be some overplotting



4 Modeling

```

#Trainset und Testset mithilfe von Stratified Sampling bilden
#install.packages("fifer")
set.seed(1)

```

```

train = stratified(df = dat, group = "DefaultLast2Years", size = .7, select = NULL)
sid = as.numeric(rownames(train)) # weil rownames() character zurückliefert
test = dat[-sid,]

#CART Baum

#Modelln des Baums
mod_rpart = rpart(DefaultLast2Years ~ ., data = train, method = "class", control = rpart.control(minsplit=1))
mod_rpart

## n= 84000
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 84000 5624 0 (0.93304762 0.06695238)
##    2) X90DaysLate< 0.5 79475 3731 0 (0.95305442 0.04694558) *
##    3) X90DaysLate>=0.5 4525 1893 0 (0.58165746 0.41834254)
##      6) X90DaysLate< 1.5 2966 999 0 (0.66318274 0.33681726)
##        12) ProportionOfUnsecuredLines< 0.4963137 975 196 0 (0.79897436 0.20102564) *
##        13) ProportionOfUnsecuredLines>=0.4963137 1991 803 0 (0.59668508 0.40331492)
##          26) X60.89DaysPast< 0.5 1440 508 0 (0.64722222 0.35277778)
##            52) NumberOfTime30.59DaysPast< 1.5 1164 377 0 (0.67611684 0.32388316) *
##            53) NumberOfTime30.59DaysPast>=1.5 276 131 0 (0.52536232 0.47463768)
##              106) ProportionOfUnsecuredLines< 0.7719685 69 20 0 (0.71014493 0.28985507) *
##              107) ProportionOfUnsecuredLines>=0.7719685 207 96 1 (0.46376812 0.53623188)
##                214) DebtRatio< 0.302415 70 29 0 (0.58571429 0.41428571) *
##                215) DebtRatio>=0.302415 137 55 1 (0.40145985 0.59854015) *
##              27) X60.89DaysPast>=0.5 551 256 1 (0.46460980 0.53539020)
##                54) DebtRatio< 0.5694972 483 238 1 (0.49275362 0.50724638)
##                  108) X60.89DaysPast< 1.5 344 160 0 (0.53488372 0.46511628)
##                    216) Dependents< 2.5 283 123 0 (0.56537102 0.43462898) *
##                    217) Dependents>=2.5 61 24 1 (0.39344262 0.60655738) *
##                  109) X60.89DaysPast>=1.5 139 54 1 (0.38848921 0.61151079) *
##                    55) DebtRatio>=0.5694972 68 18 1 (0.26470588 0.73529412) *
##                7) X90DaysLate>=1.5 1559 665 1 (0.42655548 0.57344452)
##                  14) X60.89DaysPast< 0.5 894 442 1 (0.49440716 0.50559284)
##                    28) NumberOfTime30.59DaysPast< 0.5 443 191 0 (0.56884876 0.43115124)
##                      56) X90DaysLate< 2.5 276 101 0 (0.63405797 0.36594203)
##                        112) Age>=37.5 185 59 0 (0.68108108 0.31891892) *
##                        113) Age< 37.5 91 42 0 (0.53846154 0.46153846)
##                          226) DebtRatio< 0.1735258 47 14 0 (0.70212766 0.29787234) *
##                          227) DebtRatio>=0.1735258 44 16 1 (0.36363636 0.63636364) *
##                      57) X90DaysLate>=2.5 167 77 1 (0.46107784 0.53892216)
##                        114) MonthlyIncome>=9250 7 0 0 (1.00000000 0.00000000) *
##                        115) MonthlyIncome< 9250 160 70 1 (0.43750000 0.56250000)
##                          230) Age>=55.5 14 3 0 (0.78571429 0.21428571) *
##                          231) Age< 55.5 146 59 1 (0.40410959 0.59589041) *
##                      29) NumberOfTime30.59DaysPast>=0.5 451 190 1 (0.42128603 0.57871397)
##                        58) DebtRatio< 0.131328 107 44 0 (0.58878505 0.41121495) *
##                        59) DebtRatio>=0.131328 344 127 1 (0.36918605 0.63081395) *
##                      15) X60.89DaysPast>=0.5 665 223 1 (0.33533835 0.66466165)

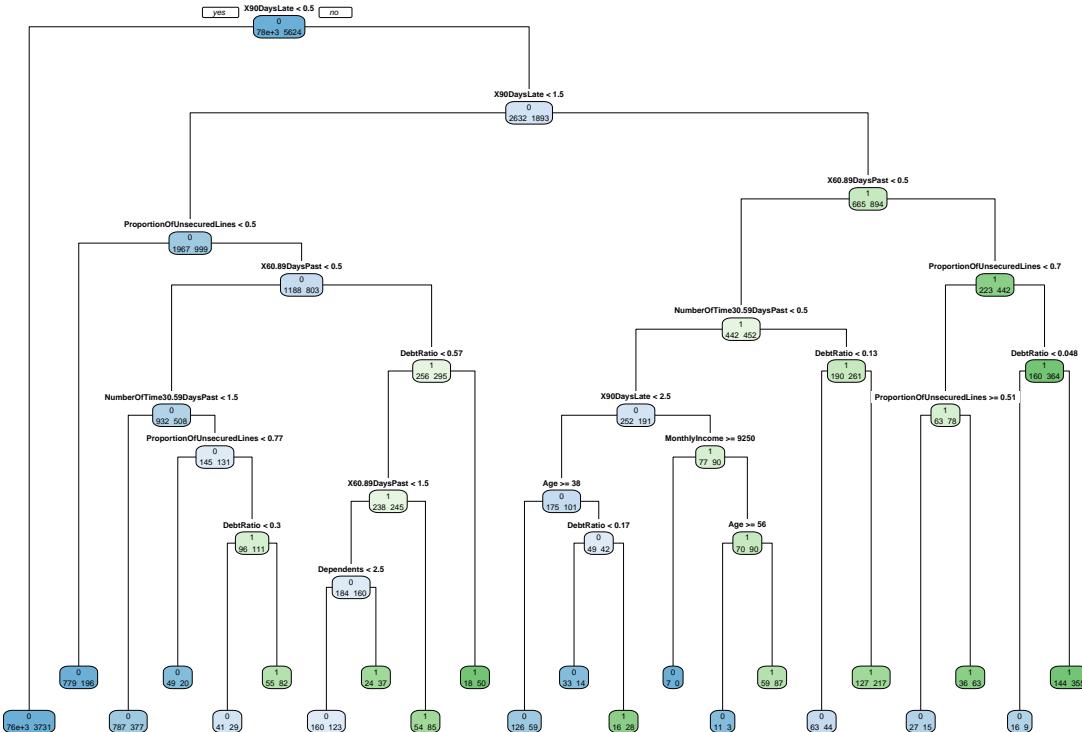
```

```

##      30) ProportionOfUnsecuredLines< 0.6968037 141    63 1 (0.44680851 0.55319149)
##      60) ProportionOfUnsecuredLines>=0.5071184 42     15 0 (0.64285714 0.35714286) *
##      61) ProportionOfUnsecuredLines< 0.5071184 99     36 1 (0.36363636 0.63636364) *
##      31) ProportionOfUnsecuredLines>=0.6968037 524    160 1 (0.30534351 0.69465649)
##      62) DebtRatio< 0.04798866 25      9 0 (0.64000000 0.36000000) *
##      63) DebtRatio>=0.04798866 499    144 1 (0.28857715 0.71142285) *

# Plotten des CART Baumes
rpart.plot(mod_rpart, type = 1, extra = 1)

```



```

# Modell auf Testmenge anwenden
prediction = as.character(predict(mod_rpart, test[,-1], type = "class"))
predicted = cbind.data.frame(observed = test$DefaultLast2Years, prediction = prediction)

# Confusion Matrix fÃ¼r Testmenge erstellen
cm = table(predicted$observed, predicted$prediction)
cm

```

```

##
##          0      1
##      0 48748   382
##      1 3027    597

```

```

# Berechnen der Accuracy auf Basis der Confusion Matrix (cm)
accuracy = (cm[1,1] + cm[2,2]) / sum(cm)
accuracy

```

```

## [1] 0.9353793

```

```

#Random Forest

rf = randomForest(DefaultLast2Years ~., data=train, ntree=50, nodesize=10, maxnodes=NULL, importance = TRUE)

# Modell auf Testmenge anwenden
prediction = as.character(predict(rf, test[,-1], type = "class"))
predicted = cbind.data.frame(observed = test$DefaultLast2Years, prediction = prediction)

# Confusion Matrix für Testmenge erstellen
cm = table(predicted$observed,predicted$prediction)
cm

## 
##          0      1
## 0 48908   222
## 1  2306 1318

# Berechnen der Accuracy auf Basis der Confusion Matrix (cm)
accuracy = (cm[1,1] + cm[2,2]) / sum(cm)
accuracy

## [1] 0.9520795

```

5 Evaluation

```

#ROC und AUC für cart baum

prognose = predict(mod_rpart, newdata = test, type="prob")
test$PD = prognose[, which(colnames(prognose) == 0)]

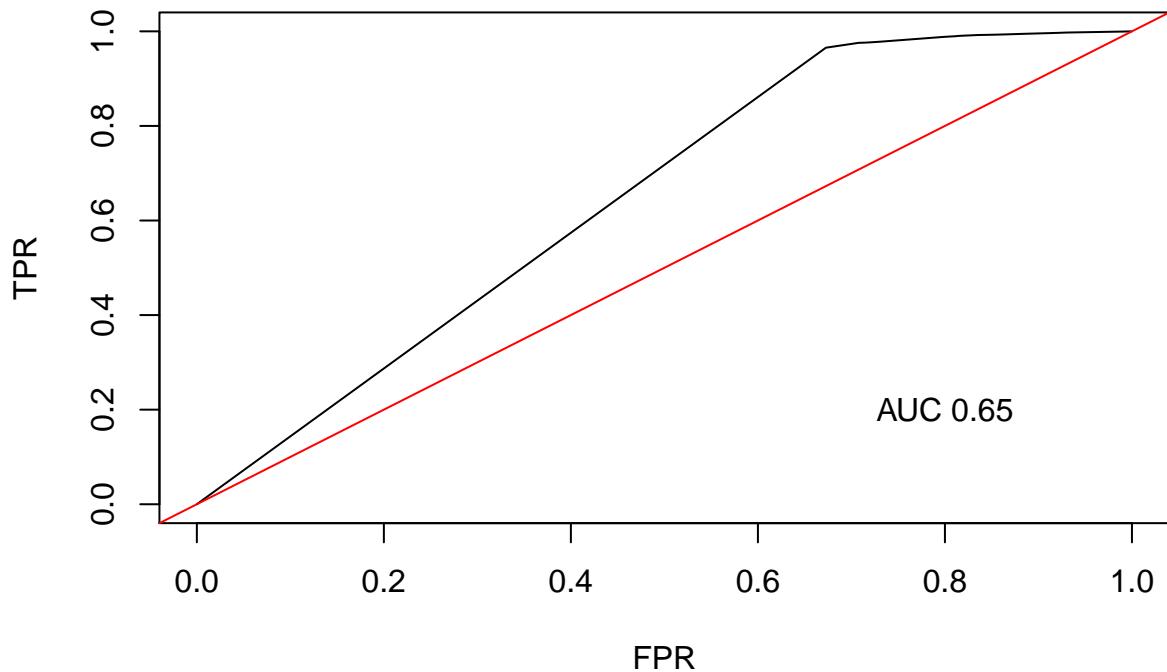
dat_copy = test
data_roc = prediction(dat_copy$PD, dat_copy$DefaultLast2Years == 0)

roc_curve = performance(data_roc, measure = "tpr", x.measure = "fpr")
AUC = performance(data_roc, measure = "auc")@y.values[[1]]

plot(roc_curve, xlab = "FPR", ylab = "TPR", main ="ROC-Kurve")
abline(a = 0, b = 1, col="red")
text(0.8, 0.2, paste("AUC", toString(round(AUC, digits = 2)), sep = " "))

```

ROC-Kurve



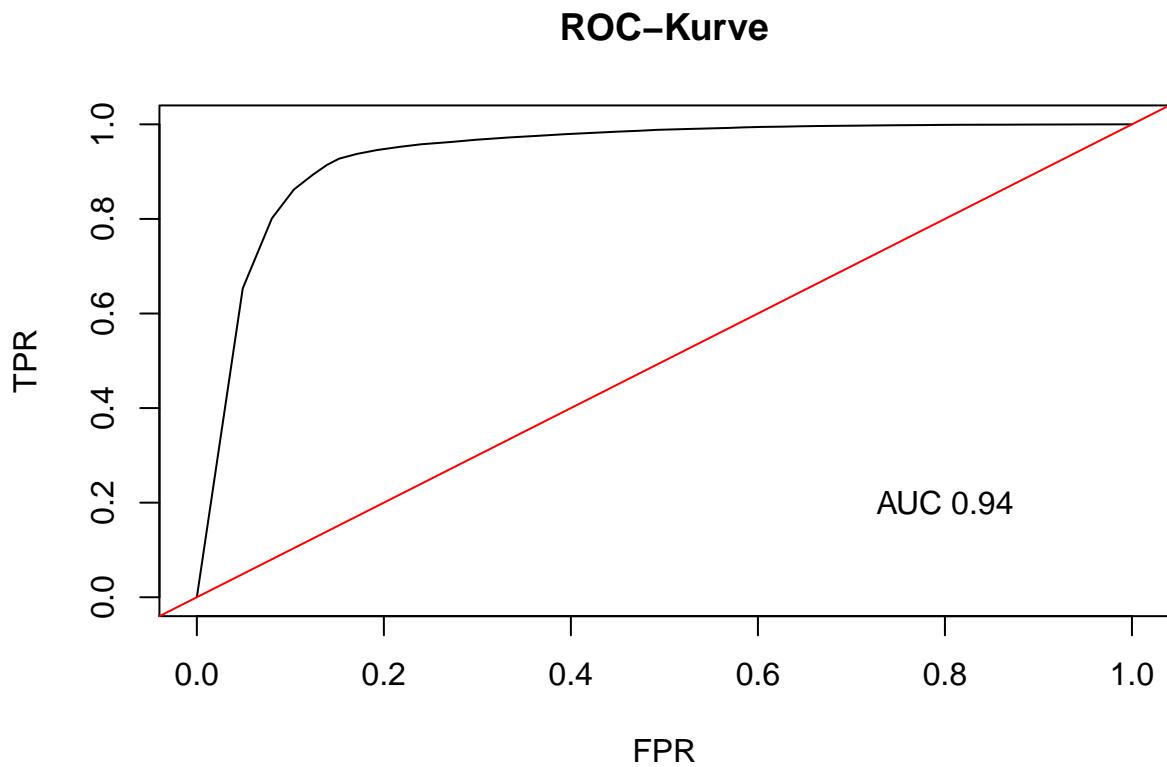
```
#ROC und AUC für randomForest

prognose = predict(rf, newdata = test, type="prob")
test$PD = prognose[, which(colnames(prognose) == 0)]

dat_copy = test
data_roc = prediction(dat_copy$PD, dat_copy$DefaultLast2Years == 0)

roc_curve = performance(data_roc, measure = "tpr", x.measure = "fpr")
AUC = performance(data_roc, measure = "auc")@y.values[[1]]

plot(roc_curve, xlab = "FPR", ylab = "TPR", main ="ROC-Kurve")
abline(a = 0, b = 1, col="red")
text(0.8, 0.2, paste("AUC", toString(round(AUC, digits = 2)), sep = " "))
```



```
# bei AUC und accuracy ist randomforest besser.
```

6 Deployment

```
#XML bauen
#install.packages("pmml")

#rf_pmml = pmml(rf)
#saveXML(rf_pmml, "C:/Users/Lenovo/Downloads/rf_pmml.xml")
#rpart_pmml=pmml(mod_rpart)
#saveXML(rpart_pmml, "C:/Users/Lenovo/Downloads/cart_pmml.xml")
```

7 Ergebnisse Cart: Accuracy = 0.9354, AUC = 0.65

8 Ergebnisse rf: Accuracy = 0.9520 , AUC = 0.94

Wir würden den Random Forest empfehlen, weil sowohl die Accuracy als auch der AUC höher sind.